A Faster Interior-Point Method for Sum-Of-Squares Optimization

Shunhua Jiang
Columbia University, New York, NY, USA

Bento Natura
London School of Economics, UK

Omri Weinstein
The Hebrew University, Jerusalem, Israel
Columbia University, New York, NY, USA

Abstract

We present a faster interior-point method for optimizing sum-of-squares (SOS) polynomials, which are a central tool in polynomial optimization and capture convex programming in the Lasserre hierarchy. Let $p = \sum_i q_i^2$ be an n-variate SOS polynomial of degree $2d$. Denoting by $L := \binom{n+d}{d}$ and $U := \binom{n+2d}{2d}$ the dimensions of the vector spaces in which q_i's and p live respectively, our algorithm runs in time $\tilde{O}(LU^{1.87})$. This is polynomially faster than state-of-art SOS and semidefinite programming solvers [16, 15, 27], which achieve runtime $\tilde{O}(L^{0.5}\min\{U^{2.37}, L^{4.24}\})$.

The centerpiece of our algorithm is a dynamic data structure for maintaining the inverse of the Hessian of the SOS barrier function under the *polynomial interpolant basis* [27], which efficiently extends to multivariate SOS optimization, and requires maintaining spectral approximations to low-rank perturbations of *elementwise (Hadamard) products*. This is the main challenge and departure from recent IPM breakthroughs using inverse-maintenance, where low-rank updates to the slack matrix readily imply the same for the Hessian matrix.

2012 ACM Subject Classification Mathematics of computing → Continuous functions; Mathematics of computing → Convex optimization; Mathematics of computing → Semidefinite programming; Mathematics of computing → Stochastic control and optimization

Keywords and phrases Interior Point Methods, Sum-of-squares Optimization, Dynamic Matrix Inverse

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.79

Category Track A: Algorithms, Complexity and Games

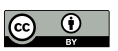
Related Version *Full Version*: `https://arxiv.org/abs/2202.08489`

Funding *Shunhua Jiang*: Supported by NSF CAREER award CCF-1844887.
Bento Natura: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 757481–ScaleOpt).
Omri Weinstein: Supported by NSF CAREER award CCF-1844887 and ISF grant #3011005535.

Acknowledgements The second author would like to thank Vissarion Fisikopoulos and Elias Tsigaridas for introducing him from a practical perspective to Sum-of-Squares Optimization under the interpolant basis.

1 Introduction

Polynomial optimization is a fundamental problem in many areas of applied mathematics, operations research, and theoretical computer science, including combinatorial optimization [5, 36, 4], statistical estimation [13, 14], experimental design [26], control theory [12], signal



processing [31], power systems engineering [11], discrete geometry [2, 3] and computational algebraic geometry [20]. In the most basic formulation, we are given a collection of k real n-variate polynomials $f_1, \cdots, f_k$ and an objective function $f : \mathbb{R}^n \to \mathbb{R}$, and the goal is to minimize f over the set $\mathcal{S} := \{t \in \mathbb{R}^n \mid \forall i \in \{1, \cdots, k\} : f_i(t) \geq 0\}$, that is, to find

$$\inf_{t \in \mathbb{R}^n} \{f(t) \mid t \in \mathcal{S}\}, \tag{1}$$

which is equivalent to checking polynomial nonnegativity $\sup_{c \in \mathbb{R}}\{c \mid f(t) - c \geq 0,\ \forall t \in \mathcal{S}\}$. This is then equivalent to computing $\sup_{c \in \mathbb{R}}\{c \mid f - c \in \mathcal{K}(\mathcal{S})\}$, where $\mathcal{K}(\mathcal{S})$ denotes the convex cone of all polynomials of degree at most $\deg(f)$ that are non-negative on the set $\mathcal{S}$. This is an instance of the more general conic programming:

$$\min_{x \in \mathbb{R}^N} c^\top x \text{ s.t. } Ax = b, x \in \mathcal{K}, \tag{2}$$

where $\mathcal{K} \subset \mathbb{R}^N$ is some convex cone[1]. The conic optimization problem over the cone $\mathcal{K}(\mathcal{S})$ is intractable in general because there is no simple characterization of $\mathcal{K}(\mathcal{S})$. Nevertheless, there always exists an increasing family of convex cones of *weighted sum-of-squares* polynomials that converges to any such cone $\mathcal{K}(\mathcal{S})$.

We first introduce the notion of *sum-of-squares* (SOS) polynomials: Denoting by $\mathcal{V}_{n,d}$ the vector space of all n-variate polynomials of (total) degree at most d, a polynomial $p \in \mathcal{V}_{n,2d}$ is said to be *sum-of-squares* (SOS) if it can be written as a finite sum of square polynomials, i.e., there exist $q_1, \cdots, q_\ell$ such that $p = \sum_{i=1}^{\ell} q_i^2$. The set $\Sigma_{n,2d}$ of SOS polynomials of degree at most $2d$ is a (proper) cone contained in $\mathcal{V}_{n,2d}$, of dimension $U := \dim(\mathcal{V}_{n,2d}) = \binom{n+2d}{2d}$, as the vector space $\mathcal{V}_{n,2d}$ is isomorphic to $\mathbb{R}^U$. If p can be written as $p = \sum_{i=1}^{k} f_i s_i$ for $s_1 \in \Sigma_{n,2d_1}, \cdots, \Sigma_{n,2d_k}$ and k nonzero polynomials $\mathbf{f} := (f_1, \cdots, f_k)$, then it is said to be *weighted sum-of-squares* (WSOS).

Putinar's Positivstellensatz [29] states that under mild conditions, any polynomial p that is non-negative on $\mathcal{S}$ can be written as a WSOS polynomial $\sum_{i=1}^{k} f_i s_i$, albeit with (potentially) *unbounded degree* s_i's. In WSOS optimization we consider sum-of-squares polynomials s_i with bounded degree, so the hierarchy of WSOS optimization with increasing degree (known as the Lasserre hierarchy) can be viewed as a tool for approximating general polynomial optimization. For more details of this approximation scheme for polynomial optimization, we refer the readers to the textbooks [19, 7].

This paper concerns algorithms for *(W)SOS optimization*, which is the conic optimization program (2) where the underlying cone $\mathcal{K}$ is the (W)SOS cone:

$$\min_{x \in \mathbb{R}^U} c^\top x \text{ s.t. } Ax = b, x \in \Sigma_{n,2d}, \tag{3}$$

where $x \in \mathcal{V}_{n,2d}$ is the vector of coefficients which encodes the polynomial. Henceforth, we focus on the case where $\mathcal{K} = \Sigma_{n,2d}$ is the SOS cone. See our full version for how to extend our algorithm for SOS optimization to WSOS.

The computational complexity of solving Problem 3 naturally depends on the dimensions

$$L := \dim(\mathcal{V}_{n,d}) = \binom{n+d}{d} \quad , \quad U := \dim(\mathcal{V}_{n,2d}) = \binom{n+2d}{2d} \tag{4}$$

of the underlying vector spaces (Note that $L \leq U \leq L^2$). We now turn to explain the previous approaches for SOS optimization solvers.

[1] A subset $\mathcal{K} \subset \mathbb{R}^N$ is a convex cone if $\forall\, x, y \in \mathcal{K}$ and $\alpha, \beta \in \mathbb{R}_+$, $\alpha x + \beta y \in \mathcal{K}$.

SOS Optimization as SDPs

A fundamental fact is that the *dual* SOS cone is a *slice of the SDP cone* [24]. More formally, for any *fixed bases* $\mathbf{p} = (p_1, p_2, \cdots, p_L)$ and $\mathbf{q} = (q_1, q_2, \cdots, q_U)$ to $\mathcal{V}_{n,d}$ and $\mathcal{V}_{n,2d}$ respectively, there exists a unique linear mapping $\Lambda : \mathbb{R}^U \to \mathbb{R}^{L\times L}$ satisfying

$$\Lambda(\mathbf{q}(t)) = \mathbf{p}(t)\mathbf{p}(t)^\top, \quad \forall t \in \mathbb{R}^n. \tag{5}$$

Here we define $\mathbf{p}(t) = (p_1(t), p_2(t), \cdots, p_L(t))^\top$ and $\mathbf{q}(t) = (q_1(t), q_2(t), \cdots, q_U(t))^\top$. An equivalent way to view the definition of Λ in (5) is as follows: For polynomials $p_i, p_j \in \mathbf{p}$ there are unique coefficients λ_{iju} such that $p_i p_j = \sum_{u\in U} \lambda_{iju} q_u$. These λ_{iju} define the mapping Λ unambiguously.

This in turn implies that a polynomial $s \in \mathcal{V}_{n,2d}$ (we view s as a vector in $\mathbb{R}^U$ that corresponds to its coefficients over the basis $\mathbf{q}$) is in the dual SOS cone $\Sigma^*_{n,2d}$ if and only if $\Lambda(s)$ is a *positive semidefinite* (PSD) matrix (proved by [24], see Theorem 10 for details). As [27] recently observed, the choice of the bases $\mathbf{p}, \mathbf{q}$ crucially affects the complexity of the optimization problem, more on this below.

Equation (5) implies the well-known fact that optimization over SOS polynomials (3) can be reduced to semidefinite programming

$$\min_{X\succeq 0}\{\langle C, X\rangle \mid \mathrm{tr}(A_i X) = b_i, \ \forall i \in [m]\}, \tag{SDP}$$

and can thus be solved using off-the-shelf SDP solvers. However, despite recent breakthroughs on the runtime of general SDP solvers via *interior-point methods* (IPMs) [16, 15], this SDP reformulation of (3) does not scale well for moderately large degrees, i.e., whenever $U \ll L^2$ in (4). This is because the SDP reformulation always incurs a factor of at least L^2, even when $U \ll L^2$, as this is the SDP variable size (the PSD matrix X has size $L \times L$). Indeed, for the current fast-matrix-multiplication (FMM) exponent $\omega \approx 2.37$ [21, 1], the running time of state-of-the-art SDP solvers [16, 15] for SOS optimization (Problem 3) is[2]

$$\widetilde{O}\left(L^{0.5} \cdot \min\{UL^2 + U^{2.37},\ L^{4.24}\}\right). \tag{6}$$

An alternative approach is to solve Problem 3 directly by designing an *ad-hoc* IPM for the dual SOS cone, avoiding the blowup in the SDP reformulation. This was exactly the motivation of [27]. Like all aforementioned SDP solvers, [27]'s SOS solver is based on IPMs [25], which iteratively minimize the original objective function plus a barrier function via Newton steps. When applied to the SOS Problem (3), *the choice of the specific bases* $\mathbf{p}, \mathbf{q}$ *crucially affects the structure of the (Hessian of the) barrier function* $F(s) = F(\Lambda(s))$, and hence the cost-per-iteration of the IPM. As such, choosing a "good" and efficient basis is key to a fast algorithm for (3). One of the main contribution of [27] is an efficient basis for the SOS cone, which efficiently scales to multivariate SOS, yielding an IPM whose total runtime is

$$\widetilde{O}\left(L^{0.5}U^{\omega}\right) \approx \widetilde{O}\left(L^{0.5}U^{2.37}\right). \tag{7}$$

Our main result is a polynomially faster IPM for Problem 3:

▶ **Theorem 1** (Main Result, Informal version of Theorem 16). *With current FMM exponent, there is an algorithm for solving Problem* (3)*, whose total running time is* $\widetilde{O}\left(LU^{1.87}\right)$.

Indeed, this runtime is polynomially faster than (7) and (6), as shown in Figure 1. We now turn to elaborate on the technical approach for proving Theorem 1.

[2] We use $\widetilde{O}(\cdot)$ to hide $U^{o(1)}$ and $\log(1/\delta)$ factors.

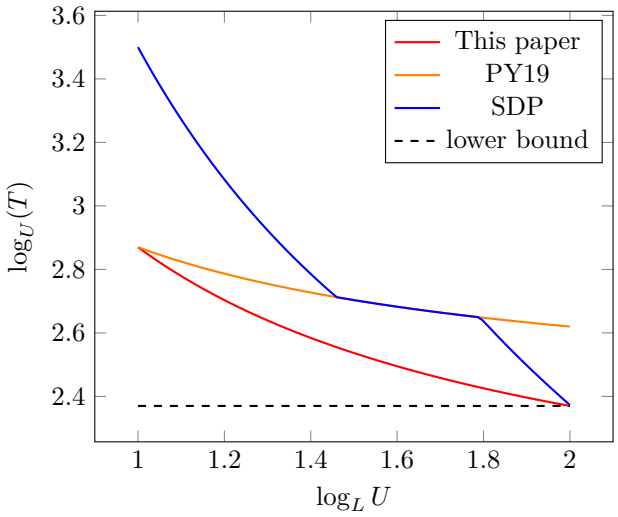


Figure 1 Overview of current running times of recent solvers for SOS. The lower bound bound stems from solving a linear system in U variables, i.e., $T = \Omega(U^\omega)$ where $\omega \approx 2.37$.

Faster IPMs via Inverse-Maintenance

Interior-Point Methods (IPMs [18, 30]) are a powerful class of second-order optimization algorithms for convex optimization, which essentially reduce a conic optimization problem (2) to solving *a sequence of slowly-changing linear systems* (via Newton steps). Since their discovery in the mid 80's, IPMs have emerged as the "gold-standard" of convex optimization, as they are known to converge fast in *both theory and practice* [35]. The main computational cost of IPMs is computing, in each iteration, the inverse of the Hessian of the underlying *barrier function* $F(s) = F(\Lambda(s))$, which naively costs at least U^ω time per iteration for the SOS optimization problem [27]. A recent influential line of work [9, 16], inspired by [37]'s seminal work, has demonstrated that *dynamically maintaining* the inverse of the Hessian matrix under *low-rank* updates using clever *data structures*, can lead to much cheaper *cost-per-iteration*. All of these results rely on a careful combination of dynamic data structures with the geometry (e.g., spectral approximation) of the underlying optimization method and barrier function. Similar techniques have been extended to other optimization problems as well [23, 38, 17, 33, 34]. This paper extends this line of work to SOS optimization.

Our Techniques

Following the framework of [27], we also choose the *polynomial interpolant basis* representation, which corresponds to a linear operator $\Lambda : \mathbb{R}^U \to \mathbb{R}^{L \times L}$ is $\Lambda(s) = P^\top \operatorname{diag}(s) P$, where $P \in \mathbb{R}^{U \times L}$ is the matrix whose entries are the evaluation of the Lagrange interpolation polynomials, through some unisolvent[3] set of points in $\mathcal{V}_{n,d}$ (see Section 3 for a formal definition). This basis induces the aforementioned convenient form of Λ, and generalizes to the multivariate case. The Hessian of the barrier function $F(s) = -\log\det(\Lambda(s))$ is given by

$$H(s) = \left(P(P^\top \operatorname{diag}(s) P)^{-1} P^\top\right)^{\circ 2} \in \mathbb{R}^{U \times U},$$

[3] Any set of points in $\mathbb{R}^n$ for which the evaluation of a polynomial in $\mathcal{V}_{n,d}$ on these points uniquely defines the polynomial.

where $A \circ B$ denotes the element-wise (Hadamard) product of two matrices. The main bottleneck of each iteration of IPMs is to compute the Hessian inverse $H(x)^{-1}$ of the Newton step, which naïvely takes $O(U^\omega)$ time.

In IPM theory, it has long been known that it suffices to compute a spectral approximation of the Hessian. We follow the "lazy update" framework in recent developments of LP and SDP solvers [9, 16], which batches together low-rank updates to $M := P(P^\top \operatorname{diag}(s)P)^{-1}P^\top$, where $\operatorname{rk}(M) = L$. In each iteration, we can compute a spectral approximation $M^{\text{new}} = M + UV^\top$, where U, V are low rank matrices with size $U \times r$ where $r \ll U$ is chosen to optimize the runtime. Since $\widetilde{M} \approx M$ implies that $\widetilde{M}^{\circ 2} \approx M^{\circ 2}$, this also gives a spectral approximation of the Hessian.

The main challenge here, compared to previous LP and SDP solvers [37, 22, 9, 16, 15], is that low-rank updates to M do not readily translate to a low-rank update to $(M^{\circ 2})^{-1}$, since Hadamard-products can *increase* the rank $\operatorname{rk}(A \circ B) \leq \operatorname{rk}(A) \cdot \operatorname{rk}(B)$, in contrast to standard matrix multiplication which does not increase the rank $\operatorname{rk}(AB) \leq \max\{\operatorname{rk}(A), \operatorname{rk}(B)\}$. This means that we cannot directly apply Woodbury's identity to efficiently update the inverse of the Hessian, which is the common approach in all aforementioned works. Instead, we employ the following property which relates rank-one Hadamard-product perturbations to standard matrix products

$$M \circ (u \cdot v^\top) = \operatorname{diag}(u) \cdot M \cdot \operatorname{diag}(v),$$

which means that we can translate the rank-r update of M into a rank-Lr update of $M^{\circ 2}$ for $r \leq L$. With some further calculations, applying Woodbury's identity on the resulting matrix, implies that we can compute $((M^{\text{new}})^{\circ 2})^{-1}$ in time

$$O\big(\mathcal{T}_{\text{mat}}(U, U, Lr)\big),$$

which is never worse than $\mathcal{T}_{\text{mat}}(U, U, U) = U^\omega$ as long as $r \leq U/L$. Modifying the amortization tools of [16] and [15], combined with basic spectral theory for Hadamard products, we show that our amortized cost per iteration is bounded by

$$O\big(U^2 + U^{\omega - 1/2} \cdot L^{1/2}\big),$$

which becomes $O\big(U^2 + U^{1.87} L^{0.5}\big)$ if we plug in the current matrix multiplication exponent.

2 Preliminaries

In this section we provide the definitions and the tools that we will use. For any integer $n > 0$, we define $[n] = \{1, 2, \cdots, n\}$. We use $\mathbb{R}_+$ and $\mathbb{R}_{\geq 0}$ to denote the set of positive and non-negative real numbers respectively. We use $0_n, 1_n \in \mathbb{R}^n$ to denote the all-zero and all-one vectors of size n.

Given a vector $v \in \mathbb{R}^n$, for any $m \leq n$, we use $v_{[:m]} \in \mathbb{R}^m$ to denote the first m entries of v. For a vector $v \in \mathbb{R}^n$, we use $\operatorname{diag}(v) \in \mathbb{R}^{n \times n}$ to denote the diagonal matrix whose diagonal entries are v. For a square matrix $A \in \mathbb{R}^{n \times n}$, we use $\operatorname{diag}(A) \in \mathbb{R}^n$ to denote the vector of the diagonal entries of A. We use $\operatorname{rk}(A)$ to denote the rank of a matrix A. We use $\ker(A)$ and $\operatorname{Im}(A)$ to denote the kernel space and the column space of A.

We say a matrix $A \in \mathbb{R}^{n \times n}$ is PSD (denoted as $A \succeq 0$) if A is symmetric and $x^\top A x \geq 0$ for all $x \in \mathbb{R}^n$. We use $\mathbb{S}^{n \times n}$ to denote the set of PSD matrices of size $n \times n$. The spectral norm of a matrix $A \in \mathbb{R}^{n \times d}$ is defined as $\|A\|_2 = \max_{x \in \mathbb{R}^d, \|x\|_2 = 1} \|Ax\|_2$. The Frobenius norm of A is defined as $\|A\|_F = \sqrt{\sum_{i \in [n]} \sum_{j \in [d]} A_{i,j}^2}$. For any PSD matrix $M \in \mathbb{S}^{n \times n}$, we define the M-norm as $\|x\|_M = \sqrt{x^\top M x}$, $\forall x \in \mathbb{R}^n$.

We use $\mathcal{T}_{\mathrm{mat}}(a,b,c)$ to denote the time to multiply two matrices of sizes $a \times b$ and $b \times c$. A basic fact of fast matrix multiplication is that $\mathcal{T}_{\mathrm{mat}}(a,b,c) = \mathcal{T}_{\mathrm{mat}}(b,c,a) = \mathcal{T}_{\mathrm{mat}}(c,a,b)$ (see e.g. [6]), and we will use these three terms interchangeably.

▶ **Fact 2** (Woodbury identity). *Let $A \in \mathbb{R}^{n\times n}, C \in \mathbb{R}^{k\times k}, U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{k\times n}$ where A and C are invertible, then*

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

▶ **Definition 3** (Hadamard product). *For any two matrices $A, B \in \mathbb{R}^{m\times n}$, the Hadamard product $A \circ B$ is defined as*

$$(A\circ B)_{i,j} = A_{i,j}\cdot B_{i,j}, \quad \forall i \in [m], j \in [n].$$

We also use $A^{\circ 2}$ to denote $A \circ A$.

The Hadamard product has the following properties (the proofs are straightforward).

▶ **Fact 4** (Properties of Hadamard product). *For matrices $A, B \in \mathbb{R}^{m\times n}$, and vectors $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$, we have the following properties.*
1. $x^\top (A\circ B)y = \mathrm{tr}[\mathrm{diag}(x)A\,\mathrm{diag}(y)B^\top]$,
2. $A\circ(x\cdot y^\top) = \mathrm{diag}(x)\cdot A\cdot \mathrm{diag}(y)$.

▶ **Definition 5** (Spectral approximation). *For any two symmetric matrices $A, \widetilde{A} \in \mathbb{R}^{n\times n}$, any parameter $\epsilon \in (0,1)$, we say $\widetilde{A}$ and A are ϵ-spectral approximation of each other, denoted as $\widetilde{A} \approx_\epsilon A$, if we have*

$$e^{-\epsilon}\cdot x^\top Ax \le x^\top \widetilde{A}x \le e^{\epsilon}\cdot x^\top Ax, \quad \forall x \in \mathbb{R}^n.$$

Spectral approximation has the following properties.

▶ **Fact 6** (Properties of spectral approximation). *For any two PSD matrices $A, \widetilde{A} \in \mathbb{R}^{n\times n}$, any parameter $\epsilon \in (0,1)$, if $\widetilde{A} \approx_\epsilon A$, then we have*
1. *$B^\top AB \approx_\epsilon B^\top \widetilde{A}B$, for any matrix $B \in \mathbb{R}^{n\times n}$.*
2. *If both A and $\widetilde{A}$ are invertible, then $A^{-1} \approx_\epsilon \widetilde{A}^{-1}$.*
3. $e^{-\epsilon}\,\mathrm{tr}[A] \le \mathrm{tr}[\widetilde{A}] \le e^{\epsilon}\,\mathrm{tr}[A]$.
4. $\widetilde{A}^{\circ 2} \approx_{2\epsilon} A^{\circ 2}$.

Proof. The proofs of the first three claims are straightforward. We only prove the last claim.

For any vector $x \in \mathbb{R}^n$, we have

$$\begin{aligned}
x^\top A^{\circ 2}x =\ & \mathrm{tr}[\mathrm{diag}(x)A\,\mathrm{diag}(x)A] \\
=\ & \mathrm{tr}[A^{1/2}\,\mathrm{diag}(x)A\,\mathrm{diag}(x)A^{1/2}] \\
\le\ & e^{\epsilon}\cdot \mathrm{tr}[A^{1/2}\,\mathrm{diag}(x)\widetilde{A}\,\mathrm{diag}(x)A^{1/2}] \\
=\ & e^{\epsilon}\cdot \mathrm{tr}[\widetilde{A}^{1/2}\,\mathrm{diag}(x)A\,\mathrm{diag}(x)\widetilde{A}^{1/2}] \\
\le\ & e^{2\epsilon}\cdot \mathrm{tr}[\widetilde{A}^{1/2}\,\mathrm{diag}(x)\widetilde{A}\,\mathrm{diag}(x)\widetilde{A}^{1/2}] = e^{2\epsilon}\cdot x^\top \widetilde{A}^{\circ 2}x
\end{aligned}$$

where the first step follows from Fact 4, the second and the fourth steps follow from the trace invariance under cyclic permutations and the fact that $A^{1/2}$ exists when A is PSD, the third and the fifth steps follow from Part 3 of this fact.

Similarly we can prove $x^\top A^{\circ 2}x \ge e^{-2\epsilon}\cdot x^\top \widetilde{A}^{\circ 2}x$. Thus we have $A^{\circ 2} \approx_{2\epsilon} \widetilde{A}^{\circ 2}$. ◀

3 Background of sum-of-squares optimization

In this section we provide the background of sum-of-squares optimization. We refer the readers to [28, 27] for more details.

▶ **Definition 7** (Polynomial space). *We use $\mathcal{V}_{n,d}$ to denote the set of n-variate polynomials over the reals of degree at most d, where the degree means the total degree, i.e., the degree of $x_1^{d_1}\cdots x_n^{d_n}$ is $\sum_{i=1}^n d_i$.*

▶ **Definition 8** (Degree of polynomial space). *We define $L := \dim(\mathcal{V}_{n,d}) = \binom{n+d}{n}$ and $U := \dim(\mathcal{V}_{n,2d}) = \binom{n+2d}{n}$.*

After fixing a basis $(p_1, p_2, \cdots, p_L)$ of $\mathcal{V}_{n,d}$, there exists a one-to-one correspondence between any polynomial $p = \sum_{i=1}^L x_i \cdot p_i \in \mathcal{V}_{n,d}$ and the vector $[x_1, x_2, \cdots, x_L] \in \mathbb{R}^L$. From now on when the basis is clear from context, we will use $\mathcal{V}_{n,d}$ and $\mathbb{R}^L$ interchangeably, and similarly $\mathcal{V}_{n,2d}$ and $\mathbb{R}^U$ interchangeably.

▶ **Definition 9** (SOS polynomials). *A polynomial $p \in \mathcal{V}_{n,2d}$ is said to be a sum-of-squares (SOS) polynomial if p can be written as a sum of squares of polynomials, i.e. $p = \sum_{i=1}^M q_i^2$ for some $M \in \mathbb{N}$ and polynomials $q_1, q_2, \cdots, q_M \in \mathcal{V}_{n,d}$.*

We use $\Sigma_{n,2d}$ to denote the set of n-variate SOS polynomials of degree at most $2d$.

The set $\Sigma_{n,2d}$ is a closed convex and pointed cone in $\mathcal{V}_{n,2d}$ with non-empty interior (Theorem 17.1 of [24]). The SOS optimization problem requires the variable $x \in \mathbb{R}^U$ to be in the SOS cone, and it is a special case of conic programming. Given a constraint matrix $A \in \mathbb{R}^{m\times U}$ where $m \le U$, and $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^U$, the SOS optimization can be written in the following primal-dual formulation:

$$\begin{array}{lll} \text{Primal:} \quad \min \langle c, x\rangle & & \text{Dual:} \quad \max \langle y, b\rangle \\ \quad \text{s.t.} \quad Ax = b & & \text{s.t.} \quad A^\top y + s = c \\ \qquad x \in \Sigma_{n,2d}\,, & & \qquad s \in \Sigma_{n,2d}^*\,. \end{array} \tag{SOS}$$

Here $\Sigma_{n,2d}^* := \{s \in \mathbb{R}^U \mid s^\top x \ge 0,\ \forall x \in \Sigma_{n,2d}\}$ denotes the dual cone of $\Sigma_{n,2d}$.

Nesterov in [24] noted that the dual SOS cone allows the following characterization.

▶ **Theorem 10** (Dual cone characterization, Theorem 17.1 of [24]). *For any ordered bases $\mathbf{p} = (p_1, \ldots, p_L)$ and $\mathbf{q} = (q_1, \ldots, q_U)$ of $\mathcal{V}_{n,d}$ and $\mathcal{V}_{n,2d}$, let $\Lambda : \mathbb{R}^U \to \mathbb{R}^{L\times L}$ be the unique linear mapping satisfying $\Lambda(\mathbf{q}) = \mathbf{p}\mathbf{p}^\top$.*[4] *Then the dual cone $\Sigma_{n,2d}^*$ admits the characterization under the bases $\mathbf{p}$ and $\mathbf{q}$:*

$$\Sigma_{n,2d}^* = \big\{s \in \mathbb{R}^U \mid \Lambda(s) \succeq 0\big\}. \tag{8}$$

As barrier functions for the cone of positive semidefinite matrices are well-known, this also gives rise to a barrier function for the dual SOS cone. With the standard log-det barrier for the semidefinite cone, the following function $F : \Sigma_{n,2d}^* \to \mathbb{R}$ is a barrier function for $\Sigma_{n,2d}^*$:

$$F(s) = -\log\det(\Lambda(s)).$$

Furthermore, the barrier parameter ν_F of $F(s)$ is bounded by the barrier parameter L of the original log-det barrier function ([24]).

[4] This equation means $\forall t \in \mathbb{R}^n$, $\Lambda([q_1(t), \cdots, q_U(t)]^\top) = [p_1(t), \cdots, p_L(t)]^\top \cdot [p_1(t), \cdots, p_L(t)]$.

Interpolant basis

The barrier function depends on the choice of the basis for both $\mathcal{V}_{n,d}$ and $\mathcal{V}_{n,2d}$, as the linear map Λ depends on these two bases. We follow the approach of [27] and focus on the so-called *interpolant* bases, which generalises well to multivariate polynomials and is numerically stable.

For the vector space $\mathcal{V}_{n,2d}$, consider a set of *unisolvent* points $\mathcal{T} = \{t_1, t_2, \cdots, t_U\} \subseteq \mathbb{R}^n$, which is a set points such that every polynomial in $\mathcal{V}_{n,2d}$ is uniquely determined by its values on the points in $\mathcal{T}$. For univariate polynomials any set of U points suffices, but this does not hold anymore for the multivariate case. To also ensure numerical stability, the so called (approximate) Fekete points can be used as unisolvent points [32, 8].

The interpolant basis is defined as follows. Let us fix a set of unisolvent points $\mathcal{T} = \{t_1, t_2, \cdots, t_U\} \subseteq \mathbb{R}^n$. Now every $t_u \in \mathcal{T}$ implies a Lagrange polynomial q_u which is the unique polynomial that satisfies $q_u(t_u) = 1$ and $q_u(t_v) = 0$ for all $t_v \neq t_u \in \mathcal{T}$. The Lagrange polynomials form a basis $\mathbf{q} = (q_1, \cdots, q_U)$ of $\mathcal{V}_{n,2d}$. Choose any basis $\mathbf{p} = (p_1, \ldots, p_L)$ of $\mathcal{V}_{n,d}$. Define the matrix $P \in \mathbb{R}^{U \times L}$ as

$$P_{u,\ell} = p_\ell(t_u), \quad \forall u \in [U], \ell \in [L].$$

By the definition of the Lagrange polynomials, $p_i p_j = \sum_{u=1}^{U} p_i(t_u) p_j(t_u) q_u$, so we have $\mathbf{p}\mathbf{p}^\top = P^\top \operatorname{diag}(\mathbf{q}) P$. Thus under the bases $\mathbf{p}$ and $\mathbf{q}$, the linear map $\Lambda : \mathbb{R}^U \to \mathbb{R}^{L \times L}$ takes on the following convenient form:

$$\Lambda(s) = P^\top \operatorname{diag}(s) P. \tag{9}$$

For more details on how to pick the unisolvent points and how to construct P, we refer the readers to [27] and the references therein.

4 Algorithm

Since in this paper we focus on the theoretical running time of the algorithm, for simplicity we use the barrier method (see e.g. [30, Chapter 2]) instead of the more sophisticated Skajaa–Ye Algorithm used by [27].

The dual formulation of (SOS) is equivalent to the following optimization problem

$$\min -b^\top y \quad s.t. \;\; y \in \overline{D}_F,$$

where with an abuse of the notation we define $F : \mathbb{R}^m \to \mathbb{R}_+$ to be the barrier function

$$F(y) = -\log\det(\Lambda(c - A^\top y)) \tag{10}$$

for $c \in \mathbb{R}^U$, and $A \in \mathbb{R}^{m \times U}$, and $\Lambda(s) = P^\top \operatorname{diag}(s) P$ is the linear operator defined in Eq (9). $D_F \subseteq \mathbb{R}^m$ is the domain of F, and $\overline{D}_F$ is the closure of D_F.

The barrier parameter of the barrier function F is $\nu_F = L$. The gradient and the Hessian of the barrier function F are (define $s := c - A^\top y$):

$$g(y) = A \cdot \operatorname{diag}\left(P\left(P^\top \operatorname{diag}(s) P\right)^{-1} P^\top\right),$$
$$H(y) = A \cdot \left(P\left(P^\top \operatorname{diag}(s) P\right)^{-1} P^\top\right)^{\circ 2} \cdot A^\top.$$

For any $\eta > 0$, define a function $F_\eta : \mathbb{R}^m \to \mathbb{R}$:

$$F_\eta(y) = -\eta \cdot b^\top y + F(y).$$

The gradient and the Hessian of $F_\eta(y)$ are:

$$g_\eta(y) = -\eta \cdot b + A \cdot \operatorname{diag}\left(P\left(P^\top \operatorname{diag}(s) P\right)^{-1} P^\top\right),$$
$$H_\eta(y) = A \cdot \left(P\left(P^\top \operatorname{diag}(s) P\right)^{-1} P^\top\right)^{\circ 2} \cdot A^\top.$$

Note that $H_\eta(y) = H(y)$ for any η.

In each iteration the barrier method increases η by a factor of $1 + \Theta(\frac{1}{\sqrt{L}})$, and it performs a Newton step

$$y \leftarrow y - H_\eta(y)^{-1} \cdot g_\eta(y).$$

By standard IPM theory it suffices to use a spectral approximation of the Hessian matrix in the Newton step. For more details see e.g. [30].

The main technical part of our algorithm is to efficiently maintain a matrix N that is the spectral approximation of the inverse of the Hessian matrix. To do this, we maintain another matrix $\widetilde{S}$ that is a spectral approximation of $S := P^\top \operatorname{diag}(s) P$, and we use the subroutine LowRankUpdate(Algorithm 3, Lemma 14) to update $\widetilde{S}$. After $\widetilde{S}$ is updated, we use another subroutine UpdateHessianInv (Algorithm 2, Lemma 11) to update N. A complete description of our algorithm can be found in Algorithm 1.

5 Updating Hessian inverse efficiently

In this section we prove how to update the Hessian inverse efficiently. We present the algorithm UpdateHessianInv in Algorithm 2.

▶ **Lemma 11** (Hessian inverse update). *In the algorithm UpdateHessianInv (Algorithm 2), the inputs are the maintained matrices T, N and the updates $V_1, V_2 \in \mathbb{R}^{L \times r}$ where r satisfies $Lr \le U$. The inputs satisfy that for some $\widetilde{S} \in \mathbb{S}^{L \times L}$,*

$$T = \widetilde{S}^{-1} \in \mathbb{R}^{L \times L},$$
$$N = \left(A \cdot (P \widetilde{S}^{-1} P^\top)^{\circ 2} \cdot A^\top\right)^{-1} \in \mathbb{R}^{m \times m},$$

Let $\widetilde{S}^{\text{new}} = \widetilde{S} + V_1 V_2^\top$. The algorithm outputs two matrices T^{new}, N^{new} such that

$$T^{\text{new}} = (\widetilde{S}^{\text{new}})^{-1} \in \mathbb{R}^{L \times L},$$
$$N^{\text{new}} = \left(A \cdot (P (\widetilde{S}^{\text{new}})^{-1} P^\top)^{\circ 2} \cdot A^\top\right)^{-1} \in \mathbb{R}^{m \times m}.$$

Furthermore, the algorithm takes $O(\mathcal{T}_{\text{mat}}(U, U, Lr))$ time.

Proof. We first prove the correctness by analyzing each step of the algorithm.

Step 1. Compute $\overline{V_1}, \overline{V_2} \in \mathbb{R}^{L \times r}$ and $T^{\text{new}} \in \mathbb{R}^{L \times L}$.

$$\begin{aligned} T^{\text{new}} &= T + \overline{V_1} \cdot \overline{V_2}^\top \\ &= T - T V_1 \cdot (I + V_2^\top T V_1)^{-1} \cdot V_2^\top T^\top \\ &= (\widetilde{S} + V_1 V_2^\top)^{-1} = (\widetilde{S}^{\text{new}})^{-1}, \end{aligned}$$

where the first two steps follow from algorithm description, the third step follows from the Woodbury identity (Fact 2) and $T = \widetilde{S}^{-1}$.

Thus T^{new} satisfies the requirement of the output.

Algorithm 1 Main SOS algorithm.

Parameters: $\delta \in (0,1)$, $\epsilon_N \in (0, 0.05)$, $\alpha = \frac{\epsilon_N}{20\sqrt{L}}$, $t = 40\epsilon_N^{-1}\sqrt{L}\log(L/\delta)$.
Input : $A \in \mathbb{R}^{m \times U}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^U$
Output : A near feasible and optimal solution.

Construct $P \in \mathbb{R}^{U \times L}$ of the interpolant basis. Convert A, b, c to the interpolant basis.
Use Lemma 22 to obtain a modified dual SOS optimization problem which has an initial solution $(y, s) \in \mathbb{R}^m \times \mathbb{R}^U$ that is optimal for F_η, where $\eta = 1$.
$\widetilde{S} \leftarrow S \leftarrow P^\top \operatorname{diag}(s) P$; // $\widetilde{S}, S \in \mathbb{R}^{L \times L}$
$T \leftarrow S^{-1}$; // $T \in \mathbb{R}^{L \times L}$
$N \leftarrow \left(A(PTP^\top)^{\circ 2} A^\top\right)^{-1}$; // $N \in \mathbb{R}^{m \times m}$
$g \leftarrow -\eta \cdot b + A \cdot \operatorname{diag}\left(P(P^\top \operatorname{diag}(s) P)^{-1} P^\top\right)$; // $g \in \mathbb{R}^m$
for $i = 1, 2, \cdots, t$ **do**
  $\delta_y \leftarrow -N \cdot g$; // $\delta_y \in \mathbb{R}^m$
  $y^{\mathrm{new}} \leftarrow y + \delta_y$; // $y^{\mathrm{new}} \in \mathbb{R}^m$
  $s^{\mathrm{new}} \leftarrow c - A^\top y^{\mathrm{new}}$; // $s^{\mathrm{new}} \in \mathbb{R}^U$
  $\eta^{\mathrm{new}} \leftarrow \eta \cdot (1 + \alpha)$;
  $S^{\mathrm{new}} \leftarrow P^\top \operatorname{diag}(s^{\mathrm{new}}) P$; // $S^{\mathrm{new}} \in \mathbb{R}^{L \times L}$
  $\widetilde{S}^{\mathrm{new}}, V_1, V_2 \leftarrow \textsc{LowRankUpdate}(S^{\mathrm{new}}, \widetilde{S})$;
    // Lemma 14, $\widetilde{S}^{\mathrm{new}} \in \mathbb{R}^{L \times L}$, $V_1, V_2 \in \mathbb{R}^{L \times r_i}$ or $V_1 = V_2 =$ null
  if $V_1 = V_2 =$ null **then**
    $T^{\mathrm{new}} \leftarrow (\widetilde{S}^{\mathrm{new}})^{-1}$; // $T^{\mathrm{new}} \in \mathbb{R}^{L \times L}$
    $N^{\mathrm{new}} \leftarrow \left(A \cdot (PT^{\mathrm{new}} P^\top)^{\circ 2} \cdot A^\top\right)^{-1}$; // $N^{\mathrm{new}} \in \mathbb{R}^{m \times m}$
  else
    $T^{\mathrm{new}}, N^{\mathrm{new}} \leftarrow \textsc{UpdateHessianInv}(T, N, V_1, V_2)$;
      // Lemma 11, $T^{\mathrm{new}} \in \mathbb{R}^{L \times L}, N^{\mathrm{new}} \in \mathbb{R}^{m \times m}$
  $g^{\mathrm{new}} \leftarrow -\eta^{\mathrm{new}} \cdot b + A \cdot \operatorname{diag}\left(P\left(P^\top \operatorname{diag}(s^{\mathrm{new}}) P\right)^{-1} P^\top\right)$; // $g^{\mathrm{new}} \in \mathbb{R}^m$
  $(\eta, y, s, \widetilde{S}, T, N, g) \leftarrow (\eta^{\mathrm{new}}, y^{\mathrm{new}}, s^{\mathrm{new}}, \widetilde{S}^{\mathrm{new}}, T^{\mathrm{new}}, N^{\mathrm{new}}, g^{\mathrm{new}})$;
return (y, s)

Step 2. Compute $Y', Z' \in \mathbb{R}^{U \times (L+r)}$ and $Y, Z \in \mathbb{R}^{U \times (L+r)r}$. We prove that Y and Z satisfy $(PTP^\top)^{\circ 2} + Y \cdot Z^\top = (PT^{\mathrm{new}} P^\top)^{\circ 2}$:

$$\begin{aligned}
(PTP^\top)^{\circ 2} + Y \cdot Z^\top &= (PTP^\top)^{\circ 2} + \sum_{i=1}^r \operatorname{diag}(u_i) \cdot \left(Y' \cdot (Z')^\top\right) \cdot \operatorname{diag}(v_i) \\
&= (PTP^\top)^{\circ 2} + \left(Y' \cdot (Z')^\top\right) \circ \left(\sum_{i=1}^r u_i \cdot v_i^\top\right) \\
&= (PTP^\top)^{\circ 2} + \left(2PTP^\top + (P\overline{V_1}) \cdot (P\overline{V_2})^\top\right) \circ \left((P\overline{V_1}) \cdot (P\overline{V_2})^\top\right) \\
&= \left(PTP^\top + (P\overline{V_1}) \cdot (P\overline{V_2})^\top\right)^{\circ 2} \\
&= (PT^{\mathrm{new}} P^\top)^{\circ 2},
\end{aligned}$$

where the first step follows from the algorithm description of Y and Z, the second step follows from Part 2 of Fact 4 that $\operatorname{diag}(x) \cdot A \cdot \operatorname{diag}(y) = A \circ (x \cdot y^\top)$, the third step follows from $Y' \cdot (Z')^\top = 2PTP^\top + (P\overline{V_1}) \cdot (P\overline{V_2})$ and $(P\overline{V_1}) \cdot (P\overline{V_2}) = \sum_{i=1}^r u_i \cdot v_i^\top$ (see algorithm description of Y' and Z'), the last step follows from $T^{\mathrm{new}} = T + \overline{V_1} \cdot \overline{V_2}^\top$.

Algorithm 2 UpdateHessianInv.

Input : $T \in \mathbb{R}^{L\times L}$, $N \in \mathbb{R}^{m\times m}$, $V_1, V_2 \in \mathbb{R}^{L\times r}$
Output: $T^{\text{new}} \in \mathbb{R}^{L\times L}$, $N^{\text{new}} \in \mathbb{R}^{m\times m}$

1 // Step 1
2 $\overline{V_1} \leftarrow -TV_1 \cdot (I + V_2^\top T V_1)^{-1}$; // $\overline{V_1} \in \mathbb{R}^{L\times r}$
3 $\overline{V_2} \leftarrow TV_2$; // $\overline{V_2} \in \mathbb{R}^{L\times r}$
4 $T^{\text{new}} \leftarrow T + \overline{V_1} \cdot \overline{V_2}^\top$; // $T^{\text{new}} \in \mathbb{R}^{L\times L}$
5 // Step 2
6 $Y' \leftarrow [2PT, P\overline{V_1}]$; // $Y' \in \mathbb{R}^{U\times(L+r)}$
7 $Z' \leftarrow [P, P\overline{V_2}]$; // $Z' \in \mathbb{R}^{U\times(L+r)}$
8 $Y \leftarrow [\text{diag}(u_1)Y', \cdots, \text{diag}(u_r)Y']$, u_i is the i-th column of $P\overline{V_1}$; // $Y \in \mathbb{R}^{U\times(L+r)r}$
9 $Z \leftarrow [\text{diag}(v_1)Z', \cdots, \text{diag}(v_r)Z']$, v_i is the i-th column of $P\overline{V_2}$; // $Z \in \mathbb{R}^{U\times(L+r)r}$
10 // Step 3
11 $N^{\text{new}} \leftarrow N - N\cdot(AY)\cdot\left(I + (AZ)^\top N(AY)\right)^{-1}\cdot(AZ)^\top \cdot N$; // $N^{\text{new}} \in \mathbb{R}^{m\times m}$
12 **return** $T^{\text{new}}, N^{\text{new}}$

Step 3. **Compute $N^{\text{new}} \in \mathbb{R}^{m\times m}$.**

$$\begin{aligned} N^{\text{new}} &= N - N\cdot(AY)\cdot\left(I + (AZ)^\top N(AY)\right)^{-1}\cdot(AZ)^\top\cdot N \\ &= \left(A\cdot(PTP^\top)^{\circ 2}\cdot A^\top + (AY)\cdot(AZ)^\top\right)^{-1} \\ &= \left(A\cdot(PT^{\text{new}}P^\top)^{\circ 2}\cdot A^\top\right)^{-1}, \end{aligned}$$

where the first step follows from the algorithm description of N^{new}, the second step follows from $N = \left(A\cdot(PTP^\top)^{\circ 2}\cdot A^\top\right)^{-1}$ and the Woodbury identity (Fact 2), and the last step follows from $(PT^{\text{new}}P^\top)^{\circ 2} = (PTP^\top)^{\circ 2} + Y\cdot Z^\top$.

Thus N^{new} satisfies the requirement of the output.

Time complexity. It is easy to see that the most time-consuming step is to compute N^{new} on Line 11, and in total this step takes $O(\mathcal{T}_{\text{mat}}(m,U,Lr) + \mathcal{T}_{\text{mat}}(m,m,Lr) + (Lr)^\omega)$ time.

Since $Lr \le U$ and $m \le U$, overall this algorithm takes at most $O(\mathcal{T}_{\text{mat}}(U,U,Lr))$ time. ◀

6 Correctness

6.1 Standard results from IPM theory

We use the following two results of the barrier method that hold for any cone with a barrier function. The proofs are standard, (see e.g., [30, Section 2.4]). For completeness we include a proof in the full version.

▶ **Lemma 12** (Invariance of Newton step, [30]). *Consider the following optimization problem:* $\min -b^\top y$ *s.t.* $y \in \overline{D}_F$, *where* $F : \mathbb{R}^m \to \mathbb{R}_+$ *is a barrier function with barrier parameter* ν_F, $D_F \subseteq \mathbb{R}^m$ *is the domain of* F, *and* $\overline{D}_F$ *is the closure of* D_F. *For any* $\eta \ge 1$, *define* $F_\eta(y) = -\eta b^\top y + F(y)$. *Let* $g_\eta(y) \in \mathbb{R}^m$ *and* $H(y) \in \mathbb{R}^{m\times m}$ *denote the gradient and the Hessian of* F_η *at* y.

Let $0 < \epsilon_N \le 0.05$ *be a parameter. If a feasible solution* $y \in D_F$, *a parameter* $\eta > 0$, *and a positive definite matrix* $\widetilde{H} \in \mathbb{S}^{n\times n}$ *satisfy the following:*

$$\|g_\eta(y)\|_{H(y)^{-1}} \le \epsilon_N, \quad \widetilde{H} \approx_{0.02} H(y).$$

Then $\eta^{\text{new}} = \eta\cdot(1 + \frac{\epsilon_N}{20\sqrt{\nu_F}})$, $y^{\text{new}} = y + \delta_y$ *where* $\delta_y = -\widetilde{H}^{-1} g_{\eta^{\text{new}}}(y)$ *satisfy* $y^{\text{new}} \in D_F$ *and*

$$\|\delta_y\|_{H(y)} \le 2\epsilon_N, \quad \|g_{\eta^{\text{new}}}(y^{\text{new}})\|_{H(y^{\text{new}})^{-1}} \le \epsilon_N.$$

▶ **Lemma 13** (Approximate optimality, [30]). *Consider the following optimization problem:* $\min -b^\top y$ *s.t.* $y \in \overline{D}_F$, *where* $F : \mathbb{R}^m \to \mathbb{R}_+$ *is a barrier function with barrier parameter* ν_F, $D_F \subseteq \mathbb{R}^m$ *is the domain of* F, *and* $\overline{D}_F$ *is the closure of* D_F. *Let* OPT *be the optimal objective value of this optimization problem. For any* $\eta \geq 1$, *define* $F_\eta(y) = -\eta b^\top y + F(y)$. *Let* $g_\eta(y) \in \mathbb{R}^m$ *and* $H(y) \in \mathbb{R}^{m \times m}$ *denote the gradient and the Hessian of* F_η *at* y.

Let $0 < \epsilon_N \leq 0.05$. *If a feasible solution* $y \in D_F$ *satisfies* $\|g_\eta(y)\|_{H(y)^{-1}} \leq \epsilon_N$, *then we have* $-b^\top y \leq \mathrm{OPT} + \frac{\nu_F}{\eta} \cdot (1 + 2\epsilon_N)$.

6.2 Low rank update

We use the following low rank update procedure of [16] and [15], which we modify by using a cutoff when $r \geq U/L$. The proof of the following lemma can be found in [15, Theorem 10.8].

Algorithm 3 LowRankUpdate of [16].

Parameters: A real number $\epsilon_S < 0.01$.
Input : New exact matrix $S^{\mathrm{new}} \in \mathbb{R}^{L \times L}$, old approximate matrix $\widetilde{S} \in \mathbb{R}^{L \times L}$,
Output : New approximate matrix $\widetilde{S}^{\mathrm{new}} \in \mathbb{R}^{L \times L}$, update matrices $V_1, V_2 \in \mathbb{R}^{L \times r}$.

$Z_{\mathrm{mid}} \leftarrow (S^{\mathrm{new}})^{-1/2} \widetilde{S} (S^{\mathrm{new}})^{-1/2} - I$
Compute spectral decomposition $Z_{\mathrm{mid}} = X \operatorname{diag}(\lambda) X^\top$
Let $\pi : [L] \to [L]$ be a sorting permutation such that $|\lambda_{\pi(i)}| \geq |\lambda_{\pi(i+1)}|$.
if $|\lambda_{\pi(1)}| \leq \epsilon_S$ **then**
  $\widetilde{S}_{\mathrm{new}} \leftarrow \widetilde{S}$;
  return $(\widetilde{S}^{\mathrm{new}}, 0, 0)$
else
  $r \leftarrow 1$;
  while $2r \leq U/L$ and $(|\lambda_{\pi(2r)}| > \epsilon_S$ or $|\lambda_{\pi(2r)}| > (1 - 1/\log L)|\lambda_{\pi(r)}|)$ **do**
    $r \leftarrow r + 1$;
  $r \leftarrow 2r$;
  if $r \geq U/L$ **then**
    $\widetilde{S}^{\mathrm{new}} \leftarrow S^{\mathrm{new}}$; `// Here we deviate from [16]`
    return $(\widetilde{S}^{\mathrm{new}}, \texttt{null}, \texttt{null})$
  else
    $\lambda^{\mathrm{new}}_{\pi(i)} \leftarrow \begin{cases} 0 & \text{if } i = 1, 2, \ldots, r \\ \lambda_{\pi(i)} & \text{else} \end{cases}$
    $\Omega \leftarrow \operatorname{supp}(\lambda^{\mathrm{new}} - \lambda)$; // $|\Omega| = r$
    $V_1 \leftarrow ((S^{\mathrm{new}})^{1/2} \cdot X \cdot \operatorname{diag}(\lambda^{\mathrm{new}} - \lambda))_{:,\Omega}$; // $V_1 \in \mathbb{R}^{L \times r}$
    $V_2 \leftarrow ((S^{\mathrm{new}})^{1/2} \cdot X)_{:,\Omega}$; // $V_2 \in \mathbb{R}^{L \times r}$
    $\widetilde{S}^{\mathrm{new}} \leftarrow \widetilde{S} + (S^{\mathrm{new}})^{1/2} X \operatorname{diag}(\lambda^{\mathrm{new}} - \lambda) X^\top (S^{\mathrm{new}})^{1/2}$;
    // $\widetilde{S}^{\mathrm{new}} = \widetilde{S} + V_1 V_2^\top \in \mathbb{R}^{L \times L}$
    return $(\widetilde{S}^{\mathrm{new}}, V_1, V_2)$;

▶ **Lemma 14** (Low rank update). *The algorithm* LowRankUpdate *(Algorithm 3) has the following properties:*

(i) *The output matrix* $\widetilde{S}^{\mathrm{new}} = \widetilde{S} + V_1 V_2^\top$ *is a spectral approximation of the input matrix:* $\widetilde{S}^{\mathrm{new}} \approx_{\epsilon_S} S^{\mathrm{new}}$.

(ii) *Consider a total of t iterations of* LowRankUpdate. *Initially $\widetilde{S}^{(0)} = S^{(0)}$, and we use $(S^{(i)}, \widetilde{S}^{(i-1)})$ and $(\widetilde{S}^{(i)}, V_1^{(i)}, V_2^{(i)})$ to denote the input and the output of the i-th iteration. We define the rank r_i to be the rank of $V_1^{(i)}$ if $V_1^{(i)} \neq \texttt{null}$, and otherwise we define $r_i = U/L$.*
If the input exact matrices $S^{(0)}, S^{(1)}, \cdots, S^{(t)} \in \mathbb{R}^{L\times L}$ satisfy

$$\|(S^{(i-1)})^{-1/2} S^{(i)} (S^{(i-1)})^{-1/2} - I\|_F \leq 0.02, \ \forall i \in [t]. \tag{11}$$

Then for any non-increasing sequence $g \in \mathbb{R}_+^L$, the ranks r_i satisfy

$$\sum_{i=1}^{t} r_i \cdot g_{r_i} \leq O(t \cdot \|g\|_2 \cdot \log L).$$

Furthermore, the algorithm LowRankUpdate *takes $O(L^\omega)$ time.*

6.3 Slowly moving guarantee

In SOS optimization, the matrix $S = P^\top \operatorname{diag}(s) P$ corresponds to the slack matrix of the SDP. The following lemma proves similar to SDP, in SOS the matrix S is changing slowly. Using this lemma we will prove that the requirement Eq. (11) of Lemma 14 is satisfied, which means we can approximate the change to the slack by a low-rank matrix.

▶ **Lemma 15** (Slowly moving guarantee). *Let $c \in \mathbb{R}^U$ and $A \in \mathbb{R}^{m\times U}$ be the input to the optimization problem. Let $P \in \mathbb{R}^{U\times L}$ be the matrix of the interpolant basis.*

For any $y \in \mathbb{R}^m$ and $y^{\text{new}} = y + \delta_y \in \mathbb{R}^m$, let $s = c - A^\top y \in \mathbb{R}^U$ and $S = P^\top \operatorname{diag}(s) P \in \mathbb{R}^{L\times L}$. Similarly define s^{new} and S^{new} from y^{new}. Let $H(y) = A \cdot \big(P\big(P^\top \operatorname{diag}(s) P\big)^{-1} P^\top\big)^{\circ 2} \cdot A^\top \in \mathbb{R}^{m\times m}$. If $s, s^{\text{new}} \in \Sigma_{n,2d}^*$, then S and S^{new} are both PSD, and we have

$$\|S^{-1/2} S^{\text{new}} S^{-1/2} - I\|_F = \|\delta_y\|_{H(y)}.$$

Proof. Note that if $s, s^{\text{new}} \in \Sigma_{n,2d}^*$, then by the dual cone characterization (Theorem 10) S and S^{new} are both PSD.

For convenience we define $M = PS^{-1}P^\top \in \mathbb{R}^{U\times U}$. Note that $H(y) = A \cdot M^{\circ 2} \cdot A^\top$. We also define $\delta_s = s^{\text{new}} - s = -A^\top \delta_y$. $\forall u \in [U]$, we use $p_u \in \mathbb{R}^L$ to denote the u-th row of P.

$$\begin{aligned}
\|S^{-1/2} S^{\text{new}} S^{-1/2} - I\|_F^2 &= \|S^{-1/2}\big(S^{\text{new}} - S\big) S^{-1/2}\|_F^2 \\
&= \|S^{-1/2}\big(P \operatorname{diag}(\delta_s) P^\top\big) S^{-1/2}\|_F^2 \\
&= \operatorname{tr}\big(S^{-1}(P^\top \operatorname{diag}(\delta_s) P) S^{-1}(P^\top \operatorname{diag}(\delta_s) P)\big) \\
&= \operatorname{tr}\big(S^{-1}(\sum_{u\in U} (\delta_s)_u \cdot p_u p_u^\top) S^{-1}(\sum_{v\in U} (\delta_s)_v \cdot p_v p_v^\top)\big) \\
&= \sum_{u,v\in U} (\delta_s)_u (\delta_s)_v \cdot \operatorname{tr}\big(S^{-1} p_u p_u^\top S^{-1} p_v p_v^\top\big) \\
&= \sum_{u,v\in U} (\delta_s)_u (\delta_s)_v \cdot (p_v^\top S^{-1} p_u)^2 \\
&= \sum_{u,v\in U} (\delta_s)_u (\delta_s)_v \cdot M_{uv}^2 \ = \ \|\delta_s\|_{M^{\circ 2}}^2,
\end{aligned} \tag{12}$$

where the third step follows from $\|A\|_F^2 = \operatorname{tr}(A^\top A)$ and the cyclic property of trace, and the sixth step again follows from the cyclic property of trace.

Since $\delta_s = -A^\top \delta_y$, we have

$$\|\delta_s\|_{M^{\circ 2}}^2 = \delta_s^\top M^{\circ 2} \delta_s = \delta_y^\top A M^{\circ 2} A^\top \delta_y = \delta_y^\top H(y) \delta_y = \|\delta_y\|_{H(y)}^2. \tag{13}$$

Combining Eq. (12) and (13) we get the bound in the lemma statement. ◀

6.4 Proof of correctness

Finally we are ready to prove the correctness of Algorithm 1.

▶ **Theorem 16** (Correctness of Algorithm 1). *Consider the following optimization problem with $A \in \mathbb{R}^{m\times U}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^U$:*

$$\begin{aligned} &\textit{Primal:} \quad \min \langle c, x\rangle && \textit{Dual:} \quad \max \langle y, b\rangle \\ &\quad \text{s.t.} \quad Ax = b && \text{s.t.} \quad A^\top y + s = c \\ &\qquad x \in \Sigma_{n,2d}\,, && \qquad s \in \Sigma^*_{n,2d}\,. \end{aligned}$$

Let OPT *denote the optimal objective value of this optimization problem. Assume Slater's condition and that any primal feasible $x \in \Sigma_{n,2d}$ satisfies $\|x\|_1 \le R$.*

Then for any error parameters $\delta \in (0,1)$, $\epsilon_S \le 0.01$, and $\epsilon_N \le 0.05$, Algorithm 1 outputs $x \in \Sigma_{n,2d}$ that satisfies

$$\langle c, x\rangle \le \mathrm{OPT} + \delta \cdot R\|c\|_\infty \quad \textit{and} \quad \|Ax - b\|_1 \le 8\delta L \cdot (LR\|A\|_\infty + \|b\|_1).$$

Proof. We consider the optimization problem $\min -b^\top y$ s.t. $y \in \overline{D}_F$, where $F : \mathbb{R}^m \to \mathbb{R}_+$ is the barrier function defined in Eq. (10), and $\overline{D}_F$ is the closure of the domain of F. The barrier parameter of F is $\nu_F = L$. This optimization problem is equivalent to the dual formulation and its optimal value is $-$OPT. For any η, let $F_\eta(y) = -\eta b^\top y + F(y)$.

In the beginning Algorithm 1 first uses Lemma 22 to convert the optimization problem to another form which has an initial feasible solution y that is close to the optimal solution of F_η with $\eta = 1$. The initial y satisfies $\|g_\eta(y)\|_{H(y)^{-1}} \le \epsilon_N$ by Lemma 22. Initially we also have $\widetilde{S} = S = P^\top \operatorname{diag}(s) P$ (Line 3 in Algorithm 1).

Next we prove the correctness of Algorithm 1 inductively. At each iteration, we assume the following induction hypothesis is satisfied: (1) $\|g_\eta(y)\|_{H(y)^{-1}} \le \epsilon_N$, (2) $\widetilde{S} \approx_{\epsilon_S} S$. We aim to prove that the updated y^{new}, η^{new}, S^{new}, and $\widetilde{S}^{\mathrm{new}}$ still satisfy these two conditions.

In Lemma 11 we have proved that in Algorithm 1 we always maintain $N = \big(A \cdot (P\widetilde{S}^{-1}P^\top)^{\circ 2} \cdot A^\top\big)^{-1}$. Let $\widetilde{H} = N^{-1}$, we have

$$\widetilde{H} = A \cdot \big(P\widetilde{S}^{-1}P^\top\big)^{\circ 2} \cdot A^\top \approx_{2\epsilon_S} A \cdot \big(PS^{-1}P^\top\big)^{\circ 2} \cdot A^\top = H(y),$$

where in the second step we use the induction hypothesis that $\widetilde{S} \approx_{\epsilon_S} S$, and by Fact 6 we have $\widetilde{S}^{-1} \approx_{\epsilon_S} S^{-1}$, and hence $P\widetilde{S}^{-1}P^\top \approx_{\epsilon_S} PS^{-1}P^\top$, and hence $(P\widetilde{S}^{-1}P^\top)^{\circ 2} \approx_{2\epsilon_S} (PS^{-1}P^\top)^{\circ 2}$.

The new vector y^{new} is computed as $y^{\mathrm{new}} = y + \delta_y$ where $\delta_y = -\widetilde{H}^{-1} g_\eta(y)$ (Line 8 and 9 of Algorithm 1). And η is updated to $\eta^{\mathrm{new}} = \eta \cdot (1 + \frac{\epsilon_N}{20\sqrt{L}})$ (Line 11 of Algorithm 1). Since $\|g_\eta(y)\|_{H(y)^{-1}} \le \epsilon_N$, and $\widetilde{H} \approx_{2\epsilon_S} H(y)$ where $2\epsilon_S \le 0.02$ by its definition in Algorithm 3, the requirements of Lemma 12 are satisfied, so we have

$$\|g_{\eta^{\mathrm{new}}}(y^{\mathrm{new}})\|_{H(y^{\mathrm{new}})^{-1}} \le \epsilon_N, \quad \text{and} \quad \|\delta_y\|_{H(y)} \le 2\epsilon_N.$$

This proves the first induction hypothesis.

Then using Lemma 15 and since $\epsilon_N \le 0.01$ by its definition in Algorithm 1, we have

$$\|S^{-1/2}(S^{\mathrm{new}})S^{-1/2} - I\|_F \le \|\delta_y\|_{H(y)} \le 2\epsilon_N \le 0.02.$$

Thus the input matrix S^{new} to LowRankUpdate satisfies the requirement of Eq. (11) of Lemma 14, and we have that $\widetilde{S}^{\mathrm{new}} \approx_{\epsilon_S} S^{\mathrm{new}}$. This proves the second induction hypothesis.

Finally, we know that after $t = 40\epsilon_N^{-1}\sqrt{L}\log(L/\delta)$ iterations, η becomes $(1+\frac{\epsilon_N}{20\sqrt{L}})^t \geq 2L/\delta^2$, so using Lemma 13, we have

$$b^\top y \geq \mathrm{OPT} - \frac{\nu_F}{\eta}\cdot(1+2\epsilon_N) \geq \mathrm{OPT} - \delta^2.$$

Thus the initialization lemma (Lemma 22) ensures that we have a solution $x \in \Sigma_{n,2d}$ to the original primal optimization problem which satisfies

$$\langle c, x\rangle \leq \mathrm{OPT} + \delta\cdot R\|c\|_\infty, \qquad \|Ax-b\|_1 \leq 8\delta L\cdot(LR\|A\|_\infty + \|b\|_1). \qquad \blacktriangleleft$$

7 Time complexity

7.1 Worst case time

We first bound the worst case running time of Algorithm 1. The running time of the i-th iteration depends on the updated rank r_i of LowRankUpdate, which is defined to be the size of $V_1^{(i)}$ if $V_1^{(i)} \neq \texttt{null}$, and U/L otherwise (see Lemma 14).

▶ **Lemma 17** (Worst case time of Algorithm 1). *In Algorithm 1, the initialization time is $O(U^\omega)$, and the running time in the i-th iteration is $O(\mathcal{T}_{\mathrm{mat}}(U,U,\min\{Lr_i,U\}))$.*

Proof.

Initialization time. The most time-consuming step of initialization is Line 5, where computing $N = \left(A(PTP^\top)^{\circ 2}A^\top\right)^{-1}$ takes $O(\mathcal{T}_{\mathrm{mat}}(U,U,L) + \mathcal{T}_{\mathrm{mat}}(U,U,m))$ time. This is bounded by $O(U^\omega)$ since $L, m \leq U$.

Time per iteration. In each iteration the most time-consuming steps are (1) computing S^{new} and calling LowRankUpdate on Line 12-14, (2) executing the if-clause on Line 15-20, and (3) computing g^{new} on Line 21.

1. Computing S^{new} on Line 12 takes $\mathcal{T}_{\mathrm{mat}}(U,L,L)$ time. Calling LowRankUpdate on Line 14 takes $O(L^\omega)$ time by Lemma 14.
2. In the if-clause on Line 15-20, if $V_1 = V_2 = \texttt{null}$, then $Lr_i \geq U$, and we compute $N^{\mathrm{new}} = \left(A\cdot(PT^{\mathrm{new}}P^\top)^{\circ 2}\cdot A^\top\right)^{-1}$, which takes $O(\mathcal{T}_{\mathrm{mat}}(U,U,U))$ time. Otherwise we call UpdateHessianInv on Line 19, which takes $O(\mathcal{T}_{\mathrm{mat}}(U,U,Lr_i))$ time by Lemma 11. In total the if-clause has running time $O(\mathcal{T}_{\mathrm{mat}}(U,U,\min\{Lr_i,U\}))$.
3. Computing the gradient $g = -\eta^{\mathrm{new}}\cdot b + A\cdot\mathrm{diag}\left(P\left(P^\top\mathrm{diag}(s^{\mathrm{new}})P\right)^{-1}P^\top\right)$ on Line 21 takes $O(\mathcal{T}_{\mathrm{mat}}(U,U,L))$ time since $m \leq U$.

Thus the total time per iteration is $O(\mathcal{T}_{\mathrm{mat}}(U,U,\min\{Lr_i,U\}))$. $\blacktriangleleft$

7.2 Amortized time

In this section we bound the amortized running time of Algorithm 1.

Let ω be the matrix multiplication exponent, let α be the dual matrix multiplication exponent. The current best values are $\omega \approx 2.373$ and $\alpha \approx 0.314$ [21, 10, 1]. Note that the current best values of ω and α satisfies that $\alpha \geq 5 - 2\omega$. We use the following modified lemma from [15]:

▶ **Lemma 18** (Helpful lemma for amortization, modified version of Lemma 10.13 of [15]). *Let t denote the total number of iterations. Let $r_i \in [L]$ be the rank for the i-th iteration for $i \in [t]$. Assume r_i satisfies the following condition: for any vector $g \in \mathbb{R}_+^L$ which is non-increasing, we have $\sum_{i=1}^t r_i \cdot g_{r_i} \leq O(t \cdot \|g\|_2)$.*

If the cost in the i-th iteration is $O(\mathcal{T}_{\mathrm{mat}}(U, U, \min\{Lr_i, U\}))$, when $\alpha \geq 5 - 2\omega$, the amortized cost per iteration is $U^{2+o(1)} + U^{\omega - 1/2 + o(1)} \cdot L^{1/2}$.

We include the proof of Lemma 18 for completeness. The main difference between this proof and that of [15] is that we cut off at U/L instead of L. Our proof makes use of the following two facts about ω and α (Lemma A.4 and Lemma A.5 of [9]).

▶ **Fact 19** (Relation of ω and α). $\omega \leq 3 - \alpha$.

▶ **Fact 20** (Upper bound of $\mathcal{T}_{\mathrm{mat}}(n, n, r)$). *For any $r \leq n$, we have that $\mathcal{T}_{\mathrm{mat}}(n, n, r) \leq n^{2+o(1)} + r^{\frac{\omega-2}{1-\alpha}} \cdot n^{2 - \frac{\alpha(\omega-2)}{(1-\alpha)} + o(1)}$.*

Proof of Lemma 18. For r_i that satisfies $r_i \leq U/L$, we have

$$\begin{aligned} \mathcal{T}_{\mathrm{mat}}(U, U, Lr_i) &\leq U^{2+o(1)} + (Lr_i)^{\frac{\omega-2}{1-\alpha}} \cdot U^{2 - \frac{\alpha(\omega-2)}{1-\alpha} + o(1)} \\ &= U^{2+o(1)} + U^{2 - \frac{\alpha(\omega-2)}{1-\alpha} + o(1)} \cdot L^{\frac{\omega-2}{1-\alpha}} \cdot r_i^{\frac{\omega-2}{1-\alpha}}, \end{aligned} \tag{14}$$

where the first step follows from Fact 20.

Define a sequence $g \in \mathbb{R}_+^L$ such that for $r \in [L]$,

$$g_r = \begin{cases} r^{\frac{\omega-2}{1-\alpha} - 1} & \text{if } r \leq U/L, \\ (U/L)^{\frac{\omega-2}{1-\alpha}} \cdot r^{-1} & \text{if } r > U/L. \end{cases}$$

Note that g is non-increasing because $\frac{\omega-2}{1-\alpha} \leq 1$ (Fact 19). Then using the condition in the lemma statement, we have

$$\begin{aligned} \sum_{i=1}^t \min\{r_i^{\frac{\omega-2}{1-\alpha}}, (U/L)^{\frac{\omega-2}{1-\alpha}}\} &= \sum_{i=1}^t r_i \cdot g_{r_i} \\ &\leq t \cdot \|g\|_2 \\ &\leq t \cdot \Big(\int_{x=1}^{U/L} x^{\frac{2(\omega-2)}{1-\alpha} - 2} \mathrm{d}x + (U/L)^{\frac{2(\omega-2)}{1-\alpha}} \cdot \int_{x=U/L}^{L} x^{-2} \mathrm{d}x \Big)^{1/2} \\ &\leq t \cdot \Big(c \cdot (U/L)^{\frac{2(\omega-2)}{1-\alpha} - 1} + (U/L)^{2(\frac{\omega-2}{1-\alpha})} \cdot (U/L)^{-1} \Big)^{1/2} \\ &= t \cdot O((U/L)^{\frac{(\omega-2)}{1-\alpha} - 1/2}), \end{aligned} \tag{15}$$

where the first step follows from the definition of $g \in \mathbb{R}^L$, the second step follows from the assumption $\sum_{t=1}^t r_i \cdot g_{r_i} \leq t \cdot \|g\|_2$ in the lemma statement, the third step follows from upper bounding the ℓ_2 norm $\|g\|_2^2 = \sum_{r=1}^L g_r^2$, and the fourth step follows $\frac{2(\omega-2)}{1-\alpha} \geq 1$ when $\alpha \geq 5 - 2\omega$, so the integral $\int_{x=1}^{U/L} x^{\frac{2(\omega-2)}{1-\alpha} - 2} \mathrm{d}x = c \cdot x^{\frac{2(\omega-2)}{1-\alpha} - 1} \big|_1^{U/L} = O\big((U/L)^{\frac{2(\omega-2)}{1-\alpha} - 1}\big)$ where $c := 1/(\frac{2(\omega-2)}{1-\alpha} - 1)$.

Thus we have

$$\begin{aligned}
&\sum_{t=1}^{t} \mathcal{T}_{\mathrm{mat}}(U,U,\min\{Lr_i,U\}) \\
&\le \sum_{t=1}^{t} \left(U^{2+o(1)} + U^{2-\frac{\alpha(\omega-2)}{1-\alpha}+o(1)} \cdot L^{\frac{\omega-2}{1-\alpha}} \cdot \min\{r_i^{\frac{\omega-2}{1-\alpha}}, (U/L)^{\frac{\omega-2}{1-\alpha}}\}\right) \\
&= t \cdot U^{2+o(1)} + U^{2-\frac{\alpha(\omega-2)}{1-\alpha}+o(1)} \cdot L^{\frac{\omega-2}{1-\alpha}} \cdot \sum_{t=1}^{t} \min\{r_i^{\frac{\omega-2}{1-\alpha}}, (U/L)^{\frac{\omega-2}{1-\alpha}}\} \\
&\le t \cdot U^{2+o(1)} + U^{2-\frac{\alpha(\omega-2)}{1-\alpha}+o(1)} \cdot L^{\frac{\omega-2}{1-\alpha}} \cdot t \cdot (U/L)^{\frac{(\omega-2)}{1-\alpha}-1/2} \\
&= t \cdot (U^{2+o(1)} + U^{\omega-1/2+o(1)} \cdot L^{1/2}),
\end{aligned}$$

where the first step follows from Eq. (14) and $\mathcal{T}_{\mathrm{mat}}(U,U,U) = U^{\omega} = U^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot L^{\frac{\omega-2}{1-\alpha}} \cdot (U/L)^{\frac{\omega-2}{1-\alpha}}$, the second step follows from moving summation inside, the third step follows from Eq. (15), and the last step follows from adding the terms together. ◀

Now we are ready to prove our main theorem for the amortized time of Algorithm 1.

▶ **Theorem 21** (Time of Algorithm 1). *When $\alpha \ge 5 - 2\omega$, the running time of Algorithm 1 is* $(U^2 \cdot L^{1/2} + U^{\omega-1/2} \cdot L) \cdot (\log(1/\delta) + U^{o(1)})$.

Proof. Using Lemma 17 the initialization time is $O(U^{\omega}) \le O(U^{\omega-1/2} \cdot L)$ since $U \le L^2$.

Using Lemma 14 we know that the ranks r_i indeed satisfy the requirement of Lemma 18, and since the worst case time per iteration is $O(\mathcal{T}_{\mathrm{mat}}(U,U,\min\{Lr_i,U\}))$ (Lemma 17), using Lemma 18 the time per iteration is $U^{2+o(1)} + U^{\omega-1/2+o(1)} \cdot L^{1/2}$. Since there are in total $t = 40\epsilon_N^{-1}\sqrt{L}\log(L/\delta)$ iterations, we get the total running time as claimed. ◀

7.3 Comparison with previous results

In this section we compare the running time of [27], [16, 15], and our result. We assume that $m = \Theta(U)$ when making the comparisons.

Ignoring $\log(1/\delta)$ and $U^{o(1)}$ factors, and since $L \le U \le L^2$, the running times are

$$\begin{aligned}
\text{[27] (SOS)} &: L^{0.5} \cdot U^{\omega}, \\
\text{[16, 15] (SDP)}^5 &: L^{0.5} \cdot \min\{UL^2 + U^{\omega},\ L^4 + L^{2\omega-0.5}\}, \\
\text{Ours (SOS)} &: L^{0.5} \cdot (U^2 + U^{\omega-0.5} \cdot L^{0.5}).
\end{aligned}$$

Current ω and α

Plugging in the current best values $\omega \approx 2.373$ and $\alpha \approx 0.314$, we have

$$\begin{aligned}
\text{[27] (SOS)} &: L^{0.5} \cdot U^{2.373}, \\
\text{[16, 15] (SDP)} &: L^{0.5} \cdot \min\{UL^2 + U^{2.373},\ L^{4.246}\} \\
&= L^{0.5} \cdot \begin{cases} UL^2 & \text{when } U \in (L, L^{1.457}], \\ U^{2.373} & \text{when } U \in (L^{1.457}, L^{1.789}], \\ L^{4.246} & \text{when } U \in (L^{1.789}, L^2), \end{cases} \\
\text{Ours (SOS)} &: L^{0.5} \cdot (U^2 + U^{1.873}L^{0.5}).
\end{aligned}$$

[5] When solving SOS, [16] has running time $O(L^{0.5} \cdot (UL^2 + U^{\omega} + L^{\omega})) \le O(L^{0.5} \cdot (UL^2 + U^{\omega}))$, and [15] has running time $O(L^{0.5} \cdot (U^2 + L^4) + U^{\omega} + L^{2\omega}) \le O(L^{4.5} + L^{2\omega})$ since $L \le U \le L^2$.

Note that our running time is always better than the previous results, and for several values of L and U we improve by a polynomial factor. See Figure 1 for an illustration.

8 Initialization

There exist standard techniques to transform a convex program to a form that has an easily obtainable strictly feasible point, see e.g. [39]. We follow the initialization procedure presented by [9] and [16] and adapt to SOS optimization. Similar initialization lemma exists for WSOS optimization. The proof of this lemma can be found in our full version.

Let the matrix $P \in \mathbb{R}^{U \times L}$ and the operator $\Lambda : \mathbb{R}^U \to \mathbb{R}^{L \times L}$ that $\Lambda(s) = P^\top \operatorname{diag}(s) P$ be defined as in the interpolant basis paragraph of Section 3.

▶ **Lemma 22** (Initialization). *Given an instance of* (SOS) *that fulfills Slater's condition, and let R be an upper bound on the ℓ_1-norm of the primal feasible solutions, i.e. all primal feasible x of* (SOS) *fulfill $\|x\|_1 \leq R$, and let $\delta \in (0,1)$. We define $\overline{A} \in \mathbb{R}^{(m+1)\times(U+2)}$, $\overline{b} \in \mathbb{R}^{m+1}$, and $\overline{c} \in \mathbb{R}^{U+2}$ as*

$$\overline{A} = \begin{bmatrix} A & 0 & \frac{1}{R}b - A\overline{g}^0 \\ 1_U^\top & 1 & 0 \end{bmatrix}, \quad \overline{b} = \begin{bmatrix} \frac{1}{R}b \\ 1 + \langle 1_U, \overline{g}^0 \rangle \end{bmatrix}, \quad \overline{c} = \begin{bmatrix} \frac{\delta}{\|c\|_\infty} c \\ 0 \\ 1 \end{bmatrix},$$

and let

$$\overline{x}^0 = \begin{bmatrix} \overline{g}^0 \\ 1 \\ 1 \end{bmatrix} \in \mathbb{R}^{U+2}, \quad \overline{y}^0 = \begin{bmatrix} 0_m \\ -1 \end{bmatrix} \in \mathbb{R}^{m+1}, \text{ and } \overline{s}^0 = \begin{bmatrix} 1_U + \frac{\delta}{\|c\|_\infty} c \\ 1 \\ 1 \end{bmatrix} \in \mathbb{R}^{U+2},$$

where $\overline{g}^0 = g_{\Sigma^*}(\overline{s}^0_{[:U]}) \in \mathbb{R}^U$ for the gradient function $g_{\Sigma^*}(s) := \operatorname{diag}(P(P^\top \operatorname{diag}(s) P)^{-1} P^\top)$ that maps from $\mathbb{R}^U$ to $\mathbb{R}^U$. This defines the auxiliary primal-dual system

$$\begin{aligned} \min \langle \overline{c}, \overline{x} \rangle \qquad & \qquad \max \langle \overline{y}, \overline{b} \rangle \\ \overline{A}\overline{x} = \overline{b} \qquad & \qquad \overline{A}^\top \overline{y} + \overline{s} = \overline{c} \\ \overline{x} \in \Sigma_{n,2d} \times \mathbb{R}^2_{\geq 0}\,, \qquad & \qquad \overline{s} \in \Sigma^*_{n,2d} \times \mathbb{R}^2_{\geq 0}\,. \end{aligned} \tag{Aux-SOS}$$

Then $(\overline{x}^0, \overline{y}^0, \overline{s}^0)$ are feasible to the auxiliary system (Aux-SOS).

Further, under the canonical barrier (we use $\overline{a}_i$ to denote the i-th column of $\overline{A}$):

$$\overline{F}_\eta(\overline{y}) = -\eta\langle \overline{y}, \overline{b} \rangle - \log\det\Big(\Lambda\big((\overline{c} - \overline{A}^\top \overline{y})_{[:U]}\big)\Big) - \log(\overline{c}_{U+1} - \langle \overline{a}_{U+1}, \overline{y} \rangle) - \log(\overline{c}_{U+2} - \langle \overline{a}_{U+2}, \overline{y} \rangle),$$

we have that $\|\overline{g}_{\eta^0}(\overline{y}^0)\|_{\overline{H}(\overline{y}^0)^{-1}} = 0$ for $\eta^0 = 1$.

Further, for any solution $(\overline{x}, \overline{y}, \overline{s})$ to (Aux-SOS) *with duality gap $\leq \delta^2$, its restriction $\widehat{x} := \overline{x}_{[:U]}$ fulfills*

$$\begin{aligned} \langle c, \widehat{x} \rangle &\leq \min_{Ax = b, x \in \Sigma_{n,2d}} \langle c, x \rangle + \delta \cdot R\|c\|_\infty, \\ \|A\widehat{x} - b\|_1 &\leq 8\delta L \cdot (LR\|A\|_\infty + \|b\|_1), \\ \widehat{x} &\in \Sigma_{n,2d}. \end{aligned}$$

References

1 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021.

2 Christine Bachoc and Frank Vallentin. New upper bounds for kissing numbers from semidefinite programming. Technical report, Journal of the American Mathematical Society, 2006.

3 Brandon Ballinger, Grigoriy Blekherman, Henry Cohn, Noah Giansiracusa, Elizabeth Kelly, and Achill Schürmann. Experimental study of energy-minimizing point configurations on spheres. *Experimental Mathematics*, 18(3):257–283, 2009. doi:10.1080/10586458.2009.10129052.

4 Boaz Barak, Samuel B. Hopkins, Jonathan A. Kelner, Pravesh K. Kothari, Ankur Moitra, and Aaron Potechin. A nearly tight sum-of-squares lower bound for the planted clique problem. *SIAM J. Comput.*, 48(2):687–735, 2019. doi:10.1137/17M1138236.

5 Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. In *2011 IEEE 52nd annual symposium on foundations of computer science (FOCS)*, pages 472–481. IEEE, 2011.

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

7 Grigoriy Blekherman, Pablo A Parrilo, and Rekha R Thomas. *Semidefinite optimization and convex algebraic geometry*. SIAM, 2012.

8 L. Bos, S. De Marchi, A. Sommariva, and M. Vianello. Computing multivariate fekete and leja points by numerical linear algebra. *SIAM Journal on Numerical Analysis*, 48(5):1984–1999, 2010. doi:10.1137/090779024.

9 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.

10 François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the 2018 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1029–1046. SIAM, 2018.

11 Bissan Ghaddar, Jakub Marecek, and M. Mevissen. Optimal power flow as a polynomial optimization problem. *IEEE Transactions on Power Systems*, 31:539–546, 2016.

12 Roxana Heß, Didier Henrion, Jean-Bernard Lasserre, and Tien Son Pham. Semidefinite approximations of the polynomial abscissa. *SIAM J. Control. Optim.*, 54(3):1633–1656, 2016. doi:10.1137/15M1033198.

13 Samuel B. Hopkins, Pravesh K. Kothari, Aaron Potechin, Prasad Raghavendra, Tselil Schramm, and David Steurer. The power of sum-of-squares for detecting hidden structures. In *58th IEEE Annual Symposium on Foundations of Computer Science, (FOCS)*, pages 720–731. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.72.

14 Samuel B. Hopkins and Jerry Li. Mixture models, robustness, and sum of squares proofs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, (STOC)*, pages 1021–1034. ACM, 2018. doi:10.1145/3188745.3188748.

15 Baihe Huang, Shunhua Jiang, Zhao Song, Runzhou Tao, and Ruizhe Zhang. Solving sdp faster: A robust ipm framework and efficient implementation, 2021. arXiv:2101.08208.

16 Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. A faster interior point method for semidefinite programming. In *2020 IEEE 61st annual symposium on foundations of computer science (FOCS)*, pages 910–918. IEEE, 2020.

17 Shunhua Jiang, Yunze Man, Zhao Song, Zheng Yu, and Danyang Zhuo. Fast graph neural tangent kernel via kronecker sketching. *arXiv preprint AAAI'22*, 2021. arXiv:2112.02446.

18 Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th annual ACM symposium on Theory of computing (STOC)*, pages 302–311, 1984.

19 Jean Bernard Lasserre. *An Introduction to Polynomial and Semi-Algebraic Optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2015. doi:10.1017/CBO9781107447226.

20 M. Laurent. *Sums of squares, moment matrices and optimization over polynomials*, pages 155–270. Number 149 in The IMA Volumes in Mathematics and its Applications Series. Springer Verlag, Germany, 2009.

21 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation*, pages 296–303, 2014.

22 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 424–433. IEEE, 2014.

23 Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *Conference on Learning Theory (COLT)*, pages 2140–2157. PMLR, 2019.

24 Yurii Nesterov. Squared functional systems and optimization problems. In *High performance optimization*, pages 405–440. Springer, 2000.

25 Yurii Nesterov and Arkadi Nemirovski. Interior-point polynomial algorithms in convex programming. In *Siam Studies in Applied Mathematics*, 1987. doi:10.1137/1.9781611970791.

26 Dávid Papp. Optimal designs for rational function regression. *Journal of the American Statistical Association*, 107(497):400–411, 2012. doi:10.1080/01621459.2012.656035.

27 Dávid Papp and Sercan Yildiz. Sum-of-squares optimization without semidefinite programming. *SIAM Journal on Optimization*, 29(1):822–851, 2019.

28 Pablo Parrilo. *Sum of squares : theory and applications : AMS short course, sum of squares : theory and applications, January 14-15, 2019, Baltimore, Maryland*. American Mathematical Society, Providence, Rhode Island, 2020.

29 Mihai Putinar and Florian-Horia Vasilescu. Positive polynomials on semi-algebraic sets. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 328(7):585–589, 1999. doi:10.1016/S0764-4442(99)80251-1.

30 James Renegar. *A Mathematical View of Interior-Point Methods in Convex Optimization*. Society for Industrial and Applied Mathematics, January 2001. doi:10.1137/1.9780898718812.

31 Tae Roh, Bogdan Dumitrescu, and Lieven Vandenberghe. Multidimensional FIR filter design via trigonometric sum-of-squares optimization. *J. Sel. Topics Signal Processing*, 1(4):641–650, 2007. doi:10.1109/JSTSP.2007.910261.

32 Alvise Sommariva and Marco Vianello. Computing approximate fekete points by qr factorizations of vandermonde matrices. *Computers & Mathematics with Applications*, 57(8):1324–1336, 2009.

33 Zhao Song, Shuo Yang, and Ruizhe Zhang. Does preprocessing help training over-parameterized neural networks? *Advances in Neural Information Processing Systems*, 34, 2021.

34 Zhao Song, Lichen Zhang, and Ruizhe Zhang. Training multi-layer over-parametrized neural network in subquadratic time. *arXiv preprint*, 2021. arXiv:2112.07628.

35 Gilbert Strang. Karmarkar's algorithm and its place in applied mathematics. *The Mathematical Intelligencer*, 9(2):4–10, 1987.

36 Ning Tan. *On the Power of Lasserre SDP Hierarchy*. PhD thesis, EECS Department, University of California, Berkeley, December 2015. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-236.html.

37 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.

38 Jan van den Brand, Binghui Peng, Zhao Song, and Omri Weinstein. Training (overparametrized) neural networks in near-linear time. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185, pages 63:1–63:15, 2021. doi:10.4230/LIPIcs.ITCS.2021.63.

39 Yinyu Ye, Michael J Todd, and Shinji Mizuno. An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of operations research*, 19(1):53–67, 1994.