Solving Min-Min Optimization Problems via Iterative Polynomial Approximation

Saya Egashira
Advisor: Professor Jiawang Nie

Abstract

Nested min-min optimization problems are challenging to solve, especially when the value function of the inner problem lacks a closed form or is computationally expensive to evaluate. This thesis develops and analyzes a framework to address this challenge by approximating the value function with a tractable polynomial. Our method first constructs an initial approximation from sampled data using a least-squares fit. It then iteratively refines this polynomial by strategically sampling new points at the minimum of the current approximation and efficiently updating the coefficients using the Sherman-Morrison formula. We demonstrate through numerical experiments that this iterative process effectively improves the accuracy of the solution.

1 Introduction

We consider the following min-min optimization problem:

$$\min_{z\in K_z}\left\{P(z) := \min_{x\in K_x} f(x,z)\right\} \tag{1.1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$ is a jointly continuous function, and $K_x \subseteq \mathbb{R}^n$, $K_z \subseteq \mathbb{R}^k$ are compact sets.

Solving the optimization problem (1.1) is challenging due to its nested structure. The value function $P(z)$ lacks an analytic form and is typically nonconvex and non-differentiable, even when $f(x, z)$ is well-behaved. Furthermore, each evaluation of $P(z)$ requires solving an inner optimization problem, creating a significant computational bottleneck. Prior algorithmic approaches have focused on settings with more structure. For instance, gradient-based methods have been proposed for cases where strong convexity is assumed [5], alongside mixed-oracle approaches [1]. However, these methods often rely on restrictive assumptions and their computational cost can be substantial, limiting their applicability to large-scale or complex problems.

To overcome these challenges, we propose an optimization framework that replaces the intractable value function $P(z)$ with a tractable polynomial approximation, $\tilde{P}(z)$. Our method proceeds in three main stages. First, we generate an initial set of N samples $\{(z_i, P(z_i))\}_{i=1}^N$ by solving the inner problem at randomly selected points $z_i \in K_z$. We then construct an

initial polynomial $\tilde{P}(z)$ by solving a least-squares regression problem. Second, we find the global minimum of the polynomial approximation $\tilde{P}(z)$ using powerful tools from computational algebraic geometry, such as moment-SOS relaxations [3]. Third, we strategically select new points at which to sample $P(z)$ and efficiently update the coefficients of $\tilde{P}(z)$ using the Sherman-Morrison formula, thereby avoiding the need to resolve the least-squares problem from scratch.

The theoretical validity of our approach rests on two cornerstone theorems from analysis. Berge's Maximum Theorem ensures that $P(z)$ is continuous under our assumptions [11]. Subsequently, the Stone-Weierstrass Theorem guarantees that this continuous function can be uniformly approximated by a polynomial on the compact set K_z, providing a solid foundation for our framework.

This approximation-based framework offers several advantages over traditional methods. By decoupling the inner and outer optimization loops, our method addresses the computational bottleneck of sequential nested evaluations through targeted parallelization. That is, the evaluation of inner problems (3.2) can be distributed across parallel workers, while the polynomial approximation $\tilde{P}(z)$ maintains a sequential refinement process. This hybrid structure is particularly effective when inner optimizations dominate computational cost.

The use of polynomials provides not only an interpretable approximation model but also computational flexibility. Unlike strictly sequential nested-loop algorithms, our approach enables the concurrent solution of new inner problems alongside model updates using existing data. Modern computational architectures (e.g., multi-core CPUs or distributed clusters) can thus be leveraged efficiently, though the ultimate scalability is constrained by the sequential refinement component.

This paper is organized as follows. Section 2 reviews preliminary concepts, including notation and the fundamentals of polynomial optimization and moment relaxations. Section 3 details the proposed method for constructing an initial polynomial approximation of the value function. Section 4 presents our main contribution, an iterative refinement framework. Section 5 provides a series of numerical experiments to demonstrate the practical performance and behavior of our algorithm. Finally, Section 6 concludes the paper with a summary of our findings and a discussion of future research directions.

2 Preliminaries

Notation

The symbol $\mathbb{N}$ (resp., $\mathbb{R}$) represents the set of nonnegative integers (resp., real numbers). For an integer $m > 0$, denote $[m] := \{1, 2, \ldots, m\}$. For a scalar $t \in \mathbb{R}$, $\lceil t \rceil$ denotes the smallest integer greater than or equal to t. For a polynomial p, $\deg(p)$ denotes its total degree, and $\mathbf{vec}(p)$ denotes its coefficient vector. For two vectors a and b, the notation $a \perp b$ means they are perpendicular. The superscript ^{T} denotes the transpose of a matrix or vector. For a symmetric matrix X, $X \succeq 0$ (resp. $X \succ 0$) means that X is positive semidefinite (resp. positive definite). The symbol S^n_+ stands for the set of all n by n real symmetric positive semidefinite matrices. For two symmetric matrices X and Y, $X \succeq Y$ (resp. $X \succ Y$) means that $X - Y \succeq 0$ (resp. $X - Y \succ 0$). For $x := (x_1, \ldots, x_n)$ and a power

vector $\alpha \coloneqq (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, let $|\alpha| \coloneqq \alpha_1 + \cdots + \alpha_n$ and the monomial $x^\alpha \coloneqq x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. For a real number $q \geq 1$, the q-norm of x is denoted as $\|x\|_q \coloneqq (|x_1|^q + \cdots + |x_n|^q)^{1/q}$. The notation $\mathbb{N}_d^n \coloneqq \{\alpha \in \mathbb{N}^n : |\alpha| \leq d\}$ denotes the set of monomial powers with degree at most d. The symbol $\mathbb{R}^{\mathbb{N}_d^n}$ denotes the space of all real vectors labeled by $\alpha \in \mathbb{N}_d^n$. The column vector of all monomials in x of degree up to d is denoted as

$$[x]_d \coloneqq [1 \quad x \quad \cdots \quad x_n \quad x_1^2 \quad x_1 x_2 \quad \cdots \quad x_n^d]^T.$$

The notation $\mathbb{R}[x] \coloneqq \mathbb{R}[x_1, \ldots, x_n]$ stands for the ring of polynomials in x with real coefficients. Let $\mathbb{R}[x]_d$ be the set of real polynomials with degree at most d. $\mathscr{P}(K)$ denotes the cone of polynomials that are nonnegative on K and let

$$\mathscr{P}_d(K) \coloneqq \mathscr{P}(K) \cap \mathbb{R}[x]_d.$$

2.1 Polynomial Optimization

In the following, we preview some basics of polynomial optimization. For a tuple $h \coloneqq (h_1, \ldots, h_s)$ of polynomials in $\mathbb{R}[x]$, let

$$\text{Ideal}[h] \coloneqq h_1 \cdot \mathbb{R}[x] + \cdots + h_s \cdot \mathbb{R}[x].$$

The degree-$2k$ truncation of Ideal$[h]$ is

$$\text{Ideal}[h]_{2k} \coloneqq h_1 \cdot \mathbb{R}[x]_{2k-\deg(h_1)} + \cdots + h_s \cdot \mathbb{R}[x]_{2k-\deg(h_s)}.$$

The real variety of h is

$$V_{\mathbb{R}}(h) \coloneqq \{x \in \mathbb{R}^n : h(x) = 0\}.$$

A polynomial $\sigma \in \mathbb{R}[x]$ is said to be a sum of squares (SOS) if there are polynomials $q_1, \ldots, q_t \in \mathbb{R}[x]$ such that $\sigma = q_1^2 + \cdots + q_t^2$. The convex cone of all SOS polynomials in x is denoted as $\Sigma[x]$. We refer to [3, 6, 7, 8] for more details.

For a tuple of polynomials $g \coloneqq (g_1, \ldots, g_t)$, its *quadratic module* is

$$\text{QM}[g] \coloneqq \left\{ \sum_{i=0}^{t} \sigma_i g_i : \sigma_i \in \Sigma[x], \text{ with } g_0 \coloneqq 1 \right\}.$$

For a positive integer k, the degree-$2k$ truncation of QM$[g]$ is

$$\text{QM}[g]_{2k} \coloneqq \left\{ \sum_{i=0}^{t} \sigma_i g_i : \sigma_i \in \Sigma[x],\ \deg(\sigma_i g_i) \leq 2k, \text{ with } g_0 \coloneqq 1 \right\}.$$

The quadratic module QM$[g]$ is said to be *archimedean* if there exists $q \in \text{QM}[g]$ such that the set $\{x \in \mathbb{R}^n : q(x) \geq 0\}$ is compact. The semialgebraic set defined by g is $S(g) \coloneqq \{x \in \mathbb{R}^n : g_1(x) \geq 0, \ldots, g_t(x) \geq 0\}$.

Theorem 2.1. [9] If QM$[g]$ is archimedean and a polynomial $f > 0$ on $S(g)$, then $f \in \text{QM}[g]$.

A vector $y := (y_\alpha)_{\alpha\in\mathbb{N}^n_{2k}}$ is said to be a *truncated multi-sequence* (tms) of degree $2k$. For $y \in \mathbb{R}^{\mathbb{N}^n_{2k}}$, the *Riesz functional* determined by y is the linear functional $\mathscr{L}_y$ acting on $\mathbb{R}[x]_{2k}$ such that

$$\mathscr{L}_y\left(\sum_{\alpha\in\mathbb{N}^n_{2k}} p_\alpha x^\alpha\right) := \sum_{\alpha\in\mathbb{N}^n_{2k}} p_\alpha y_\alpha.$$

For convenience, we denote

$$\langle p, y\rangle := \mathscr{L}_y(p) \text{ for } p \in \mathbb{R}[x]_{2k}.$$

The *localizing matrix* and *localizing vector* of p generated by y are

$$L_p^{(k)}[y] := \mathscr{L}_y\left(p(x)\cdot[x]_{s_1}[x]_{s_1}^T\right), \quad \mathscr{V}_p^{(2k)}[y] := \mathscr{L}_y\left(p(x)\cdot[x]_{s_2}\right), \tag{2.1}$$

respectively. In the above, the linear operator is applied component-wisely and

$$s_1 := k - \lceil \deg(p)/2\rceil, \quad s_2 := 2k - \deg(p). \tag{2.2}$$

We remark that $L_p^{(k)}[y] \succeq 0$ if and only if $\mathscr{L}_y \geq 0$ on $\mathrm{QM}[p]_{2k}$, and $\mathscr{V}_p^{(2k)}[y] = 0$ if and only if $\mathscr{L}_y = 0$ on $\mathrm{Ideal}[p]_{2k}$. More details for this can be found in [3, 6, 8].

2.2 Moment Relaxation

Moment relaxation is a technique used to find the global minimum of a polynomial optimization problem (POP) by solving a sequence of convex semidefinite programs (SDPs). Consider the general POP:

$$\begin{cases} f_{min} := & \min & f(x) \\ & s.t. & c_i(x) = 0 \ (i \in \mathcal{E}), \\ & & c_j(x) \geq 0 \ (j \in \mathcal{I}), \end{cases} \tag{2.3}$$

where f, c_i, c_j are real-valued polynomials in $x \in \mathbb{R}^n$.

The Moment-SOS relaxation method can be applied to any problem of the form (2.3). It generates a monotonically nondecreasing sequence of lower bounds on f_{min}. Crucially, this sequence of bounds is guaranteed to converge to the true global minimum f_{min} if the feasible set $K := \{x \in \mathbb{R}^n \mid c_i(x) = 0, c_j(x) \geq 0\}$ is compact. A sufficient condition for this convergence is that the quadratic module generated by the inequality constraints is Archimedean, as mentioned in Section 2.1.

Denote the degrees

$$\begin{aligned} d_0 &:= \max\{\lceil \deg(c_i)/2\rceil : i \in \mathcal{E}\cup\mathcal{I}\}, \\ d_1 &:= \max\{\lceil \deg(f)/2\rceil,\ d_0\}. \end{aligned}$$

For a relaxation order $k \geq d_1$, the k-th order moment relaxation of (2.3) is the SDP:

$$\begin{cases} f_{mom,k} := & \min\limits_{y} & \langle f, y\rangle \\ & s.t. & \mathscr{L}_y(c_i\cdot x^\alpha) = 0 \ \forall \alpha \text{ s.t. } |\alpha| \leq 2k - \deg(c_i) \ (i \in \mathcal{E}), \\ & & L_{c_j}^{(k)}[y] \succeq 0 \ (j \in \mathcal{I}), \\ & & M_k[y] \succeq 0, \\ & & y_0 = 1,\ y \in \mathbb{R}^{\mathbb{N}^n_{2k}}. \end{cases} \tag{2.4}$$

In the above, $M_k[y] := L_1^{(k)}[y]$ is the moment matrix. The dual of (2.4) is the k-th order SOS relaxation:

$$\begin{cases} f_{sos,k} := & \max\limits_{\gamma} \quad \gamma \\ & s.t. \quad f - \gamma \in \text{Ideal}[c_{\mathcal{E}}]_{2k} + \text{QM}[c_{\mathcal{I}}]_{2k}. \end{cases} \tag{2.5}$$

For $k = d_1, d_1 + 1, \ldots$, the primal-dual pair of SDPs (2.4)-(2.5) is called the Moment-SOS hierarchy. A fundamental property of this hierarchy is that the optimal values are always lower bounds for the true minimum.

Proposition 2.2. [8] For any relaxation order $k \geq d_1$, we have $f_{sos,k} \leq f_{mom,k} \leq f_{min}$.

This proposition implies that moment relaxations provide valid lower bounds for f_{min}. As mentioned, if the Archimedean condition holds (guaranteeing the compactness of the feasible set), the sequence of lower bounds converges to the global minimum, i.e., $\lim_{k\to\infty} f_{mom,k} = f_{min}$. This makes the Moment-SOS hierarchy a powerful tool for global polynomial optimization.

3 Constructing the Polynomial Approximation

In this section, we detail our method for constructing a polynomial function $\tilde{P}(z)$ that approximates the value function $P(z)$. The primary motivation for this approach is to leverage powerful tools from polynomial optimization. While general-purpose solvers like gradient descent often fail on nonconvex problems by becoming trapped in local minima, techniques such as moment-SOS relaxation can find the global minimum of a polynomial, even if it is nonconvex [3]. By replacing $P(z)$ with a polynomial approximation $\tilde{P}(z)$, we can apply these global optimization methods.

The validity of this approximation rests on a solid theoretical foundation. A key requirement is the continuity of the value function $P(z)$. Under our initial problem assumptions, $P(z)$ is guaranteed to be continuous on the compact set K_z by *Berge's Maximum Theorem*. We also assume standard technical conditions, such as the *restricted inf-compactness* (RIC) condition [2], to guarantee that the problem is well-posed and a finite minimum exists.

Given the continuity of $P(z)$ on the compact set K_z, the *Stone-Weierstrass Theorem* guarantees that it can be uniformly approximated by a polynomial to any desired degree of accuracy. This theorem formally justifies our strategy.

Theorem 3.1 (Stone-Weierstrass)**.** Let $K_z \subseteq \mathbb{R}^k$ be a compact set. The set of all polynomial functions is dense in $C(K_z)$, the space of all continuous real-valued functions on K_z. That is, for any function $P \in C(K_z)$ and any $\epsilon > 0$, there exists a polynomial function $\tilde{P}$ such that

$$\sup_{z\in K_z} |P(z) - \tilde{P}(z)| < \epsilon.$$

Grounded by this theoretical support, we can confidently proceed with our strategy. The remainder of this section details the practical construction of the approximating polynomial $\tilde{P}(z)$, which involves first building the approximation from samples of $P(z)$ and then finding its global minimum.

3.1 Univariate Case

To begin, we consider the case where $z \in \mathbb{R}$ for the sake of simplicity. We aim to approximate the continuous function $P(z)$ with a degree-m polynomial $\tilde{P}(z)$:

$$P(z) \approx \tilde{P}(z) \coloneqq p_0 + p_1 z + p_2 z^2 + \cdots + p_m z^m. \tag{3.1}$$

Our goal is to estimate the coefficient vector $\mathbf{vec}(p) = [p_0, \ldots, p_m]^T$. To do this, we first generate $N \geq m+1$ sample points $\{z_i\}_{i=1}^N$ from the compact domain K_z. While several sampling strategies exist (e.g., uniform grids or adaptive sampling), we adopt random sampling for its simplicity and effectiveness. For each sample z_i, we then compute the corresponding value of the function $P(z)$ by solving the inner optimization problem:

$$P(z_i) := \min_{x \in K_x} f(x, z_i). \tag{3.2}$$

This can be solved using any suitable numerical solver. For instance, gradient-based methods can be used if f is smooth, or global techniques like moment-SOS relaxations can be used if f is a polynomial.

Once we obtain the N sample pairs $\{(z_i, P(z_i))\}_{i=1}^N$, we set up a linear system to find the coefficients. This takes the form $Z \cdot \mathbf{vec}(p) \approx P$, where Z is the Vandermonde matrix and P is the vector of computed values:

$$\begin{bmatrix} 1 & z_1 & z_1^2 & \cdots & z_1^m \\ 1 & z_2 & z_2^2 & \cdots & z_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z_N & z_N^2 & \cdots & z_N^m \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_m \end{bmatrix} \approx \begin{bmatrix} P(z_1) \\ P(z_2) \\ \vdots \\ P(z_N) \end{bmatrix}. \tag{3.3}$$

When $N > m+1$, this overdetermined system is solved in the least-squares sense, i.e., we find the vector $\mathbf{vec}(p)$ that minimizes $\|Z \cdot \mathbf{vec}(p) - P\|_2^2$. If the points z_i are distinct, the matrix Z has full column rank, guaranteeing a unique solution. The computational cost of this step using standard methods like QR decomposition is $O(N(m+1)^2)$.

Once the polynomial approximation $\tilde{P}(z)$ is constructed, we replace the original problem (1.1) with the task of minimizing this polynomial:

$$\min_{z \in K_z} \quad \tilde{P}(z) = p_0 + p_1 z + \cdots + p_m z^m. \tag{3.4}$$

This is a constrained polynomial optimization problem over a compact set. Since $\tilde{P}(z)$ is a polynomial, we can apply global optimization techniques such as moment-SOS relaxation to find its true minimum, which is a key advantage over local descent methods that can get trapped in non-global optima.

3.2 Multivariate Case

We now extend the polynomial approximation framework to the multivariate case where $z \in \mathbb{R}^k$. The goal remains to construct a polynomial approximation $\tilde{P}(z)$ of the value function $P(z)$.

Let m be the total degree of the approximating polynomial. The polynomial takes the form:

$$\tilde{P}(z) = \sum_{|\alpha| \leq m} p_\alpha z^\alpha, \tag{3.5}$$

where $\alpha = (\alpha_1, \ldots, \alpha_k) \in \mathbb{N}^k$ is a multi-index with $|\alpha| = \sum_{i=1}^k \alpha_i \leq m$, and $z^\alpha = z_1^{\alpha_1} \cdots z_k^{\alpha_k}$. The total number of monomial terms is $D = \binom{k+m}{k}$.

The construction process follows the same three steps as the univariate case. We draw $N \geq D$ distinct sample points $\{z_i\}_{i=1}^N$ from the compact domain K_z and evaluate the corresponding values $P(z_i)$ by solving the inner optimization problem (3.2).

With the sample pairs, we solve a multivariate linear least-squares problem. We form the design matrix $Z_m \in \mathbb{R}^{N \times D}$, where each row is the vector of all monomials up to degree m evaluated at a point z_i. The system is then:

$$\underbrace{\begin{bmatrix} [z_1]_m^T \\ [z_2]_m^T \\ \vdots \\ [z_N]_m^T \end{bmatrix}}_{Z_m} \mathbf{vec}(p) \approx \underbrace{\begin{bmatrix} P(z_1) \\ P(z_2) \\ \vdots \\ P(z_N) \end{bmatrix}}_{P}. \tag{3.6}$$

We find the coefficient vector $\mathbf{vec}(p)$ by minimizing $\|Z_m \cdot \mathbf{vec}(p) - P\|_2^2$. The computational cost for solving this system scales as $O(ND^2)$. This cost becomes significant as the dimension k and degree m increase (a challenge known as the curse of dimensionality), but can be mitigated in some cases using sparse polynomial bases or other advanced techniques.

Once the coefficient vector $\mathbf{vec}(p)$ is obtained, we solve the resulting polynomial optimization problem:

$$\min_{z \in K_z} \quad \tilde{P}(z) = \sum_{|\alpha| \leq m} p_\alpha z^\alpha. \tag{3.7}$$

This is again a problem that can be solved for a global minimum using techniques such as moment-SOS relaxation.

4 Iterative Refinement of the Approximation

The initial polynomial approximation $\tilde{P}(z)$, constructed from a set of N samples, serves as a starting point. To improve its accuracy, we can adopt an iterative refinement strategy. This process involves sequentially adding new, informative data points to our sample set and updating the polynomial coefficients accordingly. A key aspect of this approach is the strategic selection of new sample points.

Simply adding another randomly chosen point may offer little new information. To maximize the benefit of an additional sample, we must choose it strategically. A powerful strategy is to select the point that is most relevant to our ultimate goal of finding the minimum of $P(z)$.

At a given iteration, we have the current polynomial approximation $\tilde{P}_t(z)$. We can find its global minimum:

$$z_t^* = \arg\min_{z \in K_z} \tilde{P}_t(z). \tag{4.1}$$

This point z_t^* represents our current best estimate for the true minimizer. We then select this point as our next sample, $z_{new} = z_t^*$, and compute the true function value $P(z_{new})$. Adding the pair $(z_{new}, P(z_{new}))$ to our dataset forces the model to correct its estimation error precisely at the location it deemed most promising, effectively steering the approximation towards the true minimum of $P(z)$.

Adding a new data point $(z_{new}, P(z_{new}))$ corresponds to appending a new row to the design matrix Z_m and a new entry to the observation vector P. A naive approach would be to re-solve the entire least-squares problem from scratch with the augmented $(N+1)\times D$ system. However, repeatedly solving a growing system of equations is computationally inefficient, especially if many refinement steps are desired.

This computational challenge motivates the need for a more intelligent approach. The following section introduces how the Sherman-Morrison formula can be used to perform this update efficiently, allowing for rapid and low-cost refinement of the polynomial coefficients.

4.1 Sherman-Morrison Formula

Instead of re-solving the linear system from scratch at each iteration, we use the Sherman-Morrison formula to efficiently perform a rank-one update on the solution. The standard least-squares solution to the system $Z_m \cdot \mathbf{vec}(p) \approx P$ is given by the normal equations:

$$\mathbf{vec}(p) = (Z_m^T Z_m)^{-1} Z_m^T P. \tag{4.2}$$

Let us denote the inverse matrix we store and update as $A_t := (Z_{m,t}^T Z_{m,t})^{-1}$ at iteration t. When a new data point gives a new monomial row vector $[z_{new}]_m^T$, the matrix to be inverted, $Z_{m,t+1}^T Z_{m,t+1}$, is a rank-one update of $Z_{m,t}^T Z_{m,t}$:

$$Z_{m,t+1}^T Z_{m,t+1} = Z_{m,t}^T Z_{m,t} + [z_{new}]_m [z_{new}]_m^T. \tag{4.3}$$

The Sherman-Morrison formula provides an explicit formula for the new inverse A_{t+1} based on the old inverse A_t:

$$A_{t+1} = A_t - \frac{A_t [z_{new}]_m [z_{new}]_m^T A_t}{1 + [z_{new}]_m^T A_t [z_{new}]_m}. \tag{4.4}$$

The updated coefficient vector $\mathbf{vec}(p)_{t+1}$ can then be computed efficiently using this new inverse. This update procedure avoids a full matrix inversion at each step, and the overall iterative scheme is detailed in Algorithm 1.

4.2 Computational Costs

Here, we compare the computational costs of finding the new coefficient vector after adding a single data point.

Solving from Scratch The standard method to solve the least-squares problem with an updated design matrix $Z_m \in \mathbb{R}^{(N+1)\times D}$ is to use a QR decomposition, at a cost of $O((N+1)D^2)$, or simply $O(ND^2)$. Alternatively, forming the normal equations matrix $Z_m^T Z_m$ also costs $O(ND^2)$. This cost is incurred at each refinement iteration.

Algorithm 1 Iterative Polynomial Approximation Algorithm

Input: Objective function $f(x, z)$, domains K_x, K_z, polynomial degree m.
Parameters: Initial sample size $N \geq D = \binom{k+m}{k}$, max iterations T, tolerance $\varepsilon > 0$.

Phase 1: Initialization
Choose N random points $\{z_i\}_{i=1}^{N}$ from K_z.
Compute $P_i \leftarrow \min_{x \in K_x} f(x, z_i)$ for $i = 1, \ldots, N$.
Construct initial design matrix Z_m and observation vector P.
Compute $\mathbf{vec}(p)_0 \leftarrow (Z_m^T Z_m)^{-1}(Z_m^T P)$.
Store $A \leftarrow (Z_m^T Z_m)^{-1}$ and $q \leftarrow Z_m^T P$.
Initialize $t \leftarrow 0$, $P_{best} \leftarrow \infty$.

Phase 2: Iterative Refinement
while $t < T$ **do**
    Define current polynomial $\tilde{P}_t(z)$ using coefficient vector $\mathbf{vec}(p)_t$.
    Find minimizer of approximation: $z_{new} \leftarrow \arg\min_{z \in K_z} \tilde{P}_t(z)$.
    Calculate minimum value of approximation: $\tilde{P}_{min} \leftarrow \tilde{P}_t(z_{new})$.
    if $t > 0$ and $|\tilde{P}_{min} - \tilde{P}_{prev\_min}| < \varepsilon$ **then**
        break ▷ Converged
    end if
    $\tilde{P}_{prev\_min} \leftarrow \tilde{P}_{min}$.
    Evaluate true function: $P_{new} \leftarrow \min_{x \in K_x} f(x, z_{new})$.
    if $P_{new} < P_{best}$ **then**
        $P_{best} \leftarrow P_{new}$, $z_{best} \leftarrow z_{new}$.
    end if
    Form new monomial vector $[z_{new}]_m$ from point z_{new}.
    Update inverse matrix: $A \leftarrow A - (A[z_{new}]_m [z_{new}]_m^T A)/(1 + [z_{new}]_m^T A [z_{new}]_m)$.
    Update right-hand side vector: $q \leftarrow q + [z_{new}]_m P_{new}$.
    Update coefficient vector: $\mathbf{vec}(p)_{t+1} \leftarrow Aq$.
    $\mathbf{vec}(p)_t \leftarrow \mathbf{vec}(p)_{t+1}$, $t \leftarrow t + 1$.
end while

Output: The best found solution (z_{best}, P_{best}).

Updating with Sherman-Morrison In contrast, the Sherman-Morrison update avoids re-forming large matrices. The update consists of several matrix-vector and vector-vector operations. The most expensive operation is a matrix-vector product like $A_k[z_{new}]_m$, which costs $O(D^2)$, where $A_k \in \mathbb{R}^{D \times D}$. The total cost for an update is therefore dominated by operations of complexity $O(D^2)$. $O(D^2)$ for the update versus $O(ND^2)$ for re-solving. As the number of total samples N grows, the Sherman-Morrison formula offers a significant computational advantage, making the iterative refinement strategy practical.

4.3 Equivalence of the Iterative Update

In this section, we formally prove that the coefficient vector $\mathbf{vec}(p)$ produced by our iterative refinement process is mathematically identical to the vector that would be obtained by solving a single, large least-squares problem with all data points at once.

Proposition 4.1. Let $\mathbf{vec}(p)_k$ be the least-squares solution using a set of k data points, represented by the design matrix Z_k and observation vector P_k. Let a new data point (z_{new}, P_{new}) be added, forming the new matrices Z_{k+1} and P_{k+1}. The new coefficient vector $\mathbf{vec}(p)_{k+1}$ calculated via the Sherman-Morrison update is identical to the direct least-squares solution with all $k+1$ points.

Proof. The direct least-squares solution for all $k+1$ data points is given by the normal equations:

$$\mathbf{vec}(p)_{direct} = (Z_{k+1}^T Z_{k+1})^{-1}(Z_{k+1}^T P_{k+1}). \tag{4.5}$$

Let $[z_{new}]_m^T$ be the new row vector of monomials corresponding to the point z_{new}. By construction, the new matrices can be written in block form:

$$Z_{k+1} = \begin{bmatrix} Z_k \\ [z_{new}]_m^T \end{bmatrix}, \quad P_{k+1} = \begin{bmatrix} P_k \\ P_{new} \end{bmatrix}.$$

We can expand the terms in (4.5):

$$Z_{k+1}^T Z_{k+1} = \begin{bmatrix} Z_k^T & [z_{new}]_m \end{bmatrix} \begin{bmatrix} Z_k \\ [z_{new}]_m^T \end{bmatrix} = Z_k^T Z_k + [z_{new}]_m [z_{new}]_m^T. \tag{4.6}$$

$$Z_{k+1}^T P_{k+1} = \begin{bmatrix} Z_k^T & [z_{new}]_m \end{bmatrix} \begin{bmatrix} P_k \\ P_{new} \end{bmatrix} = Z_k^T P_k + [z_{new}]_m P_{new}. \tag{4.7}$$

Substituting (4.6) and (4.7) back into (4.5), we get:

$$\mathbf{vec}(p)_{direct} = (Z_k^T Z_k + [z_{new}]_m [z_{new}]_m^T)^{-1}(Z_k^T P_k + [z_{new}]_m P_{new}). \tag{4.8}$$

Now, we apply the Sherman-Morrison formula to the inverted term in (4.8). Let $A_k = (Z_k^T Z_k)^{-1}$. The formula states:

$$(Z_k^T Z_k + [z_{new}]_m [z_{new}]_m^T)^{-1} = A_k - \frac{A_k [z_{new}]_m [z_{new}]_m^T A_k}{1 + [z_{new}]_m^T A_k [z_{new}]_m}.$$

This resulting matrix is precisely the updated inverse matrix A_{k+1} used in our iterative algorithm. Substituting this back into (4.8) gives:

$$\mathbf{vec}(p)_{direct} = \left(A_k - \frac{A_k [z_{new}]_m [z_{new}]_m^T A_k}{1 + [z_{new}]_m^T A_k [z_{new}]_m} \right) (Z_k^T P_k + [z_{new}]_m P_{new}) = A_{k+1}(Z_{k+1}^T P_{k+1}).$$

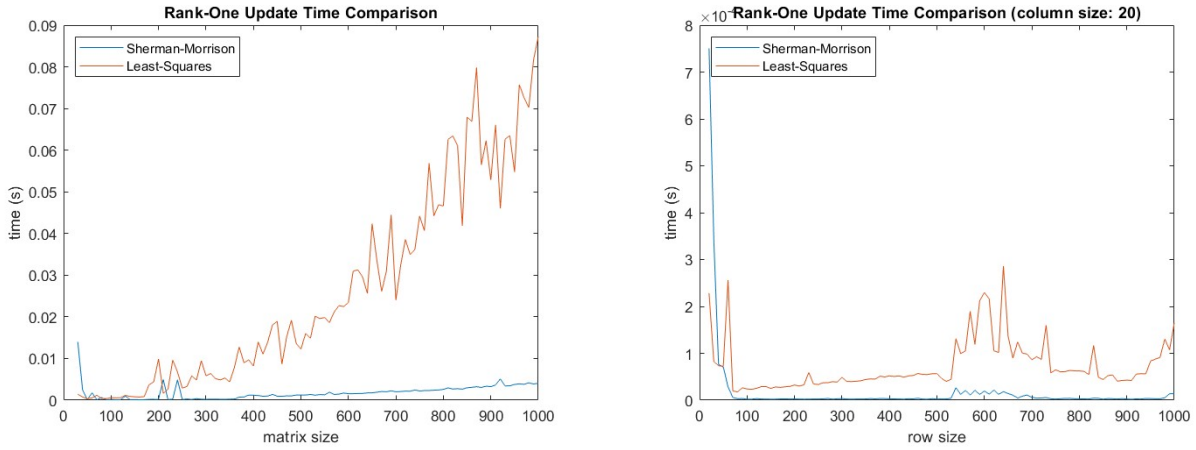
This final expression is exactly the formula for the updated coefficient vector $\mathbf{vec}(p)_{k+1}$ from our iterative procedure. Thus, the direct solution and the sequentially updated solution are identical. □

5 Numerical Experiments

In this section, we present numerical experiments to demonstrate the effectiveness and practical behavior of our proposed polynomial approximation framework. All computations are implemented using MATLAB R2024a on an Inspiron 16 7630 2-in-1 equipped with a 13th Gen Intel(R) Core(TM) i5-1335U processor and 8GB RAM. The polynomial optimization problems are solved using the software `Gloptipoly` [4], which calls the SDP package `SeDuMi` [10]. For neatness, all computational results are displayed to four decimal digits.

5.1 Experimental Validation of the Update Method's Efficiency

Here, we experimentally validate the computational efficiency of the Sherman-Morrison formula compared to direct re-computation for rank-one updates. While our computational cost analysis in Section 4.2 showed an advantage, these results provide empirical confirmation. The Sherman-Morrison formula updates the existing inverse matrix in $O(D^2)$ time, whereas the standard approach of re-inverting the matrix from scratch costs $O(D^3)$, where D is the matrix dimension.



(a) Square Matrices

(b) Rectangular Matrices

Figure 1: Time comparison for a single rank-one update to $(A^\top A)^{-1}$ when appending a row to A. (a)The y-axis shows the computation time in seconds for updating the inverse of a $D \times D$ matrix with Sherman-Morrison ($O(D^2)$) vs. full re-inversion ($O(D^3)$), where D is the matrix size on the x-axis. (b)Fixed column size (20), showing update time versus row count. Both plots confirm the theoretical advantage of Sherman-Morrison, with direct recomputation becoming prohibitively expensive for larger matrices.

To illustrate this performance difference, we measured the time required to update the inverse of matrices of increasing size D using both methods. The results, shown in Figure 1, provide empirical confirmation of the theoretical complexity across two scenarios: square matrices (Figure 1a), where D is the dimension of matrix A, showing the expected scaling as D grows, and rectangular matrices (Figure 1b), with fixed column size (20), demonstrating how update time scales with row count. In both cases, the Sherman-Morrison formula significantly outperforms direct re-inversion. While the methods perform comparably for

small D, the cost of re-computation grows rapidly, becoming impractical for larger matrices. In contrast, the Sherman-Morrison update maintains efficiency even at scale, with sub-millisecond times for $D \leq 1000$ in the square case and linear growth in row count for rectangular matrices.

These findings confirm that the Sherman-Morrison formula is a highly effective strategy for the iterative refinement step of our algorithm, making it a preferred choice for applications requiring frequent updates, especially in high-dimensional settings.

5.2 Numerical Examples

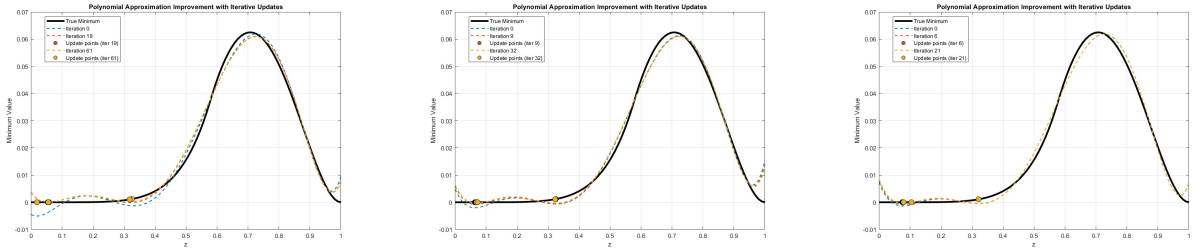
Example 5.1. Consider the dehomogenized Motzkin polynomial $f(x, z) = x^4 + x^2 + z^6 - 3x^2z^2$ on the compact domain $(x, z) \in [-1, 1] \times [0, 1]$. This polynomial is a well-known example of a nonnegative polynomial that is not a sum of squares (SOS) [8]. The optimization problem is:

$$\min_{z\in[0,1]} \left\{ P(z) := \min_{x\in[-1,1]} f(x, z) \right\}.$$

We apply our iterative algorithm to solve this problem. To analyze the impact of the initial sample size, we run the experiment with three different sizes: $N \in \{10, 50, 100\}$. For all cases, the approximating polynomial degree is $m = 6$, and the refinement process terminates when the change in the approximated minimum is less than $\varepsilon = 10^{-6}$ or a maximum of $T = 100$ iterations is reached.

The initial polynomial $\tilde{P}_0(z)$ is constructed for each case by solving the inner minimization problem at N points sampled uniformly at random from $[0, 1]$. We then apply the iterative refinement procedure as described in Algorithm 1.

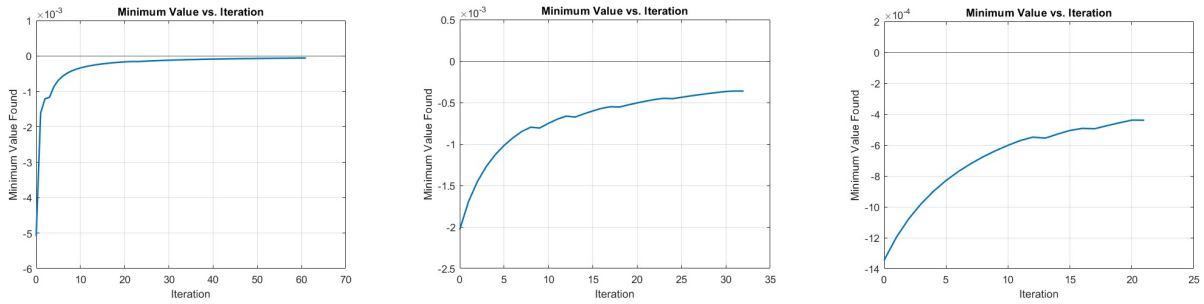
Figure 2 shows how the polynomial approximation improves over iterations for each initial sample size. A larger N results in a better initial approximation, which requires fewer iterations to closely match the true value function (the solid black curve). Figure 3 tracks the convergence of the minimum value of the approximation, $\min \tilde{P}_k(z)$, over the iterations. All three cases converge towards the true minimum of 0.



(a) $N = 10$ (b) $N = 50$ (c) $N = 100$

Figure 2: Polynomial approximation improvement with different initial sample sizes N. The initial approximation (blue) is refined over iterations (red, yellow) to better match the true value function (black).

The results, summarized in Table 1a, highlight an important trade-off. A smaller initial sample size ($N = 10$) requires more refinement iterations (61) to converge. In contrast, a larger initial sample size ($N = 100$) requires significantly fewer iterations (23). This suggests



(a) $N = 10$ (b) $N = 50$ (c) $N = 100$

Figure 3: Convergence of the approximated minimum value over iterations for each case.

that while a larger N increases the cost of the initial model construction, it can reduce the number of expensive iterative refinement steps, each of which requires solving a global polynomial optimization problem. For the $N = 50$ case, the final approximated polynomial after 32 iterations is:

$$\tilde{P}_{\text{final}}(z) = 0.0065 - 0.2557z + 3.2571z^2 - 16.7981z^3 + 39.1733z^4 - 40.1089z^5 + 14.7386z^6.$$

The time comparison in Table 1b for this case also confirms the computational advantage of the Sherman-Morrison update.

N	Iterations	Initial Min	Final Min
10	61	-0.0051	-0.0001
50	32	-0.0020	-0.0004
100	21	-0.0013	-0.0004

(a) Convergence Results vs. Initial Sample Size

Method ($N = 50$)	Avg Time	Total Time
Regular	0.0002s	0.0054s
S-M	0.0001s	0.0037s

(b) Update Time Comparison

Table 1: Summary of experimental results for the Motzkin polynomial example.

Example 5.2. Consider the dehomogenized Robinson polynomial, a classical example of a nonnegative polynomial that is not a sum of squares (SOS):

$$f(x_1, x_2, z) = x_1^6 + x_2^6 + z^6 + 3x_1^2x_2^2z^2 - (x_1^4x_2^2 + x_1^2x_2^4 + x_1^4z^2 + x_1^2z^4 + x_2^4z^2 + x_2^2z^4).$$

We aim to solve the min-min problem for $x = (x_1, x_2)$ on the unit disk and $z \in [-1.5, 1.5]$:

$$\min_{z\in[-1.5,1.5]} \left\{ P(z) \coloneqq \min_{\|x\|_2\leq 1} f(x, z) \right\}.$$

We apply our framework with a degree $m = 6$ polynomial approximation, starting with $N = 10$ random samples. The inner minimization problems are solved using moment relaxation via `GloptiPoly`. After 4 iterations, the refinement process terminates ($\varepsilon = 10^{-4}$).

$$\begin{aligned}\tilde{P}_{\text{initial}}(z) &= 0.0026 + 0.0015z - 2.6630z^2 + 0.0001z^3 - 1.1322z^4 - 0.001z^5 + 1.0288z^6\\ \tilde{P}_{\text{final}}(z) &= 0.0031 + 0.0012z - 2.6689z^2 + 0.0002z^3 - 1.1233z^4 - 0.0008z^5 + 1.0259z^6\end{aligned}$$

As shown in Figure 4, the final approximation $\tilde{P}_{\text{final}}(z)$ captures the shape of the true value function $P(z)$ with remarkable accuracy. However, a significant discrepancy in value is observed. While the true minimum of $P(z)$ is 0, our method reports a minimum of approximately -3.1239 (Table 2a).

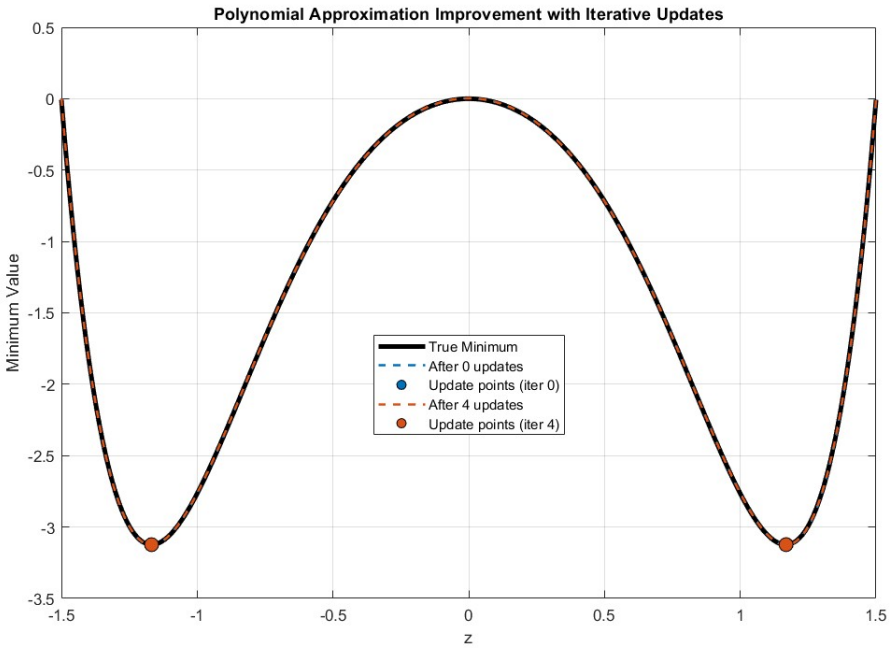


Figure 4: Polynomial approximation ($m = 6$) for the Robinson polynomial. The approximation curve (dashed) closely matches the shape of the value function computed by the inner solver (solid).

This difference does not represent a failure of our polynomial approximation method. Rather, it highlights that our method accurately learns the data it is given. The discrepancy arises because moment relaxation provides a non-tight lower bound for this specific problem. As shown in Table 2a, `GloptiPoly` computes the minimum of the inner problem as -3.1233, not 0. Our algorithm successfully learns this value function, but inherits its inherent error.

Furthermore, the time comparison in Table 2b shows that the Sherman-Morrison update was slower for this case. This is attributable to the very small problem size (both the number of coefficients $D = 7$ and the number of samples $N = 10$), where the overhead of the iterative method can exceed the cost of a direct solve. The asymptotic advantage of Sherman-Morrison, as demonstrated theoretically and in other experiments, becomes evident in problems with a larger number of coefficients.

Example 5.3. Consider the dehomogenized Schmüdgen polynomial, where the value function $P(z)$ depends on a bivariate parameter $z = (z_1, z_2) \in \mathbb{R}^2$:

$$f(x, z) = 200(x^3 - 4xz_2^2)^2 + (z_1^3 - 4z_1z_2^2)^2 + (z_1^2 - x^2)x(x + 2z_2)(x^2 - 2xz_2 + 2z_1^2 - 8z_2^2).$$

We aim to solve the min-min problem on the domain $(x, z) \in [-1, 1] \times ([-1, 1] \times [-1, 1])$:

$$\min_{z\in[-1,1]^2} \left\{ P(z) \coloneqq \min_{x\in[-1,1]} f(x, z) \right\}.$$

We approximate $P(z)$ with a bivariate polynomial $\tilde{P}(z)$ of total degree $m = 5$, which has $D = \binom{2+5}{5} = 21$ coefficients. We begin with $N = 50$ random samples and apply our iterative refinement algorithm.

After 93 iterations, the process converges ($\varepsilon = 10^{-4}$). The initial and final approximations are given below:

$$\begin{aligned}\tilde{P}_{\text{initial}}(z) = {} & 0.1838 + 0.8926z_1 + 0.7511z_2 - 0.8513z_1^2 + 1.5883z_1z_2 - 1.7312z_2^2 \\ & - 5.7813z_1^3 - 0.1409z_1^2z_2 - 2.2676z_1z_2^2 - 3.8164z_2^3 + 0.4059z_1^4 \\ & - 2.8898z_1^3z_2 + 7.4409z_1^2z_2^2 - 1.1196z_1z_2^3 + 2.6172z_2^4 + 6.4156z_1^5 \\ & - 3.6141z_1^4z_2 + 5.9898z_1^3z_2^2 + 3.3235z_1^2z_2^3 + 1.0870z_1z_2^4 + 3.4136z_2^5\end{aligned}$$

$$\begin{aligned}\tilde{P}_{\text{final}}(z) = {} & 0.4146 - 0.0900z_1 + 0.1104z_2 - 1.5586z_1^2 - 0.1021z_1z_2 - 2.5984z_2^2 \\ & - 0.1789z_1^3 + 0.9296z_1^2z_2 + 0.9112z_1z_2^2 - 0.5666z_2^3 + 1.5207z_1^4 \\ & - 1.3159z_1^3z_2 + 7.6308z_1^2z_2^2 + 1.4555z_1z_2^3 + 3.7771z_2^4 + 0.5434z_1^5 \\ & - 2.5378z_1^4z_2 + 2.2058z_1^3z_2^2 + 0.1835z_1^2z_2^3 - 1.6872z_1z_2^4 + 0.6533z_2^5.\end{aligned}$$

Figure 5 visualizes the progress of the algorithm. The true value function $P(z)$ is a complex surface, as illustrated by the yellow-to-blue gradient color for reference. To illustrate convergence, we plot the minimum value of our polynomial approximation at different stages as horizontal planes. The sequence of planes (blue for initial, yellow for final) rises towards the true global minimum of 0, demonstrating the effectiveness of the refinement. Figure 6 shows this convergence path more directly.

The results are summarized in Table 3. The final approximated minimum value, -0.0480, is significantly closer to the true minimum of 0 than the initial estimate of -2.0828. Furthermore, the time comparison confirms that the Sherman-Morrison update is substantially more efficient in this higher-dimensional setting, providing considerable runtime savings.

Example 5.4. This example tests the framework's performance on a problem with a higher-dimensional inner variable, $x \in \mathbb{R}^3$. We consider the dehomogenized Horn polynomial:

$$f(x, z) = (x_1^2 + x_2^2 + x_3^2 + z_1^2 + z_2^2)^2 - 4(x_1^2x_2^2 + x_2^2x_3^2 + x_3^2z_1^2 + z_1^2z_2^2 + z_2^2x_1^2).$$

The min-min optimization problem is defined for $z \in [-2, 2]^2$ and x on the unit ball, $\|x\|_2 \leq 1$:

$$\min_{z\in[-2,2]^2} \left\{ P(z) \coloneqq \min_{\|x\|_2\leq 1} f(x, z) \right\}.$$

Function	Minimum
True $f(x, z)$	0
$P(z_n)$ via Moment-SOS	-3.1233
$\tilde{P}_{\text{initial}}(z)$	-3.1257
$\tilde{P}_{\text{final}}(z)$	-3.1239

(a) Comparison of Minimum Values

Method	Avg Time	Total Time
Regular	0.0025s	0.0100s
S-M	0.0064s	0.0257s

(b) Update Time Comparison

Table 2: Summary of results for the Robinson polynomial example.

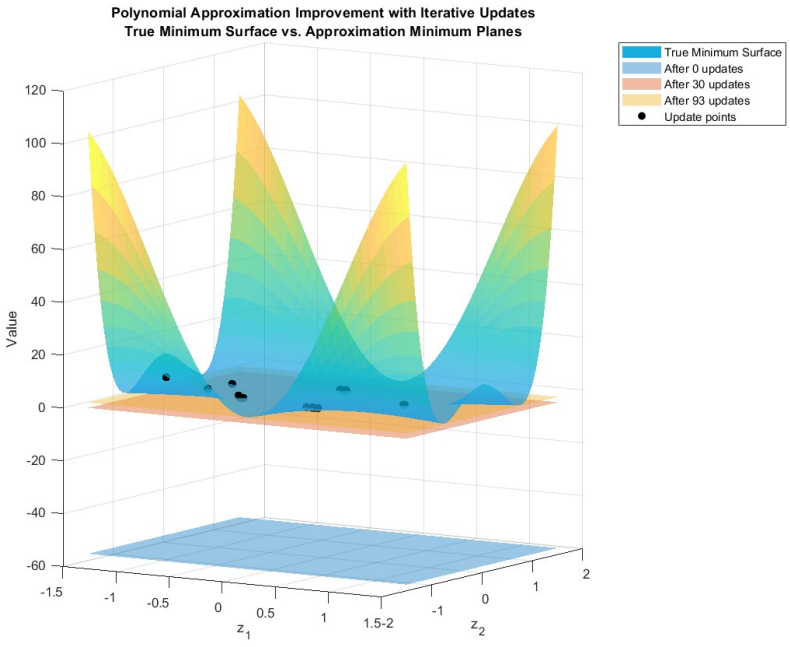


Figure 5: Visualization of the approximation improvement for the Schmüdgen polynomial. The true value surface $P(z)$ is shown at the bottom. The horizontal planes represent the minimum value of the polynomial approximation $\tilde{P}_k(z)$ at different iterations (0, 30, and 93), showing convergence toward the true minimum.

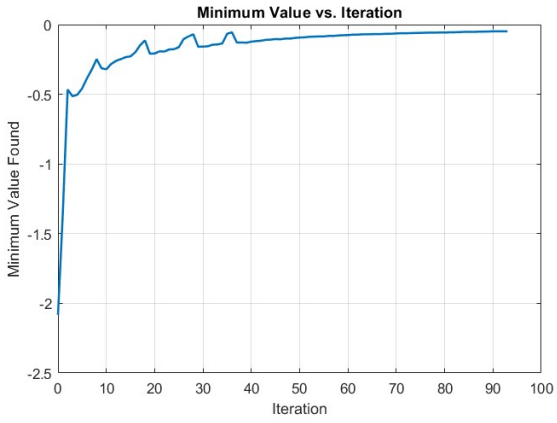


Figure 6: Convergence of the approximated minimum value over 93 iterations for the Schmüdgen polynomial.

To approximate the bivariate value function $P(z)$, we use a polynomial $\tilde{P}(z)$ of total degree $m = 3$, which has $D = \binom{2+3}{3} = 10$ coefficients. The initial model is built from $N = 100$ random samples.

The initial approximation $\tilde{P}_{\text{initial}}(z)$ is found by solving the least-squares problem. The iterative refinement process is then applied for a maximum of 100 iterations. The initial and

Function	Minimum
True $f(x, z)$	0
$\tilde{P}_{\text{initial}}(z)$	-2.0828
$\tilde{P}_{\text{final}}(z)$	-0.0480

(a) Minimum Values

Method	Avg Time (s)	Total Time (s)
Regular	0.0003	0.0322
S-M	0.0001	0.0095

(b) Update Time Comparison

Table 3: Summary of results for the Schmüdgen polynomial example.

final polynomials are:

$$\tilde{P}_{\text{initial}}(z) = 0.1322 - 0.3820z_1 - 0.1833z_2 + 0.5016z_1^2 - 0.1001z_1z_2 + 0.4256z_2^2 + 0.1497z_1^3 + 0.0437z_1^2z_2 - 0.0071z_1z_2^2 + 0.0871z_2^3.$$

$$\tilde{P}_{\text{final}}(z) = 0.1059 - 0.3960z_1 - 0.1994z_2 + 0.5088z_1^2 - 0.1000z_1z_2 + 0.4335z_2^2 + 0.1540z_1^3 + 0.0461z_1^2z_2 - 0.0058z_1z_2^2 + 0.0917z_2^3.$$

The results demonstrate that our method remains effective even with a more complex inner problem structure. Figure 7 visualizes the convergence, with the horizontal plane representing the minimum of our approximation rising towards the true minimum. Figure 8 shows the path of this convergence over the 100 iterations. The final estimated minimum of 0.0074 is a significant improvement over the initial 0.0408 (Table 4a). The efficiency of the Sherman-Morrison update is again confirmed in Table 4b.

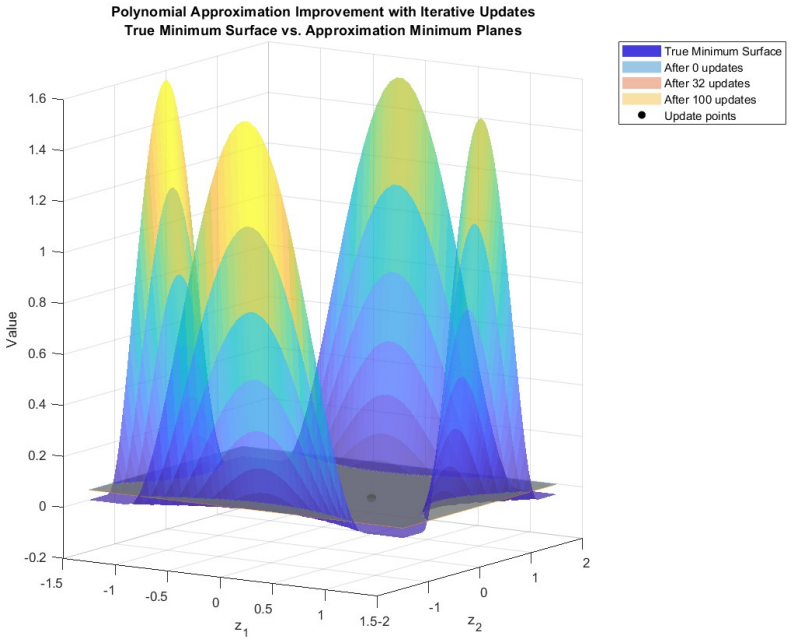


Figure 7: Approximation improvement for the Horn polynomial. Horizontal planes show the minimum value of the approximation at different iterations.

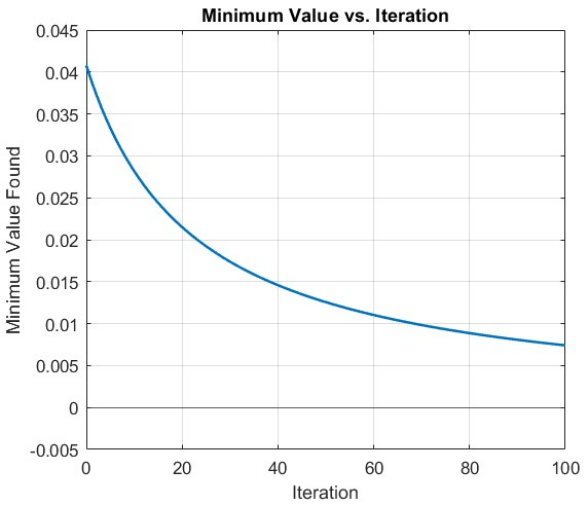


Figure 8: Convergence of the approximated minimum value over 100 iterations for the Horn polynomial.

Function	Minimum
True $f(x, z)$	0
$\tilde{P}_{\text{initial}}(z)$	0.0408
$\tilde{P}_{\text{final}}(z)$	0.0074

(a) Minimum Values

Method	Avg Time (s)	Total Time (s)
Regular	0.0005	0.0473
S-M	0.0002	0.0238

(b) Update Time Comparison

Table 4: Summary of results for the Horn polynomial example.

6 Conclusion

In this thesis, we developed and analyzed a computational framework for solving min-min optimization problems, particularly those where the value function is intractable. Our primary research question was how to efficiently find a global minimum for such nested problems without relying on traditional, often infeasible, nested-loop methods.

Our approach was to replace the difficult value function $P(z)$ with a tractable polynomial approximation, $\tilde{P}(z)$. We presented a three-stage methodology: (1) an initial approximation from sampled data using least-squares, (2) global optimization of the resulting polynomial, and (3) an iterative refinement strategy. The refinement stage, a key contribution of this work, strategically selects new sample points at the minimum of the current approximation and uses the Sherman-Morrison formula to efficiently update the polynomial coefficients. Our numerical experiments demonstrated that this iterative process successfully improves the solution accuracy and confirmed the computational advantages of the update formula.

The significance of this work lies in providing a practical and computationally flexible paradigm for a challenging class of optimization problems. By decoupling the expensive inner and outer optimization loops, our method is well-suited for parallel and distributed computing environments, thereby making previously intractable problems more approachable.

However, our framework has limitations that open avenues for future work. First, while our experiments demonstrate practical convergence, this thesis does not provide a formal proof that the sequence of generated solutions converges to the true global optimum. Such an analysis remains a non-trivial future challenge. Second, our iterative refinement relies on a purely exploitative sampling strategy (selecting the current minimum), which could risk premature convergence if the initial approximation is poor. Lastly, the method's practicality is subject to the curse of dimensionality in the parameter space.

Several avenues for future research arise from these limitations. A primary theoretical goal would be to establish conditions under which the algorithm is guaranteed to converge. On the practical side, exploring more sophisticated sampling strategies that balance exploitation with exploration could enhance the robustness of the refinement process. To combat the curse of dimensionality, investigating alternative approximation bases like sparse polynomials or tensor-train formats is a promising direction. The framework's potential extension to min-max problems, though requiring additional care for inner maximization, could further broaden its applicability in robust optimization and game theory. Finally, applying the framework to a wider range of real-world problems would further validate its utility.

In conclusion, this thesis has demonstrated that iterative polynomial approximation provides a powerful and adaptable framework for solving nested optimization problems. This approach effectively bridges the gap between global optimization theory and computational practice.

Acknowledgements

I would like to express my deepest gratitude to Professor Jiawang Nie for his generous guidance as my thesis advisor and for welcoming me into the honors program. I am also sincerely thankful to Jiyoung Choi for her patient mentorship and insightful feedback throughout this research. My appreciation extends to the UCSD Department of Mathematics for this academic opportunity, and to my family for their unwavering support.

References

[1] Gladin, E., Sadiev, A., Gasnikov, A., Dvurechensky, P., Beznosikov, A., and Alkousa, M. Solving smooth min-min and min-max problems by mixed oracle algorithms. *Communications in Computer and Information Science* (2021), 19–40.

[2] Guo, L., Lin, G.-H., Ye, J. J., and Zhang, J. Sensitivity analysis of the value function for parametric mathematical programs with equilibrium constraints. *SIAM Journal on Optimization 24*, 3 (2014), 1206–1237.

[3] Henrion, D., Korda, M., and Lasserre, J. B. *The Moment-SOS Hierarchy.* World Scientific, 2020.

[4] Henrion, D., Lasserre, J., and Lofberg, J. *Gloptipoly 3: moments, optimization and semidefinite programming.* Optimization Methods & Software, 2009, p. 761–779.

[5] Kovalev, D., Gasnikov, A., and Malinovsky, G. Optimal and nearly optimal algorithms for min-min optimization, 2023.

[6] Lasserre, J. B. *Introduction to polynomial and semi-algebraic optimization.* Cambridge University Press, 2015.

[7] Laurent, M. *Sums of Squares, Moment Matrices and Optimization Over Polynomials.* Emerging Applications of Algebraic Geometry of IMA Volumes in Mathematics and its Applications, Springer, 2009, ch. 7.

[8] Nie, J. *Moment and Polynomial Optimization.* SIAM - Society for Industrial and Applied Mathematics, 2023.

[9] Putinar, M. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal* (1993).

[10] Sturm, J. *Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones.* Optimization methods and software, 1999, p. 625–653.

[11] Weretka, M. An ordinal theorem of the maximum. *Economic Theory 76*, 1 (2022), 353–373.