ADAPTIVE PRECONDITIONING PARADIGMS FOR OPTIMIZATION ALGORITHMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MATHEMATICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Ya-Chi Chu
April 2026

Preface

This thesis spans several fields in optimization algorithms, specifically encompassing second-order methods accelerated by randomized numerical linear algebra (RandNLA), adaptive preconditioning by online learning in gradient-based methods, and hard constraint enforcement on neural networks. Despite the breadth of these topics, the central, unifying theme throughout this work is *preconditioning*. This focus is encapsulated in the dissertation's title, *Adaptive Preconditioning Paradigms for Optimization Algorithms*.

Before joining Madeleine's group, I had already developed a strong interest in optimization during my master's degree in Taiwan, where I focused on the hidden convexity of nonconvex quadratically constrained quadratic programs. Those works were primarily theoretical, and I had not yet been exposed to the practical optimization algorithms. My computational research starts after joining Madeleine's group in my second year PhD. I had the privilege to collaborate with visiting scholar Professor Luiz-Rafael Santos, whom we usually called Rafa. Together with Rafa and Madeleine, we sought to accelerate the generalized eigenvalue reformulation of the trust-region subproblem utilizing RandNLA. While this initial exploration did not culminate in a standalone publication, the methodological insights gained were instrumental. The concepts and techniques we explored carried directly into a subsequent project on interior-point methods, which is one of the most celebrated algorithmic advances in the history of optimization. This work forms Chapter 2 of this thesis.

Motivated by this introduction to RandNLA, I expanded my collaboration beyond Madeleine's group. I partnered with Professor Alice Cortinovis, who was a postdoctoral scholar in mathematics at Stanford back to 2024. This collaboration yielded a published paper that, while not included as a chapter herein, deepened my understanding of the theoretical foundations of randomized algorithms.

After wrapping up the RandNLA work, I was fortunate to collaborate with my colleague Wenzhi Gao. We worked on an online learning framework for adaptive preconditioning in gradient-based optimization methods, called the Online Scaled Gradient Method (`OSGM`). This project facilitated my transition into first-order optimization and online convex optimization. This work was ultimately expanded into a broader study of hypergradient-based adaptive stepsizes, which is presented in Chapter 3.

In the summer of 2025, I completed an internship at Apple, developing machine learning models

for iPhone sales demand forecasting. My primary objective was to reconcile forecasts across hierarchical data structures to ensure consistency. This experience highlighted the practical importance of enforcing hard constraints in applied machine learning. Inspired by these practical challenges, my final year of doctoral research shifted toward the problem of hard constraint enforcement in neural networks, resulting in the development of `SnareNet`, a framework presented in Chapter 4.

Reflecting on this academic trajectory, I am grateful for how considerably my perspective has broadened. When I initially joined Professor Udell's group, I faced a steep learning curve navigating the interdisciplinary presentations given by colleagues from diverse backgrounds. Over time, through successive engagements across pure optimization, randomized algorithms, online learning, and applied machine learning, I progressively built the context necessary to engage with frontier research in each of these domains. I deeply appreciate the intellectual richness of this journey and for the world-class researchers I have had the privilege to learn from and collaborate alongside.

Acknowledgments

First and foremost, I would like to thank my advisor, Professor Madeleine Udell, for her guidance on my research, her patience with my growth, and her encouragement during challenging times. I deeply appreciate her enduring optimism regarding her students and their research; I have learned a lot from her and the group she has built. I am equally grateful to my co-advisor, Professor Lexing Ying, for his generous support navigating both the administrative and research aspects of my PhD. In addition, I want to thank my early PhD collaborator, Professor Luiz-Rafael Santos (Rafa), who acted as another co-advisor to me during his time at Stanford. He taught me invaluable coding and research skills that have been foundational to my work. I am also grateful to my reading and examining committee members, Professors Yinyu Ye and Stephen Boyd, for their time and insightful comments.

I have a great deal of thanks to my collaborators: Professor Alice Cortinovis for our collaboration on the analysis of randomized trace estimators; and Wenzhi Gao for our work on online scaled gradient methods (`OSGM`) and the follow-up work on hypergradient descent methods (`HDM`) presented in Chapter 3. Collaborating with my co-authors has been an incredibly enjoyable and fruitful experience. I deeply appreciate the opportunity to work alongside them and have learned a great deal from these projects.

I would also like to acknowledge the Department of Mathematics at Stanford for providing funding, teaching and research assistantships, travel funds, and the academic flexibility to pursue my varied interests. A special thank you to Gretchen for her help with all the administrative intricacies of my PhD. Her presence and expertise provided immense relief and support throughout my time here.

To my labmates: thank you. I am grateful to Zach and Pratik for our discussions on randomized numerical linear algebra in optimization and machine learning; to Mike for his mentorship and industry advice; and to Alkis for our collaboration on the hard-constraint neural solvers in Chapter 4. I also want to thank Ali, Wanyu, Connor, Yinjun, and Angikar for their excellent talks during group meetings and for their constructive feedback on my own work. Additionally, thank you to everyone who has rotated through the Udell group. The presentations were a highlight of my PhD journey and a constant source of learning.

I am deeply grateful to the friends I made at Stanford. To Yingxi, Yi-Ting, and Boyang: thank you for listening to my PhD struggles, for encouraging me during difficult stretches, and for all the fun times that so richly enriched my life outside the lab. I also want to thank my Math PhD cohort for the camaraderie during our early qualifying exam preparations and the memorable times we shared.

Special thanks to Professor Ruey-Lin Sheu for his mentorship; to Ssu-Hsien for her unwavering support; and to many other friends in Taiwan who were always there when I needed someone to talk to.

I deeply appreciate Uncle Henry and Auntie Diane for teaching me everything I needed to know about living in the US, and for their warm invitations to their home for holidays and events. Their local support has been an incredibly important anchor for me during my doctoral journey.

Finally, I would like to thank my family. To my parents and Cheng's family: thank you for your unconditional love and support through the highs and lows of this PhD. You gave me the strength to overcome the challenges along this journey, and I share this milestone with you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Optimization algorithms have evolved continuously from World War II operations research to modern machine learning. Building on classic foundations like gradient descent and Newton's method, the simplex method was introduced in the 1940s for linear programming. By the 1980s and 1990s, interior-point methods (`IPMs`) were developed and generalized to handle convex and conic programming. As problem sizes surged in the 2000s, operator splitting methods emerged to manage increasingly complex constraints. Over the last decade, massive machine learning models have renewed the focus on stochastic first-order methods like `Adam` and quasi-Newton methods like `L-BFGS`. Most recently, *neural solvers* have emerged as fast surrogates: neural networks are trained to solve parameterized problem families via a single rapid forward pass, which is typically faster than rerunning an iterative optimization algorithm from scratch or even warm-starting it.

Preconditioning is a powerful technique to accelerate the convergence of optimization algorithms both in theory and practice. The convergence of first order optimization methods, as well as iterative linear system solvers that arise as subproblems within second order methods, is highly sensitive to the imbalance scaling across different directions of the optimization landscape. Such imbalance is formally captured by the *condition number*, the ratio of the largest to the smallest eigenvalue of the objective's Hessian. When the condition number is large, first order methods are prone to zig-zagging along narrow valleys and exhibit slow, empirically stalling progress. Preconditioning addresses this problem by applying a linear transformation to the gradient or to the linear system that reduces the effective condition number of the problem, thereby enabling faster convergence. A well-chosen preconditioner approximates the local curvature of the objective at low computational cost. Designing such preconditioners is an active area of research in optimization. **This thesis introduces new adaptive preconditioning techniques for three distinct algorithmic families: interior-point methods, gradient methods, and neural solvers.**

`IPMs` use both the gradient and Hessian to account for local curvature and yield significantly lower iteration complexity. However, the primary bottleneck for `IPMs` is the computational overhead per

iteration. Computing or approximating the inverse of the Hessian, even implicitly, scales poorly as the number of decision variables grows, rendering IPMs impractical for modern large-scale problems. The first part of the thesis presents a new algorithm for convex separable quadratic programming (QP) called Nys-IP-PMM, a regularized interior-point solver that uses low-rank structure to accelerate solution of the Newton system. The algorithm combines the interior point proximal method of multipliers (IP-PMM) with the randomized Nyström preconditioned conjugate gradient method as the inner linear system solver. Nys-IP-PMM is matrix-free: it accesses the input matrices solely through matrix-vector products, as opposed to methods involving matrix factorization. It works particularly well for separable QP instances with dense constraint matrices. This thesis establishes the convergence of Nys-IP-PMM and demonstrates its performance through numerical experiments.

First order methods rely solely on gradient information to move toward the minimizer(s) and have much lower per-iteration computational cost compared to IPMs. They are highly scalable and more practical than second order methods like IPMs when the problem size grows large, since the iterative linear system solve in each iteration of IPMs becomes less stable and often fails to converge to the required accuracy. However, the convergence of first order methods is notoriously sensitive to the conditioning of the problem and often stalls on ill-conditioned problems.

The second chapter of this thesis builds on our work in [Gao et al., 2024] to develop an analysis for hypergradient descent. In [Gao et al., 2024], we develop an online learning framework to accelerate the convergence of gradient methods. The induced family of algorithms are called online scaled gradient methods (OSGM). OSGM learns the preconditioners through an online learning algorithm and this adaptive preconditioning strategy provably accelerates gradient methods asymptotically. OSGM framework is applied to analyze the hypergradient descent method (HDM), a 25-year-old heuristic originally proposed for adaptive stepsize selection in stochastic first order methods [Almeida et al., 1999, Baydin et al., 2018]. We prove the convergence properties of HDM and show that HDM automatically identifies the optimal stepsize for the local optimization landscape and achieves local superlinear convergence. Our analysis explains the instability of HDM reported in the literature and proposes efficient strategies to address it. Experiments on deterministic convex problems show OSGM with heavy-ball momentum exhibits robust performance and significantly outperforms other adaptive first order methods. Moreover, OSGM often matches the performance of L-BFGS using less memory and cheaper iterations.

The last part of the thesis focuses on the constraint enforcement problem for neural solvers. Neural solvers are neural networks trained on a family of optimization problems that share the same structure but differ in their data. They have emerged as a new paradigm that exploits known problem structure to achieve faster solutions. However, a major challenge for neural solvers is to ensure the feasibility of the solutions (i.e., output from neural network). We propose SnareNet, a feasibility enforcement layer which can be appended to any neural network architecture to ensure that the output satisfies input-dependent nonlinear constraints. SnareNet steers the output of the neural

network toward feasibility by navigating in the constraint map's range space. The user can specify a tolerance level for constraint satisfaction. To stabilize end-to-end training, `SnareNet` uses *adaptive relaxation*, which designs a relaxed feasible set that snares the neural network at initialization and shrinks it into the feasible set. This training strategy enables early exploration and enforces strict feasibility later in training. On optimization-learning and neural policy benchmarks, `SnareNet` consistently attains improved objective quality while satisfying constraints more reliably than prior work.

In summary, this thesis contributes to the preconditioning literature by introducing new adaptive preconditioning techniques that accelerate the convergence of optimization algorithms across second order methods, first order methods, and neural solvers.

Chapter 2

Randomized Preconditioners in Interior Point Method

Solving a large-scale quadratic optimization problem to high precision, such as 10^{-6} to 10^{-8} relative accuracy, poses a considerable challenge. First-order methods scale and parallelize well but converge so slowly that accuracy below around 10^{-4} is generally not achievable. Conversely, interior point methods might become expensive if nontrivial sparsity patterns cause large fill-in in Cholesky factors.

Matrix-free interior point solvers present a solution to this conundrum: these solvers access the original input only through matrix-vector products (*matvecs*) [Gondzio, 2012a], and so can benefit from algorithmic advances and hardware for fast matrix-vector multiplication. For example, an $n \times n$ discrete Fourier transform (DFT) matrix can be applied to a vector in $O(n \log n)$ time [Golub and Van Loan, 2013, sect. 1.4]; and for general dense matrices, hardware accelerators such as GPUs enable fast matvecs.

In this chapter, we aim to solve a primal-dual pair of linearly constrained separable convex quadratic programs (QPs):

$$\begin{array}{lllll} \text{(P):} & \underset{x}{\text{minimize}} & \frac{1}{2}x^T Q x + c^T x & \text{(D):} & \underset{x,y,z}{\text{maximize}} \quad b^T y - \frac{1}{2}x^T Q x \\ & \text{subject to} & Ax = b, & & \text{subject to} \quad A^T y + z = Qx + c, \\ & & x \geq 0; & & y\text{: free, } z \geq 0, \end{array}$$

where $x \in \mathbb{R}^n$ is the primal variable, $y \in \mathbb{R}^m$ and $z \in \mathbb{R}^n$ are the dual variables, $Q \in \mathbb{R}^{n \times n}$ is positive semi-definite diagonal matrix, $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$. We assume Q is diagonal throughout this paper, i.e., problem (P) is separable[1].

[1] A non-diagonal Q can be transformed into a diagonal form through a change of variables, but one factorization of Q and this transformation may destroy sparsity in the problem. Let $Q = U \Lambda U^T$ be an eigendecomposition of Q. Define $x' = U^T x$. Then the diagonalized objective is $\frac{1}{2}x'^T \Lambda x' + c^T x'$ with constraints $AUx' = b$, $Ux' \geq 0$. Moreover, the inequality constraints $Ux' \geq 0$ can be reformulated as linear constraints by introducing slack variables. Note that

This problem template includes many real-world problems, ranging from robotics to aeronautics to finance. For example, many control problems, ranging from robotics to aeronautics to finance, model the control effort to be minimized as the sum of squares of control inputs (a diagonal quadratic) subject to given initial and final states and to linear dynamics (linear constraints). Appendix A.1 provides a concrete example of separable QP with matrix-free constraints.

Contributions. This chapter investigates variants of a regularized interior point framework, the interior point proximal method of multipliers (`IP-PMM`) [Pougkakiotis and Gondzio, 2021], for solving separable QPs. These variants use iterative methods to solve the Newton systems that arise as `IP-PMM` subproblems. Regularization is critical for matrix-free methods, which must rely on iterative linear system solvers rather than (generally more stable) direct methods. We propose to use the randomized Nyström preconditioner [Frangella et al., 2023b] to accelerate the iterative solve, and call the resulting algorithm `Nys-IP-PMM`. We show that `Nys-IP-PMM` enjoys both numerical stability and faster convergence than other variants of `IP-PMM`. `Nys-IP-PMM` solves the Newton system inexactly at each IPM iteration, so it is also an inexact variant of matrix-free `IP-PMM`. We prove that, for any $\epsilon \in (0, 1)$, inexact `IP-PMM` achieves duality measure $\mu_k \leq \epsilon$ after $k = O(n^4 \log \frac{1}{\epsilon})$ iterations, provided the error in the search direction decreases in the order of duality measure μ_k (see Theorem 2.7). This result allows us to establish probabilistic convergence results for `Nys-IP-PMM` (see Theorem 2.12). In our experiments, we compare the randomized Nyström preconditioner with a more standard choice of preconditioner, the partial Cholesky preconditioner [Gondzio, 2012a, Bellavia et al., 2013, Morini, 2018], to assess which one improves the performance of `IP-PMM` the most. We demonstrate that the randomized Nyström preconditioner generally improves the condition number of the normal equations compared to the partial Cholesky preconditioner. The results also demonstrate a significant speed-up in terms of wallclock time when using `Nys-IP-PMM` compared with using partial Cholesky as the preconditioner in `IP-PMM`, called Chol-IP-PMM. We provide a publicly available implementation of `IP-PMM` in Julia that incorporates both preconditioners and additional industry-standard heuristics such as Mehrotra's initial point and predictor-corrector method. This implementation is the first matrix-free regularized interior point method that is open source and freely available to use or modify: see `https://github.com/udellgroup/Nys-IP-PMM`.

Comparison to other variants of `IP-PMM`. Table 2.1 summarizes the differences between previous work on `IP-PMM` and our contributions. The first paper to propose `IP-PMM` [Pougkakiotis and Gondzio, 2021] considers QP with exact search direction (i.e., using a direct linear system solver), providing polynomial complexity results and numerical experiments. Numerical performance of inexact `IP-PMM` on QP has been explored in experimental papers [Bergamaschi et al., 2021, Gondzio et al., 2022] by the same authors along with other researchers. These papers introduce novel preconditioners for the iterative solvers used by inexact `IP-PMM`, but they use entrywise access to Q

this change of variables needs entrywise access of Q and increases the size of constraint matrix to $(m + n) \times 2n$.

and A, and so are not matrix-free. However, no theoretical convergence result of inexact `IP-PMM` on QP is presented. The first theoretical convergence analysis of inexact `IP-PMM` is established in [Pougkakiotis and Gondzio, 2022], but on linearly constrained semidefinite programming (SDP), without implementation. In our work, we adapt the convergence proof of inexact `IP-PMM` for SDP [Pougkakiotis and Gondzio, 2022] as a theoretical basis for inexact `IP-PMM` algorithm for solving QP. We propose the randomized Nyström PCG, an iterative matrix-free solver, within inexact `IP-PMM` and establish probabilistic convergence results, along with numerical experiments to show the efficacy of the method in practice.

	Problem	Search direction	Convergence proof	Matrix-free preconditioning	Numerical results
Pougkakiotis and Gondzio [2021]	QP	Exact	✓	-	✓
Bergamaschi et al. [2021], Gondzio et al. [2022]	QP	Inexact	✗	✗	✓
Pougkakiotis and Gondzio [2022]	SDP	Inexact	✓	✗	✗
`Nys-IP-PMM` (ours)	QP	Inexact	✓	✓	✓

Table 2.1: Features for variants of `IP-PMM` for convex QP / SDP

Organization. Section 2.1 contains an overview of `IP-PMM`, randomized Nyström preconditioner, and partial Cholesky preconditioner. Section 2.2 details convergence results for inexact `IP-PMM` on QP. Section 2.3 introduces our main algorithm, `Nys-IP-PMM`, proves convergence of the algorithm, and provides implementation details. Section 2.4 demonstrates the numerical performance of `Nys-IP-PMM`.

Notation. For a vector $x \in \mathbb{R}^n$, subindex $x_i \in \mathbb{R}$ denotes the i-th component of x, and superindex $x^k \in \mathbb{R}^n$ denotes the k-th iterate. For scalar/matrix, we use subscript k for the iteration count, as usual. Given a set of indices $\mathcal{S} \subset [n] \coloneqq \{1, 2, \ldots, n\}$ and an arbitrary vector $x \in \mathbb{R}^n$, let $x_{\mathcal{S}}$ denote the sub-vector containing the elements of x whose indices belong to $\mathcal{S}$. The diagonal matrix with main diagonal x is denoted by $\text{diag}(x)$. Given a matrix M, we use $\lambda_i(M)$ to denote the i-th largest eigenvalue of M and use $M^\dagger$ to denote the pseudoinverse of M. For a positive definite M, we write the M-norm $\|u\|_M^2 = u^T M u$. The set of $m \times m$ real symmetric positive semidefinite matrices is denoted by $\mathbb{S}_+^m(\mathbb{R})$. The n-by-n identity matrix is denoted by I_n and $\mathbb{1}_n = (1, \ldots, 1)^T \in \mathbb{R}^n$. The cost of computing a matrix-vector product with given matrix M is denoted by T_M.

2.1 Background: IPMs and Conjugate Gradient

This section provides background to develop our main algorithm, `Nys-IP-PMM`, a matrix-free variant of the interior point proximal method of multipliers (`IP-PMM`) [Pougkakiotis and Gondzio, 2021] for solving separable QPs.

2.1.1 Interior-point methods (IPMs)

IPMs are among the most important methods, both in theory and in practice, for solving convex QPs. IPMs were pioneered by the landmark paper of Karmarkar [Karmarkar, 1984] in 1984 on linear programming (LP), with a flurry of activity improving these initial results both in theory [Megiddo, 1989, Kojima et al., 1989, 1993] and in practice [Marsten et al., 1990, Mehrotra, 1992, Lustig et al., 1992, 1994]. One key reason for the success of the IPM approach is the use of the logarithmic barrier function [Gill et al., 1986], a nonlinear programming technique. Later, IPMs were extended to linearly constrained convex QP [Kapoor and Vaidya, 1986, Ye and Tse, 1989, Mehrotra and Sun, 1990, Goldfarb and Liu, 1990, Monteiro and Adler, 1989], semidefinite programming (SDP) [Vandenberghe and Boyd, 1996], and more generally conic optimization problems [Nesterov and Nemirovskii, 1994, Forsgren et al., 2002, Lobo et al., 1998]. For a more complete history of IPMs, see the review paper [Gondzio, 2012b].

Regularized IPMs

In primal–dual IPMs for (P) and (D) with diagonal Q, the majority of the computation is devoted to solving a symmetric positive semi-definite linear system to determine the search direction, $(\Delta x^k, \Delta y^k, \Delta z^k)$, for each iteration k. At each iteration, the algorithm forms a new diagonal matrix Θ_k and right-hand side (RHS) vector ξ^k and must solve the *normal equations*

$$A(Q + \Theta_k^{-1})^{-1} A^T \Delta y^k = \xi^k. \tag{2.1}$$

The system (2.1), whether solved by direct or iterative methods, is unstable when A is nearly rank-deficient or when $Q + \Theta_k^{-1}$ is nearly singular. *Regularized IPMs* [Saunders, 1996, Saunders and Tomlin, 1996, Altman and Gondzio, 1999, Friedlander and Orban, 2012] improve stability by regularizing the primal problem (P) and/or dual problem (D). Compared to standard IPMs, regularized IPMs generally require more iterations (linear system solves), but the problem to solve at each iteration is more stable [Pougkakiotis and Gondzio, 2021]. Regularization is particularly important to matrix-free methods, which must rely on iterative linear system solvers rather than (more stable) direct methods.

Interior point proximal method of multipliers (`IP-PMM`)

`IP-PMM` is the first provably polynomial-time primal-dual regularized IPM, proposed by Pougkakiotis and Gondzio [Pougkakiotis and Gondzio, 2021] in 2021. At each `IP-PMM` iteration, the algorithm determines the search direction by applying a *single* IPM iteration to the PMM subproblem associated with the primal-dual pair (P)–(D) [Rockafellar, 1976]:

$$\underset{x \geq 0}{\text{minimize}} \quad \frac{1}{2} x^T Q x + c^T x + (\lambda^k)^T (b - Ax) + \frac{1}{2\delta_k} \|Ax - b\|_2^2 + \frac{\rho_k}{2} \|x - \zeta^k\|^2. \tag{2.2}$$

These PMM subproblems are parametrized by estimates ζ^k and λ^k of the primal variable x and Lagrange multiplier y respectively, and parameters $\rho_k > 0$ and $\delta_k > 0$ controlling the strength of the regularization. The linear and quadratic terms $(\lambda^k)^T(b - Ax) + \frac{1}{2\delta_k}\|Ax - b\|_2^2$ are motivated by the *method of multipliers* (also called the *augmented Lagrangian method*) [Hestenes, 1969, Powell, 1969], and ensure the dual objective is strongly concave. The *proximal* term $\frac{\rho_k}{2}\|x - \zeta^k\|^2$ provides strong convexity for primal problem and ensures better numerical behavior. When $\rho_k = \delta_k \to 0$ and the dual estimate is updated by gradient ascent $\lambda^{k+1} \leftarrow \lambda^k - \frac{1}{\delta_k}(Ax^{k+1} - b)$, the solution to the PMM subproblem (2.2) converges to the solution to (P) [Rockafellar, 1976]. Note that `IP-PMM` does not solve the PMM subproblem to any particular precision, but simply applies a single IPM iteration to (2.2) to obtain a search direction and update the iterates.

To apply an IPM iteration to (2.2) and find the search direction, a logarithmic barrier function is introduced to enforce the non-negativity constraint $x \geq 0$, so that the Lagrangian to minimize becomes

$$\mathcal{L}^{\texttt{IP-PMM}}_{\mu_k,\rho_k,\delta_k}(x;\zeta^k,\lambda^k) = \frac{1}{2}x^TQx + c^Tx + (\lambda^k)^T(b - Ax) + \frac{1}{2\delta_k}\|Ax - b\|_2^2 + \\ + \frac{\rho_k}{2}\|x - \zeta^k\|^2 - \mu_k\sum_{j=1}^{n}\ln x^j. \quad (2.3)$$

Introducing the new variables $y = \lambda^k - \frac{1}{\delta_k}(Ax - b)$ and $z = \mu_k \operatorname{diag}(x)^{-1}\mathbb{1}_n$, the first-order optimality conditions for minimizing (2.3) result in the nonlinear system

$$\begin{bmatrix} c + Qx - A^Ty - z + \rho_k(x - \zeta^k) \\ Ax + \delta_k(y - \lambda^k) - b \\ \operatorname{diag}(x)z - \mu_k\mathbb{1}_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \quad (2.4)$$

`IP-PMM` applies a variation of Newton's method to solve a perturbed form of (2.4). Specifically, it linearizes the optimality conditions and perturbs the right-hand side (RHS) (see Section 2.2.1 for details and Equation (2.19) for definitions of $r_d^k \in \mathbb{R}^n$, $r_p^k \in \mathbb{R}^m$, and $r_\mu^k \in \mathbb{R}^n$):

$$\begin{bmatrix} -(Q + \rho_k I_n) & A^T & I_n \\ A & \delta_k I_m & 0 \\ \operatorname{diag}(z^k) & 0 & \operatorname{diag}(x^k) \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = \begin{bmatrix} r_d^k \\ r_p^k \\ r_\mu^k \end{bmatrix}. \quad (2.5)$$

When Q is diagonal, by eliminating Δx^k and Δz^k using the first and third block of equations in (2.5), the Newton system (2.5) can be reduced to solving (2.6) and computing Δx^k and Δz^k via

(2.7)–(2.8)

$$(N_k + \delta_k I_m)\Delta y^k = \xi^k; \tag{2.6}$$

$$\Delta x^k = (Q + \Theta_k^{-1} + \rho_k I_n)^{-1}(A^T \Delta y^k - r_d^k + \text{diag}(x^k)^{-1} r_\mu^k); \tag{2.7}$$

$$\Delta z^k = \text{diag}(x^k)^{-1}(r_\mu^k - \text{diag}(z^k)\Delta x^k), \tag{2.8}$$

where $N_k \coloneqq A(Q + \Theta_k^{-1} + \rho_k I)^{-1} A^T$, $\Theta_k = \text{diag}(x^k)\,\text{diag}(z^k)^{-1}$ is diagonal, and

$$\xi^k = r_p^k + A(Q + \Theta_k^{-1} + \rho_k I_n)^{-1}(r_d^k - \text{diag}(x^k)^{-1} r_\mu^k).$$

Equation (2.6) is the regularized version of normal equations (2.1) in original IPMs and is solved to determine the search direction of `IP-PMM`.

The polynomial complexity results of `IP-PMM` are built upon the choice $\rho_k = \delta_k = \mu_k$ and the barrier parameter μ_k is chosen to be the average complementarity product $\mu_k := \frac{x_k^T z_k}{n}$, as in standard IPMs. When primal and dual feasibility is approached, μ_k becomes a measure of duality gap. The convergence theory for `IP-PMM` requires μ_k to control both the IPM and PMM. Under this choice and standard assumptions, Pougkakiotis and Gondzio [Pougkakiotis and Gondzio, 2021, Thm. 3] show that every limit point of $\{(x_k, y_k, z_k) : k \in \mathbb{N}\}$ generated by `IP-PMM` determines a primal-dual solution of pair (P)–(D). They show linear convergence of the method: for any given tolerance error $\epsilon \in (0, 1)$, `IP-PMM` produces a sequence of iterates $\{(x_k, y_k, z_k)\}_{k\in\mathbb{N}}$ such that $\mu_k \leq \epsilon$ after $O(n^4 \log \frac{1}{\epsilon})$ iterations [Pougkakiotis and Gondzio, 2021, Thm. 2]. Their results assume the Newton system (2.5) is solved exactly. While this assumption is reasonable when a direct method solves (2.6), it becomes untenable when using an iterative method such as CG. We prove convergence for *inexact* `IP-PMM` for QP in Section 2.2. Note that the resulting algorithm, presented as Algorithm 3 has two loops: the outer loop of `IP-PMM` and the inner loop of the iterative solver for the normal equations (2.6).

2.1.2 Preconditioned conjugate gradient

The conjugate gradient (CG) method is the most common iterative method to solve (2.6). However, even with the extra stability conferred by `IP-PMM`, CG usually *stalls*, i.e. fails to converge to sufficient accuracy, especially during the final iterations of `IP-PMM`. Preconditioning is used to improve convergence of CG [Golub and Van Loan, 2013, sect. 11.5] by choosing a positive definite matrix P, called a *preconditioner*, replacing the system to be solved with an equivalent symmetric positive definite system:

$$P^{-1/2}(N_k + \delta_k I_m)P^{-1/2} v^k = P^{-1/2}\xi^k$$

with new variable $v^k = P^{1/2}\Delta y^k$. The resulting algorithm is called *preconditioned conjugate gradient* (PCG), and its update relies on the matrix-vector product $v \mapsto P^{-1}v$. Therefore, a good preconditioner has two properties: 1) its inverse P^{-1} must be computationally cheap to apply, and 2) the

preconditioned system $P^{-1/2}(N_k + \delta_k I_m)P^{-1/2}$ must have a spectrum that allows CG to converge faster than on the original system $N_k + \delta_k I_m$. For example, we may seek a preconditioner so that the preconditioned system has a reduced condition number [Johnson et al., 1983, Chan, 1988, de Prenter et al., 2017].

Randomized IPMs for linear programming

Many preconditioners in the IPM literature require entrywise access to the matrices Q and A [Bergamaschi et al., 2021, Gondzio et al., 2022, Chowdhury et al., 2022, Oliveira and Sorensen, 2005, Bocanegra et al., 2007, Durazzi and Ruggiero, 2003, Schork and Gondzio, 2020, Casacio et al., 2017]. Among these, the randomized preconditioner by Chowdhury et al. [Chowdhury et al., 2022] is most related to our work. Chowdhury et al. [Chowdhury et al., 2022] pioneered the use of randomized preconditioners in non-regularized IPMs for LPs (i.e., $Q = 0$) with wide constraint matrix $m \ll n$. The same randomized preconditioner is later used to accelerate the predictor-corrector IPM for LPs in a follow-up work [Dexter et al., 2022]. In their works, the normal equations take the form $A\Theta_k A^T \Delta y^k = \xi^k$. They propose the preconditioner $P = A\Theta_k^{-1/2}\Omega\Omega^T\Theta_k^{-1/2}A^T$, where $\Omega \in \mathbb{R}^{n\times\ell}$ is a random sketching matrix such that P approximates $A\Theta_k A^T$ well in spectral norm with probability at least $1 - \eta$ [Chowdhury et al., 2022, Lemma 2]. The sketch size ℓ is of order $\ell = O(m\log(m/\eta))$ and matrix Ω has $O(\log(m/\eta))$ non-zero entries. As a result, the total computational cost for P^{-1} is $O((\text{nnz}(A) + m^3)\log(m/\eta))$. The method suffers cubic complexity in the number of constraints m. In contrast, the Nyström preconditioner can be constructed in $O(\ell\,\text{nnz}(A) + \ell^3)$ and thus offers the greatest advantage over [Chowdhury et al., 2022] when $m \approx n$ and/or m is much larger than the rank ℓ required for a good approximation.

Partial Cholesky preconditioners

A matrix-free preconditioner offers improved scalability, particularly when entrywise access to A is expensive. Gondzio [Gondzio, 2012a] propose a matrix-free preconditioner in the context of (regularized) IPMs, called the *partial Cholesky preconditioner*, which is elaborated in Bellavia et al. [Bellavia et al., 2013] and Morini [Morini, 2018]. Given a target rank $\ell \ll m$, partial Cholesky forms the greedily pivoted Cholesky factorization $LL^T = E(N_k + \delta_k I)E^T$ with appropriate permutation matrix $E \in \mathbb{R}^{m\times m}$ and takes the first ℓ columns of L, $L_\ell \in \mathbb{R}^{m\times\ell}$, which corresponds to a rank-ℓ approximation $L_\ell L_\ell^T \approx N_k + \delta_k I$, together with a diagonal matrix that ensures the approximation is positive definite. The key limitation of the partial Cholesky preconditioner lies in its construction, which relies on the entire diagonal of N_k. In a matrix-free setting without entrywise access to A, computing the diagonal of N_k requires at least m matvecs with A^T. In practice, this feature slows down the partial Cholesky preconditioner considerably, as these additional m matvecs are performed at every iteration.

We describe the partial Cholesky preconditioner in more detail here for completeness, since it

is a standard choice for a matrix-free IPM implementation. We omit the iteration count k when it is clear from context. The first ℓ columns of the Cholesky factor, $L_\ell \coloneqq \begin{bmatrix} L_{11} \\ L_{21} \end{bmatrix}$, yield the following factorization in block form:

$$E(N+\delta I)E^T \coloneqq \begin{bmatrix} N_{\delta,11} & N_{\delta,21}^T \\ N_{\delta,21} & N_{\delta,22} \end{bmatrix} = \begin{bmatrix} L_{11} & \mathbf{0} \\ L_{21} & I_{m-\ell} \end{bmatrix} \begin{bmatrix} I_\ell & \mathbf{0} \\ \mathbf{0} & S \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ \mathbf{0} & I_{m-\ell} \end{bmatrix}, \tag{2.9}$$

where $N_{\delta,11} \in \mathbb{R}^{\ell\times\ell}$ is the leading principal submatrix containing the ℓ largest diagonal elements and $S = N_{\delta,22} - N_{\delta,21}N_{\delta,11}^{-1}N_{\delta,21}^T$ is the Schur complement of $N_{\delta,11}$. In practice, S is never explicitly formed when performing the Cholesky factorization and hence is not available. The partial Cholesky preconditioner approximates S by its diagonal and takes the form

$$P_{\text{Chol}} = E^T \begin{bmatrix} L_{11} & \mathbf{0} \\ L_{21} & I_{m-\ell} \end{bmatrix} \begin{bmatrix} I_\ell & \mathbf{0} \\ \mathbf{0} & \operatorname{diag}(S) \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ \mathbf{0} & I_{m-\ell} \end{bmatrix} E, \tag{2.10}$$

the inverse of which has the closed-form formula

$$P_{\text{Chol}}^{-1} = E^T \begin{bmatrix} L_{11}^{-T} & -L_{11}^{-T}L_{21}^T \\ \mathbf{0} & I_{m-\ell} \end{bmatrix} \begin{bmatrix} I_\ell & \mathbf{0} \\ \mathbf{0} & \operatorname{diag}(S)^{-1} \end{bmatrix} \begin{bmatrix} L_{11}^{-1} & \mathbf{0} \\ -L_{21}L_{11}^{-1} & I_{m-\ell} \end{bmatrix} E. \tag{2.11}$$

Pseudocode for the construction of this preconditioner appears in [Bellavia et al., 2013, Alg. 1] and [Morini, 2018, Alg. 1–2], while spectral analysis for the preconditioned system is developed in [Bellavia et al., 2013, Gondzio, 2012a]. In our comparison, we construct the partial Cholesky preconditioner by [Morini, 2018, Alg. 2], which yields a more efficient matvec with P_{Chol}^{-1} when the first ℓ columns of $E(N+\delta I)E^T$ are more sparse than L_{11}^{-1} and L_{21}; and a comparable cost of matvec otherwise [Morini, 2018, sect. 3.1].

Remark 2.1 (sparsity of L_{21}). In this paper, we consider both sparse and dense constraint matrices A. When A is dense, the Cholesky factors are in general dense. When the L_{21} block of the Cholesky factorization is sparse, the matvec with the partial Cholesky preconditioner can potentially exploit the sparsity of L_{21}. However, even when A is sparse, the L_{21} block of the Cholesky factorization may not be sparse. Sparsity of this block depends on the sparsity pattern of A and the pivots used. However, it is difficult to choose these pivots well without entrywise access to A. In the partial Cholesky literature, these pivots are chosen greedily based on the diagonal entries of N_k. In the problems we consider in our numerical experiments, the L_{21} block is always dense.

Constructing the partial Cholesky preconditioner requires access to the complete diagonal of $N+\delta I$ since we need entry-wise access of $N_{\delta,11}$ for factorization (2.9) and $\operatorname{diag}(N_{\delta,22})$ for computing $\operatorname{diag}(S)$. Using the form $N = A(Q+\Theta^{-1}+\rho I)^{-1}A^T$, the diagonal of $N+\delta I$ can be computed in

the following matrix-free manner:

$$r_i = (Q + \Theta^{-1} + \rho I)^{-\frac{1}{2}} A^T e_i, \quad (N + \delta I)_{ii} = r_i^T r_i + \delta, \quad \text{for all } i = 1, 2, \ldots, m.$$

This computation requires m matrix-vector products with A^T and componentwise scaling of a length n vector, which costs $O(m(T_A + n))$[2]. The diagonal of S can be similarly computed, as a cost $O(m(T_{N_{\delta,21}} + \ell^2))$. The total construction cost of partial Cholesky preconditioner is dominated by computing $\text{diag}(N + \delta I)$ and $\text{diag}(S)$, resulting in $O(m(T_A + n) + m(T_{N_{\delta,21}} + \ell^2))$ arithmetic operations. The cost of building $\text{diag}(N + \delta I)$ may be significantly reduced if access to rows of matrix A is available, because the matrix-vector products $A^T e_i$ can be trivially avoided [Gondzio, 2012a, Morini, 2018].

Randomized Nyström preconditioners

The randomized Nyström preconditioner is built upon the randomized Nyström low-rank approximation, which we introduce now. Let $N \in \mathbb{S}_+^m(\mathbb{R})$ be a real symmetric positive semidefinite matrix and $\Omega \in \mathbb{R}^{m\times\ell}$ be a random Gaussian test matrix (i.e., each entry is drawn i.i.d. from a normal distribution). The integer $\ell \ll m$ is called the *sketch size* and the matrix $Y = N\Omega \in \mathbb{R}^{m\times\ell}$ is called the *sketch* of N. We observe the sketch can be obtained by ℓ matrix-vector products with N. The *randomized Nyström approximation* with respect to the range of Ω is defined as

$$\hat{N} = (N\Omega)(\Omega^T N\Omega)^\dagger (N\Omega)^T = Y(\Omega^T Y)^\dagger Y^T. \tag{2.12}$$

Observe from (2.12) that this approximation can be constructed directly from the test matrix Ω and the sketch Y, without further access to N. The rank of $\hat{N}$ is equal to ℓ with probability 1, and hence the terms sketch size and rank are used interchangeably. The formula (2.12) is not numerically stable. Instead, our algorithm uses Algorithm 1 to construct a randomized rank-ℓ Nyström approximation. It provides a stable and efficient implementation with a computational cost of $O(T_N\ell + \ell^2 m)$, where T_N is the cost of a matvec with N. In practice, the thin SVD in line 7 of Algorithm 1 is the most computationally intensive operation. Moreover, Algorithm 1 returns the truncated eigendecomposition of the randomized Nyström approximation: $\hat{N} = \hat{U}\hat{\Lambda}\hat{U}^T$, where $\hat{U} \in \mathbb{R}^{m\times\ell}$ has orthonormal columns and $\hat{\Lambda} \in \mathbb{R}^{\ell\times\ell}$ is diagonal with diagonal entries $\hat{\lambda}_1 \geq \cdots \geq \hat{\lambda}_\ell$.

Given a rank-ℓ randomized Nyström approximation $\hat{N} = \hat{U}\hat{\Lambda}\hat{U}^T$ returned from Algorithm 1, the *randomized Nyström preconditioner* for regularized system (2.6) and its inverse take the form

$$P_{\text{Nys}} = \frac{1}{\hat{\lambda}_\ell + \delta}\hat{U}(\hat{\Lambda} + \delta I)\hat{U}^T + (I - \hat{U}\hat{U}^T), \tag{2.13}$$

$$P_{\text{Nys}}^{-1} = (\hat{\lambda}_\ell + \delta)\hat{U}(\hat{\Lambda} + \delta I)^{-1}\hat{U}^T + (I - \hat{U}\hat{U}^T). \tag{2.14}$$

[2]This cost can be reduced to $\mathcal{O}(\text{nnz}(A))$ if entrywise access to A is available, since $A^T e_i$ is the i-th row of A.

Algorithm 1 Randomized Nyström Approximation [Martinsson and Tropp, 2020, Algorithm 16]

Input: Matrix-vector product oracle of an $m \times m$ positive semidefinite matrix N, target rank $\ell \ll m$
Output: Randomized Nyström approximation $\hat{N} = \hat{U}\hat{\Lambda}\hat{U}^T$

1: Draw a Gaussian test matrix $\Omega \in \mathbb{R}^{m\times\ell}$ ▷ $\mathcal{O}(\ell m)$
2: $Y = N\Omega$ ▷ $\mathcal{O}(\ell T_N)$
3: $\nu = \text{eps}(\text{norm}(Y, \text{'fro'}))$ ▷ $\mathcal{O}(\ell m)$
4: $Y_\nu = Y + \nu\Omega$ ▷ $\mathcal{O}(\ell m)$
5: $C = \text{chol}\left(\Omega^T Y_\nu\right)$ ▷ $\mathcal{O}(\ell^2 m)$
6: $B = Y_\nu C^{-1}$ ▷ $\mathcal{O}(\ell^2 m)$
7: $[\hat{U}, \Sigma, \sim] = \text{svd}(B)$ ▷ $\mathcal{O}(\ell^2 m)$
8: $\hat{\Lambda} = \max\left\{0, \Sigma^2 - \nu I\right\}$ ▷ $\mathcal{O}(\ell)$

Algorithm 2 Nyström Preconditioner [Frangella et al., 2023b]

Input: Positive semidefinite N_k, sketch size ℓ_k, regularization parameter δ_k
Output: Inverse of randomized Nyström preconditioner P_k^{-1} as a linear operator

1: Compute randomized Nyström rank-ℓ_k approximation by Algorithm 1:

$$\hat{N}_k = \hat{U}\hat{\Lambda}\hat{U}^T$$

2: Construct inverse Nyström preconditioner P_k^{-1} as in (2.14) with $\hat{U}$, $\hat{\Lambda}$, and δ_k.

It is important to highlight that the m^2 matrix elements of the randomized Nyström preconditioner are not explicitly formed in practice. Instead, P_{Nys} is viewed as a linear operator defined by $\hat{U}$, $\hat{\Lambda}$, and δ. Algorithm 2 summarizes the procedure.

In other words, the randomized Nyström preconditioner is directly available once the rank-ℓ randomized Nyström approximation has been constructed, and thus shares the same construction cost $O(T_N\ell + \ell^2 m)$ as the Nyström approximation. Both the Nyström preconditioner and its inverse are cheap to apply: a matvec with either requires $O(m\ell)$ arithmetic operations, dominated by the cost of applying $\hat{U}$ and $\hat{U}^T$ to a vector. The required storage for the randomized Nyström preconditioner is $O(m\ell)$. All these properties offer potential benefits compared to partial Cholesky, particularly when the target rank ℓ is much smaller than m; see Table 2.2. Our numerical experiments in Section 2.4 further demonstrate the advantages of the Nyström preconditioner in terms of both performance and computational efficiency in large-scale (dense) problems.

The randomized Nyström preconditioner can effectively accelerate the convergence of CG on linear systems arising from data-driven models [Frangella et al., 2023b, Zhao et al., 2022]. The reason is that most data matrices usually have rapidly decaying spectra and hence have smaller *effective dimension* [Udell and Townsend, 2019, Frangella et al., 2023b, Zhao et al., 2022, Lacotte and Pilanci, 2020]. Given a positive semidefinite matrix $N \in \mathbb{S}_+^m(\mathbb{R})$ and a regularization parameter

Preconditioner	Construction Cost	Matvec Cost	Storage
Randomized Nyström	$\mathcal{O}(\ell(T_A+n)+\ell^2 m)$	$(2\ell+1)m+5\ell$	$\mathcal{O}(\ell m+\ell)$
Partial Cholesky	$\mathcal{O}(m(T_A+n)+m(T_{N_{\delta,21}}+\ell^2))$ (no entrywise access to A) $\mathcal{O}(\text{nnz}(A)+m(T_{N_{\delta,21}}+\ell^2))$ (has entrywise access to A)	$(2\ell+1)(m-\ell)+2\ell^2$	$\mathcal{O}(\ell m+\ell^2+m)$

Table 2.2: Comparison of randomized Nyström preconditioner and the partial Cholesky preconditioner. The construction cost assumes the matrix of the system takes the form $A(Q+\Theta_k+\rho_k I)^{-1}A^T+\delta_k I$.

$\delta>0$, the *effective dimension* of N is defined as

$$d_{\text{eff}}(N,\delta)=\text{tr}(N(N+\delta I)^{-1})=\sum_{i=1}^{m}\frac{\lambda_i}{\lambda_i+\delta}, \tag{2.15}$$

where λ_i's are eigenvalues of N. The ratio $\lambda_i/(\lambda_i+\delta)$ is close to 1 if $\lambda_i \gg \delta$; and is close to 0 if $\lambda_i \ll \delta$. As a result, the effective dimension can be understood as a smoothed count of the eigenvalues of N that are greater than or equal to δ. Zhao et al. [Zhao et al., 2022] show that if the sketch size $\ell = O(d_{\text{eff}}(N,\delta)+\log(\frac{1}{\eta}))$, then with probability at least $1-\eta$, the condition number of the Nyström preconditioned system is bounded by a constant [Zhao et al., 2022, Thm. 4.1], and hence the Nyström preconditioned CG (NysPCG) can achieve ϵ-relative error in $O\left(\log(\frac{1}{\epsilon})\right)$ iterations, independent of the condition number (see Lemma 2.10 in appendix).

2.2 Inexact `IP-PMM` for convex QP

Pougkakiotis and Gondzio [Pougkakiotis and Gondzio, 2021] assume the search direction in `IP-PMM` satisfies the Newton system (2.5) exactly. This assumption is unrealistic when the inner linear system solver in `IP-PMM` is iterative. Here, we prove convergence for `IP-PMM` with errors in the search direction, which we call *inexact `IP-PMM`* (see Algorithm 3). Inexact `IP-PMM` on QP is exactly the same as [Pougkakiotis and Gondzio, 2021, Alg. `IP-PMM`], except that in line 7 of Algorithm 3, the inexact version we analyze satisfies the inexact Newton system (see Equation (2.19) for explicit formula) instead of the exact one. We adopt the assumption $\rho_k=\delta_k=\mu_k$ [Pougkakiotis and Gondzio, 2021] throughout this section. Section 2.2.1 introduces some technical definitions, and Section 2.2.2 presents convergence results for inexact `IP-PMM`.

Algorithm 3 Inexact `IP-PMM` (adapted from [Pougkakiotis and Gondzio, 2021, Alg. `IP-PMM`])

Input: QP data A, Q, b, c as in (P), tol
Parameters: $0 < \sigma_{\min} \leq \sigma_{\max} \leq 0.5,\ C_N > 0,\ 0 < \gamma_{\mathcal{A}} < 1,\ 0 < \gamma_{\mu} < 1$
Starting point: Set as in (2.16)

Compute infeasibility $r_p^0 \leftarrow Ax^0 - b$ and $r_d^0 \leftarrow c + Qx^0 - A^T y^0 - z^0$.
for $(k = 0, 1, 2, \cdots)$ **do**
  if $\left(\|r_p^k\|_2 < \text{tol}\right) \wedge \left(\|r_d^k\|_2 < \text{tol}\right) \wedge (\mu_k < \text{tol})$ **then**
    return $\left(x^k, y^k, z^k\right)$
  else
    Choose $\sigma_k \in [\sigma_{\min}, \sigma_{\max}]$.
    Solve system (2.19) such that the residuals satisfy Assumption 2.2.
    Choose largest stepsize $\alpha_k \in (0, 1]$ such that $\mu_k(\alpha) \leq (1 - 0.01\alpha)\mu_k$ and

$$\begin{bmatrix} x^k + \alpha_k \Delta x^k \\ y^k + \alpha_k \Delta y^k \\ z^k + \alpha_k \Delta z^k \end{bmatrix} \in \mathcal{N}_{\mu_k(\alpha)}\left(\zeta^k, \lambda^k\right),$$

    where $\mu_k(\alpha) = \left(x^k + \alpha_k \Delta x^k\right)^T \left(z^k + \alpha_k \Delta z^k\right)/n$.
    Update the iterate

$$\begin{bmatrix} x^{k+1} \\ y^{k+1} \\ z^{k+1} \end{bmatrix} \leftarrow \begin{bmatrix} x^k + \alpha_k \Delta x^k \\ y^k + \alpha_k \Delta y^k \\ z^k + \alpha_k \Delta z^k \end{bmatrix} \text{ and } \mu_{k+1} \leftarrow \frac{(x^{k+1})^T z^{k+1}}{n}.$$

    Set $r_p^{k+1} \leftarrow Ax^{k+1} - b$ and $r_d^{k+1} \leftarrow c + Qx^{k+1} - A^T y^{k+1} - z^{k+1}$.
    Set $\tilde{r}_p \leftarrow r_p^{k+1} - \frac{\mu_{k+1}}{\mu_0}\bar{b}$ and $\tilde{r}_d \leftarrow r_d^{k+1} + \frac{\mu_{k+1}}{\mu_0}\bar{c}$.
    if $\left(\left(\|(\tilde{r}_p, \tilde{r}_d)\|_2 \leq C_N \frac{\mu_{k+1}}{\mu_0}\right) \wedge \left(\|(\tilde{r}_p, \tilde{r}_d)\|_{\mathcal{A}} \leq \gamma_{\mathcal{A}} \rho \frac{\mu_{k+1}}{\mu_0}\right)\right)$ **then**
      $(\zeta^{k+1}, \lambda^{k+1}) \leftarrow (x^{k+1}, y^{k+1})$
    else
      $(\zeta^{k+1}, \lambda^{k+1}) \leftarrow (\zeta^k, \lambda^k)$
    end if
  end if
  $k \leftarrow k + 1$
end for

2.2.1 Definitions for inexact `IP-PMM`

In this section, we provide a detail on inexact `IP-PMM`, including the choice of starting point, neighborhoods, and inexact Newton systems, following [Pougkakiotis and Gondzio, 2021]. Refer to [Pougkakiotis and Gondzio, 2021] for more details.

Starting point For the analysis, we consider starting point $(x^0, z^0) = \rho(\mathbb{1}_n, \mathbb{1}_n)$ for some $\rho > 0$ and y^0 being an arbitrary vector such that $\|y^0\|_\infty = O(1)$ (i.e., the absolute values of its entries are

independent of n and m). The initial primal and dual estimates are taken as $\zeta^0 = x^0$ and $\lambda^0 = y^0$, and the initial duality measure is denoted by $\mu_0 = \frac{(x^0)^T z^0}{n}$. In addition, there exist two vectors $\bar{b}$ and $\bar{c}$ such that

$$Ax^0 = b + \bar{b}, \;\; -Qx^0 + A^T y^0 + z^0 = c + \bar{c}. \tag{2.16}$$

Neighborhoods `IP-PMM` is a path-following method that requires each iterate of `IP-PMM` to lie within a specified neighborhood. The neighborhoods in `IP-PMM` depend on the parameters in the PMM subproblems (2.2), as well as the starting point, and are parametrized by $(\zeta^k, \lambda^k, \mu_k)$. `IP-PMM` uses a semi-norm that depends on the input matrices A and Q to measure primal-dual infeasibility:

$$\|(b,c)\|_{\mathcal{A}} := \min_{x,y,z} \left\{ \|(x,z)\|_2 : Ax = b, \; -Qx + A^T y + z = c \right\}.$$

This norm measures the minimum 2-norm of (x, z) among all primal-dual feasible tuples (x, y, z) and can be evaluated via QR factorization of A [Mizuno and Jarre, 1999, sect. 4]. Now, given the starting point (x^0, y^0, z^0) and vectors $(\bar{b}, \bar{c})$ in (2.16), penalty parameter μ_k, and primal-dual estimates ζ^k, λ^k, Pougkakiotis and Gondzio [Pougkakiotis and Gondzio, 2021] define the set

$$\tilde{\mathcal{C}}_{\mu_k}\left(\zeta^k, \lambda^k\right) := \left\{ \begin{array}{c} Ax + \mu_k\left(y - \lambda^k\right) = b + \frac{\mu_k}{\mu_0}\left(\bar{b} + \tilde{b}^k\right), \\ (x,y,z): \;\; -Qx + A^T y + z - \mu_k\left(x - \zeta^k\right) = c + \frac{\mu_k}{\mu_0}\left(\bar{c} + \tilde{c}^k\right), \\ \left\|\left(\tilde{b}^k, \tilde{c}^k\right)\right\|_2 \leq C_N, \;\; \left\|\left(\tilde{b}^k, \tilde{c}^k\right)\right\|_{\mathcal{A}} \leq \gamma_{\mathcal{A}}\rho \end{array} \right\}, \tag{2.17}$$

where $C_N > 0$ is a constant, $\gamma_{\mathcal{A}} \in (0,1)$, and $\rho > 0$ is defined as in the starting point. The vectors $\tilde{b}^k$ and $\tilde{c}^k$ represent the current scaled (by $\frac{\mu_0}{\mu_k}$) infeasibility:

$$\begin{aligned} \tilde{b}^k &:= \tfrac{\mu_0}{\mu_k}(Ax + \mu_k(y - \lambda^k) - b - \tfrac{\mu_k}{\mu_0}\bar{b}), \\ \tilde{c}^k &:= \tfrac{\mu_0}{\mu_k}(-Qx + A^T y + z - \mu_k(x - \zeta^k) - c - \tfrac{\mu_k}{\mu_0}\bar{c}). \end{aligned}$$

The set in (2.17) contains all points (x, y, z) whose scaled infeasibilities are bounded by a constant in both $\|\cdot\|_2$ and $\|\cdot\|_{\mathcal{A}}$. Then the family of neighborhoods is

$$\mathcal{N}_{\mu_k}\left(\zeta^k, \lambda^k\right) := \left\{ (x,y,z) \in \tilde{\mathcal{C}}_{\mu_k}\left(\zeta^k, \lambda^k\right) : \begin{array}{l} (x,z) > (0,0), \\ x_i z_i \geq \gamma_\mu \mu_k \text{ for all } i \end{array} \right\}, \tag{2.18}$$

where $\gamma_\mu \in (0,1)$ is a constant that prevents the component-wise complementarity products from approaching zero faster than $\mu_k = (x^k)^T z^k / n$.

Inexact Newton system The search direction $(\Delta x^k, \Delta y^k, \Delta z^k)$ in the inexact `IP-PMM` solves the following inexact Newton system:

$$\begin{bmatrix} -(Q+\mu_k I_n) & A^T & I \\ A & \mu_k I_m & 0 \\ \operatorname{diag}(z^k) & 0 & \operatorname{diag}(x^k) \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = -\begin{bmatrix} -\left(c+\frac{\sigma_k\mu_k}{\mu_0}\bar{c}\right) - Qx^k + A^T y^k + z^k - \sigma_k\mu_k\left(x^k-\zeta^k\right) \\ Ax^k + \sigma_k\mu_k\left(y^k-\lambda^k\right) - \left(b+\frac{\sigma_k\mu_k}{\mu_0}\bar{b}\right) \\ \operatorname{diag}(x^k)\operatorname{diag}(z^k)\mathbb{1}_n - \sigma_k\mu_k\mathbb{1}_n \end{bmatrix} + \begin{bmatrix} \varrho_d^k \\ \varrho_p^k \\ \varrho_\mu^k \end{bmatrix}, \quad (2.19)$$

where the *centering parameter* $\sigma_k \in (0,1)$ is chosen to control how fast the duality measure μ_k must decrease at the next iteration, and $(\varrho_d^k, \varrho_p^k, \varrho_\mu^k)$ model the residuals when the system is solved by iterative methods.

2.2.2 Convergence results of inexact `IP-PMM` on QP

This section presents convergence results for inexact `IP-PMM` on QP, a necessary building block for `Nys-IP-PMM`. We need the following assumption on the residuals in Newton system (2.19).

Assumption 2.2. The residuals $(\varrho_p^k, \varrho_d^k, \varrho_\mu^k)$ in inexact Newton system (2.19) satisfy

$$\varrho_\mu^k = 0, \quad \left\|\left(\varrho_p^k, \varrho_d^k\right)\right\|_2 \le \frac{\sigma_{\min}}{4\mu_0} C_N \mu_k, \quad \left\|\left(\varrho_p^k, \varrho_d^k\right)\right\|_{\mathcal{A}} \le \frac{\sigma_{\min}}{4\mu_0}\gamma_{\mathcal{A}}\rho\mu_k,$$

where $C_N, \gamma_{\mathcal{A}}$ are constants defined as in (2.17), $\sigma_{\min}$ is the lower bound for σ_k in Algorithm 3, and ρ is determined by the starting point chosen in Section 2.2.1.

The assumption $\varrho_\mu^k = 0$ is reasonable since a practical solution method reduces the Newton system (2.19) to the augmented system or the normal equations, and recovers Δz^k using a closed-form formula. The other two assumptions require that the residuals are small whenever the iterate is close to a solution.

We also make the following two standard assumptions to ensure the solution and the problem data are bounded, as in [Pougkakiotis and Gondzio, 2021, Santos et al., 2019].

Assumption 2.3. The primal-dual QP pair in (P)–(D) has an optimal solution (x^*, y^*, z^*) with $\|x^*\|_\infty \le C^*$, $\|y^*\|_\infty \le C^*$ and $\|z^*\|_\infty \le C^*$, for a constant $C^* \ge 0$ independent of n and m.

Assumption 2.4. Let $\eta_{\min}(A)$ (resp. $\eta_{\max}(A)$) denote the minimum (resp. maximum) singular value of A and $\nu_{\max}(Q)$ denote the maximum eigenvalue of Q. The constraint matrix of (P) has full row rank $\operatorname{rank}(A) = m$, and there exist constants $C_{\min} > 0$, $C_{\max} > 0$, $C_Q > 0$, and $C_r > 0$, independent of n and m, such that $\eta_{\min}(A) \ge C_{\min}$, $\eta_{\max}(A) \le C_{\max}$, $\nu_{\max}(Q) \le C_Q$, and $\|(c,b)\|_\infty \le C_r$.

Below, we present the key lemmas, Lemmas 2.5 and 2.6, to build the polynomial complexity for inexact `IP-PMM`. These two lemmas guarantee the existence of a stepsize $\bar{\alpha} = O(\frac{1}{n^4})$ at each iteration of inexact `IP-PMM`. The critical analysis are built upon [Pougkakiotis and Gondzio, 2021, Lemma 1–4], which were originally devised for exact `IP-PMM`, yet their proofs still hold for inexact `IP-PMM`, provided that the iterates (x^k, y^k, z^k) lie within the neighborhood $\mathcal{N}_{\mu_k}\left(\zeta^k, \lambda^k\right)$ defined in (2.18). This condition is established in Lemmas 2.5 and 2.6, adapted from [Pougkakiotis and Gondzio, 2021, Lemma 5–6] respectively. The modifications are inspired by the techniques used for inexact `IP-PMM` on linearly constrained SDP [Pougkakiotis and Gondzio, 2022].

Lemma 2.5. *Assume Assumptions 2.3 and 2.4 and suppose the search direction* $\left(\Delta x^k, \Delta y^k, \Delta z^k\right)$ *satisfies inexact Newton system* (2.19) *at an arbitrary iteration* $k \geq 0$ *of inexact* `IP-PMM`. *Let* $D_k^2 := \operatorname{diag}(x^k)\operatorname{diag}(z^k)^{-1}$. *Then*

$$\left\|D_k^{-1}\Delta x^k\right\|_2 = O(n^2\mu_k^{\frac{1}{2}}), \quad \left\|D_k\Delta z^k\right\|_2 = O(n^2\mu_k^{\frac{1}{2}}), \quad \left\|\left(\Delta x^k, \Delta y^k, \Delta z^k\right)\right\|_2 = O(n^3).$$

Proof. The proof follows that of [Pougkakiotis and Gondzio, 2021, Lemma 5] with a few modifications. For simplicity, we only sketch the proof with emphasis on modifications.

Following the beginning of the proof for [Pougkakiotis and Gondzio, 2021, Lemma 5], we invoke [Pougkakiotis and Gondzio, 2021, Lemma 2] and [Pougkakiotis and Gondzio, 2021, Lemma 3] to obtain two triples $(x^*_{r^k}, y^*_{r^k}, z^*_{r^k})$ and $(\tilde{x}, \tilde{y}, \tilde{z})$ satisfying [Pougkakiotis and Gondzio, 2021, Eq. (22)] and [Pougkakiotis and Gondzio, 2021, Eq. (23)] respectively. Using the centering parameter σ_k, instead of defining $\hat{c}$ and $\hat{b}$ as in [Pougkakiotis and Gondzio, 2021, Eq. (29)], we define

$$\hat{c} := -\left(\sigma_k\frac{\bar{c}}{\mu_0} - (1-\sigma_k)\left(x^k - \zeta^k + \frac{\mu_k}{\mu_0}\left(\tilde{x} - x^*_{r^k}\right)\right) + \frac{1}{\mu_k}\varrho_d^k\right), \tag{2.20}$$

$$\hat{b} := -\left(\sigma_k\frac{\bar{b}}{\mu_0} + (1-\sigma_k)\left(y^k - \lambda^k + \frac{\mu_k}{\mu_0}\left(\tilde{y} - y^*_{r^k}\right)\right) + \frac{1}{\mu_k}\varrho_p^k\right), \tag{2.21}$$

where $\bar{c}, \bar{b}, \mu_0$ are defined in (2.16). The additional terms ϱ_d^k/μ_k and ϱ_p^k/μ_k in (2.20)–(2.21) take into account the residuals in Newton system (2.19). With this new pair $(\hat{b}, \hat{c})$, define the triple $(\bar{x}, \bar{y}, \bar{z})$ as in [Pougkakiotis and Gondzio, 2021, Eq. (31)]. Using [Pougkakiotis and Gondzio, 2021, Eq. (22)–(23), (31)] and inexact Newton system (2.19), it can be directly verified that

$$A\bar{x} + \mu_k\bar{y} = 0, \quad -Q\bar{x} + A^T\bar{y} + \bar{z} - \mu_k\bar{x} = 0. \tag{2.22}$$

Additional error terms are absorbed into the inexactness of the Newton system (2.19), yielding (2.22). Given (2.22), the rest of the proof follows the proof of [Pougkakiotis and Gondzio, 2021, Lemma 5]. □

Lemma 2.6 serves as the keystone for Theorem 2.7, establishing a connection between inexact and

exact IP-PMM. The lemma demonstrates that a step in the *inexact* search direction with a sufficiently small stepsize yields sufficient reduction in the duality measure (Lemma 2.6 ((i))) while staying within neighborhood $\mathcal{N}_{\mu_k}\left(\zeta^k, \lambda^k\right)$ (Lemma 2.6 ((iii))). Additionally, it ensures that the stepsizes are uniformly bounded away from zero (for every iteration), on the order of $O(\frac{1}{n^4})$ (Lemma 2.6 ((ii))).

Lemma 2.6. *Instate Assumptions 2.3 and 2.4 and assume that the search direction* $(\Delta x^k, \Delta y^k, \Delta z^k)$ *satisfies the inexact Newton system* (2.19)*, with residuals satisfying Assumption 2.2 for every iteration* $k \geq 0$ *of inexact IP-PMM. Then there exists a step-length* $\bar{\alpha} \in (0,1)$ *such that the following hold for all iterations* $k \geq 0$*:*

(i) for all $\alpha \in [0, \bar{\alpha}]$*,*

$$\left(x^k + \alpha\Delta x^k\right)^T \left(z^k + \alpha\Delta z^k\right) \geq \left(1 - \alpha\left(1 - \frac{\sigma_{\min}}{2}\right)\right)(x^k)^T z^k; \tag{2.23}$$

$$\left(x_i^k + \alpha\Delta x_i^k\right)\left(z_i^k + \alpha\Delta z_i^k\right) \geq \frac{\gamma_\mu}{n}\left(x^k + \alpha\Delta x^k\right)^T \left(z^k + \alpha\Delta z^k\right), \ \forall i \in [n]; \tag{2.24}$$

$$\left(x^k + \alpha\Delta x^k\right)^T \left(z^k + \alpha\Delta z^k\right) \leq (1 - 0.01\alpha)(x^k)^T z^k; \tag{2.25}$$

(ii) $\bar{\alpha} \geq \bar{\kappa}/n^4$ *with* $\bar{\kappa} > 0$ *being a constant independent of* n *and* m*;*

(iii) suppose $\left(x^k, y^k, z^k\right) \in \mathcal{N}_{\mu_k}\left(\zeta^k, \lambda^k\right)$ *and, for any* $\alpha \in (0, \bar{\alpha}]$*, let*

$$\left(x^{k+1}, y^{k+1}, z^{k+1}\right) = \left(x^k + \alpha\Delta x^k, y^k + \alpha\Delta y^k, z^k + \alpha\Delta z^k\right) \tag{2.26}$$

and $\mu_{k+1} = (x^{k+1})^T z^{k+1}/n$*. Then*

$$\left(x^{k+1}, y^{k+1}, z^{k+1}\right) \in \mathcal{N}_{\mu_{k+1}}\left(\zeta^{k+1}, \lambda^{k+1}\right), \tag{2.27}$$

where λ^k *and* ζ^k *are updated as in Algorithm 3.*

Proof. The proof follows that of [Pougkakiotis and Gondzio, 2021, Lemma 6]. Again, we only sketch the outline of the proof and point out the necessary modifications. The assumption $\varrho_\mu^k = 0$ in Assumption 2.2 guarantees the third block of Newton system (2.19) is exact, so same argument in the proof of [Pougkakiotis and Gondzio, 2021, Lemma 6] guarantees that (2.23)–(2.25) hold for every $\alpha \in (0, \alpha^*)$ with exactly the same α^* in [Pougkakiotis and Gondzio, 2021, Eq. (40)], i.e.,

$$\alpha^* := \min\left\{\frac{\sigma_{\min}}{2C_\Delta^2 n^3}, \frac{\sigma_{\min}(1-\gamma_\mu)}{2C_\Delta^2 n^4}, \frac{0.49}{C_\Delta^2 n^3}, 1\right\}, \tag{2.28}$$

where $C_\Delta > 0$ is a constant such that $(\Delta x^k)^T \Delta z^k \leq \|D_k^{-1}\Delta x^k\|_2 \|D_k \Delta z^k\|_2 \leq C_\Delta^2 n^4 \mu_k$ from Lemma 2.5. Next, we must find the maximal $\bar{\alpha} \in (0, \alpha^*]$ such that

$$\left(x^k(\alpha), y^k(\alpha), z^k(\alpha)\right) \in \mathcal{N}_{\mu_k(\alpha)}(\zeta^k, \lambda^k), \quad \text{for all } \alpha \in (0, \bar{\alpha}], \tag{2.29}$$

where $\mu_k(\alpha) = \frac{x(\alpha)^T z(\alpha)}{n}$ and

$$\left(x^k(\alpha), y^k(\alpha), z^k(\alpha)\right) = \left(x^k + \alpha\Delta x^k, y^k + \alpha\Delta y^k, z^k + \alpha\Delta z^k\right).$$

By definitions (2.17) and (2.18), Equation (2.29) is equivalent to

$$\|\tilde{r}_p(\alpha), \tilde{r}_d(\alpha)\|_2 \leq C_N \tfrac{\mu_k(\alpha)}{\mu_0}, \quad \|\tilde{r}_p(\alpha), \tilde{r}_d(\alpha)\|_{\mathcal{A}} \leq \gamma_{\mathcal{A}}\rho\tfrac{\mu_k(\alpha)}{\mu_0}, \quad \forall \alpha \in (0, \bar{\alpha}], \tag{2.30}$$

where $\tilde{r}_p(\alpha)$ and $\tilde{r}_d(\alpha)$ are the residuals of two equalities in $\tilde{\mathcal{C}}_{\mu_k}\left(\zeta^k, \lambda^k\right)$ in (2.17):

$$\begin{aligned}
\tilde{r}_p(\alpha) &= Ax^k(\alpha) + \mu_k(\alpha)\left(y^k(\alpha) - \lambda^k\right) - \left(b + \frac{\mu_k(\alpha)}{\mu_0}\bar{b}\right), \\
\tilde{r}_d(\alpha) &= -Qx^k(\alpha) + A^T y^k(\alpha) + z^k(\alpha) - \mu_k(\alpha)\left(x^k(\alpha) - \zeta^k\right) - \left(c + \frac{\mu_k(\alpha)}{\mu_0}\bar{c}\right).
\end{aligned}$$

Following the calculations in the proof of [Pougkakiotis and Gondzio, 2021, Lemma 6], now with the inexact Newton system (2.19), the two residuals $\tilde{r}_p(\alpha)$ and $\tilde{r}_d(\alpha)$ have additional inexact error terms $\alpha\varrho_p^k$ and $\alpha\varrho_d^k$ respectively:

$$\begin{aligned}
\tilde{r}_p(\alpha) &= (1-\alpha)\frac{\mu_k}{\mu_0}\tilde{b}^k + \alpha^2(\sigma_k - 1)\mu_k\Delta y^k + \alpha^2\tfrac{(\Delta x^k)^T\Delta z^k}{n}\left(y^k - \lambda^k + \alpha\Delta y^k - \frac{\bar{b}}{\mu_0}\right) + \alpha\varrho_p^k, \\
\tilde{r}_d(\alpha) &= (1-\alpha)\frac{\mu_k}{\mu_0}\tilde{c}^k - \alpha^2(\sigma_k - 1)\mu_k\Delta x^k - \alpha^2\tfrac{(\Delta x^k)^T\Delta z^k}{n}\left(x^k - \zeta^k + \alpha\Delta x^k + \frac{\bar{c}}{\mu_0}\right) + \alpha\varrho_d^k.
\end{aligned}$$

Using the same quantities $\xi_2 = O(n^4\mu_k)$ and $\xi_{\mathcal{A}} = O(n^4\mu_k)$ defined as in [Pougkakiotis and Gondzio, 2021, Eq. (44)], we have the bounds

$$\begin{aligned}
\|(\tilde{r}_p(\alpha), \tilde{r}_d(\alpha))\|_2 &\leq (1-\alpha)C_N\frac{\mu_k}{\mu_0} + \alpha^2\mu_k\xi_2 + \alpha\|(\varrho_p^k, \varrho_d^k)\|_2, \\
\|(\tilde{r}_p(\alpha), \tilde{r}_d(\alpha))\|_{\mathcal{A}} &\leq (1-\alpha)\gamma_{\mathcal{A}}\rho\frac{\mu_k}{\mu_0} + \alpha^2\mu_k\xi_{\mathcal{A}} + \alpha\|(\varrho_p^k, \varrho_d^k)\|_{\mathcal{A}},
\end{aligned}$$

for all $\alpha \in (0, \alpha^*]$, where α^* is given by (2.28). Assumption 2.2 on error bounds for the inexactness further yields that, for all $\alpha \in (0, \alpha^*]$,

$$\|\tilde{r}_p(\alpha), \tilde{r}_d(\alpha)\|_2 \leq \frac{C_N}{\mu_0}\left(1 - \alpha\left(1 - \frac{\sigma_{\min}}{4} - \alpha\frac{\mu_0\xi_2}{C_N}\right)\right)\mu_k, \tag{2.31}$$

$$\|\tilde{r}_p(\alpha), \tilde{r}_d(\alpha)\|_{\mathcal{A}} \leq \frac{\gamma_{\mathcal{A}}\rho}{\mu_0}\left(1 - \alpha\left(1 - \frac{\sigma_{\min}}{4} - \alpha\frac{\mu_0\xi_{\mathcal{A}}}{\gamma_{\mathcal{A}}\rho}\right)\right)\mu_k. \tag{2.32}$$

On the other hand, (2.23) implies $\mu_k(\alpha) \geq \left(1 - \alpha\left(1 - \frac{\sigma_{\min}}{2}\right)\right)\mu_k$ for all $\alpha \in (0, \alpha^*]$, which together

with (2.31)–(2.32) yields

$$\|\tilde{r}_p(\alpha), \tilde{r}_d(\alpha)\|_2 \leq \frac{\mu_k(\alpha)}{\mu_0} C_N, \quad \text{for all } \alpha \in \left(0, \min\left\{\alpha^*, \frac{\sigma_{\min} C_N}{4\xi_2 \mu_0}\right\}\right];$$
$$\|\tilde{r}_p(\alpha), \tilde{r}_d(\alpha)\|_{\mathcal{A}} \leq \frac{\mu_k(\alpha)}{\mu_0} \gamma_{\mathcal{A}} \rho, \quad \text{for all } \alpha \in \left(0, \min\left\{\alpha^*, \frac{\sigma_{\min} \gamma_{\mathcal{A}} \rho}{4\xi_{\mathcal{A}} \mu_0}\right\}\right].$$

Finally, let

$$\bar{\alpha}^k := \min\left\{\alpha^*, \frac{\sigma_{\min} C_N}{4\xi_2 \mu_0}, \frac{\sigma_{\min} \gamma_{\mathcal{A}} \rho}{4\xi_{\mathcal{A}} \mu_0}\right\} \quad \text{and} \quad \bar{\alpha} := \min_{k \geq 0} \bar{\alpha}^k. \tag{2.33}$$

Since $\xi_2 = O(n^4 \mu_k)$, $\xi_{\mathcal{A}} = O(n^4 \mu_k)$, and μ_k is decreasing, there must exist a constant $\bar{\kappa} > 0$, independent of n, m, and k such that $\bar{\alpha}^k \geq \frac{\bar{\kappa}}{n^4}$ for all $k \geq 0$, and hence $\bar{\alpha} \geq \frac{\bar{\kappa}}{n^4}$. This proves ((i)) and ((ii)). To prove ((iii)), we consider two cases:

- Suppose the estimates ζ^k and λ^k are not updated, that is, $\zeta^{k+1} = \zeta^k$ and $\lambda^{k+1} = \lambda^k$. The choice of $\bar{\alpha}$ in (2.33) guarantees (2.30) and hence (2.29), so that the new iterate in (2.26) satisfies (2.27).

- Suppose `IP-PMM` updates the estimates ζ^k and λ^k, that is, $\zeta^{k+1} = x^{k+1}$ and $\lambda^{k+1} = y^{k+1}$. This happens only when

$$\|(r_p, r_d)\|_2 \leq C_N \frac{\mu_{k+1}}{\mu_0}, \quad \|(r_p, r_d)\|_{\mathcal{A}} \leq \gamma_{\mathcal{A}} \rho \frac{\mu_{k+1}}{\mu_0}, \tag{2.34}$$

  where $r_p = Ax^{k+1} - (b + \frac{\mu_{k+1}}{\mu_0}\bar{b})$ and $r_d = (c + \frac{\mu_{k+1}}{\mu_0}\bar{c}) + Qx^{k+1} - A^T y^{k+1} - z^{k+1}$. Condition (2.34) is exactly equivalent to (2.27).

□

Now, we are ready to present the main result, Theorem 2.7, of this section. Theorem 2.7 proves the polynomial complexity for inexact `IP-PMM`, which extends [Pougkakiotis and Gondzio, 2021, Thm. 2].

Theorem 2.7. *Let $\epsilon \in (0, 1)$ be a given error tolerance. Choose a starting point as in* (2.16) *for inexact `IP-PMM` and let C and ω be positive constants such that $\mu_0 \leq \frac{C}{\epsilon^\omega}$. Assume that at each iteration, the residuals in* (2.19) *satisfy Assumption 2.2. Given Assumptions 2.3 and 2.4, the iterates $\{(x^k, y^k, z^k)\}_{k \geq 0}$ generated by inexact `IP-PMM` satisfy $\mu_k \leq \epsilon$ after $k := \frac{n^4}{0.01\bar{\kappa}} [(1+\omega)\log(1/\epsilon) + \log C] = O\left(n^4 \log \frac{1}{\epsilon}\right)$ iterations, where $\bar{\kappa}$ is a constant independent of n and m defined in Lemma 2.6((ii)).*

Proof. Given Lemma 2.6 that guarantees the existence of a stepsize $\bar{\alpha} = O(\frac{1}{n^4})$ at each iteration of inexact `IP-PMM`, the proof follows [Pougkakiotis and Gondzio, 2021, Thm. 2]. □

Remark 2.8. Given Lemma 2.5 and Lemma 2.6, one can extend [Pougkakiotis and Gondzio, 2021,

Thm. 1] and [Pougkakiotis and Gondzio, 2021, Thm. 3] to inexact `IP-PMM` using the parallel argument. The former states that the duality measure μ_k converges Q-linearly to zero, and the latter guarantees global convergence to an optimal solution of (P)–(D).

2.2.3 Errors from the normal equations

When Q is diagonal, the normal equations (2.6) are solved (inexactly) to determine the search direction, and the other two components, Δx^k and Δz^k, are obtained (exactly) using the closed-form formula (2.7)–(2.8). Denote by ϑ^k the residual in the normal equations at the k-th iteration:

$$[A(Q+\Theta_k^{-1}+\mu_k I_n)^{-1}A^T+\mu_k I_m]\Delta y^k = \xi^k + \vartheta^k. \tag{2.35}$$

The next result characterizes how the error ϑ^k in (2.35) propagates to the Newton system, which will be used in the later analysis for `Nys-IP-PMM`.

Proposition 2.9. *Suppose* $(\Delta x^k, \Delta y^k, \Delta z^k)$ *satisfies Equations* (2.7), (2.8) *and* (2.35) *with* $\|\vartheta^k\|_2 \leq C\mu_k$ *for some constant* $C > 0$ *and* $\mu_k > 0$. *Then* $(\Delta x^k, \Delta y^k, \Delta z^k)$ *satisfies* (2.19) *with* $(\varrho_d^k, \varrho_p^k, \varrho_\mu^k) = (0, \vartheta^k, 0)$ *and the residuals satisfy*

$$\varrho_\mu^k = 0, \quad \left\|(\varrho_p^k, \varrho_d^k)\right\|_2 \leq C\mu_k, \quad \left\|(\varrho_p^k, \varrho_d^k)\right\|_{\mathcal{A}} \leq C\mu_k. \tag{2.36}$$

In particular, Assumption 2.2 is fulfilled if

$$C \leq \frac{\sigma_{\min}}{4\mu_0}\min\left\{C_N, \gamma_{\mathcal{A}}\rho\right\}. \tag{2.37}$$

Proof. We omit the proof for $(\varrho_d^k, \varrho_p^k, \varrho_\mu^k) = (0, \vartheta^k, 0)$ as it follows from direct calculations. The equality and bounds for 2-norm in (2.36) follow directly from the fact $(\varrho_d^k, \varrho_p^k, \varrho_\mu^k) = (0, \vartheta^k, 0)$ and the assumption $\|\vartheta^k\|_2 \leq C\mu_k$. To verify bound for semi-norm $\|(\varrho_p^k, \varrho_d^k)\|_{\mathcal{A}}$, observe that

$$\|(\varrho_p^k, \varrho_d^k)\|_{\mathcal{A}} = \|(0,\epsilon)\|_{\mathcal{A}} = \min_{x,y,z}\left\{\|(x,z)\|_2 \mid Ax = 0, \ -Qx + A^T y + z = \epsilon\right\},$$

and $(\tilde{x}, \tilde{y}, \tilde{z}) = (0, 0, \vartheta^k)$ is a solution to the systems $Ax = 0$ and $-Qx + A^T y + z = \vartheta^k$, so that $\|(\varrho_p^k, \varrho_d^k)\|_{\mathcal{A}} \leq \|(\tilde{x}, \tilde{z})\|_2 = \|(0, \vartheta^k)\|_2 \leq C\mu_k$. □

Proposition 2.9 says that, if the search direction is obtained through solving the normal equations inexactly and using (2.7)–(2.8), then only the second block of Newton system in (2.19) is inexact. The first and third blocks of equations in Newton system remain exact due to the exact formulas in (2.7)–(2.8) for Δx^k and Δz^k.

2.3 Nyström preconditioned IP-PMM (Nys-IP-PMM)

We introduce our main algorithm: Nyström preconditioned IP-PMM (Nys-IP-PMM) in this section. Given the assumption that Q in (P)–(D) is diagonal, Nys-IP-PMM uses the Nyström PCG to solve the normal equations (2.6). Section 2.3.1 provides a convergence analysis for Nys-IP-PMM, which specifies the preconditioner sketch size ℓ required for our convergence theory to hold. We have released a reference implementation of Nys-IP-PMM in Julia [Bezanson et al., 2017], discussed in Section 2.3.2.

2.3.1 Convergence analysis of Nys-IP-PMM

The analysis of inexact IP-PMM in Section 2.2.2 shows that the convergence of inexact IP-PMM requires the residuals of Newton system (2.19) to satisfy the bounds in Assumption 2.2, which hold true if, by Proposition 2.9, NysPCG finds a solution Δy^k to the inexact normal equations (2.35) satisfying (2.37). Indeed, when the sketch size ℓ_k for Nyström PCG is chosen appropriately, the Nyström PCG can achieve a solution such that $\|\vartheta^k\|_2 \leq C\mu_k$, where C satisfies (2.37), in $O(\log \frac{n}{\mu_k})$ PCG iterations with high probability (Lemma 2.11). The result is built upon Lemma 2.10 from [Zhao et al., 2022], which gives the number of Nyström PCG iterations to obtain a solution within error ε.

Lemma 2.10 ([Zhao et al., 2022, Corollary 4.2])**.** *Let $\delta > 0$ and consider regularized system*

$$(N + \delta I)\Delta y = \xi. \tag{2.38}$$

Suppose P is the Nyström preconditioner constructed as in (2.13) *and sketch size*

$$\ell \geq 8\left(\sqrt{d_{\mathit{eff}}(N,\delta)} + \sqrt{8\log(16/\eta)}\right)^2.$$

Let Δy^* denote the true solution of (2.38) *and $\{\Delta y^{(t)}\}_{t\geq 1}$ denote the iterates generated by Nyström PCG on problem* (2.38)*. Then with probability at least $1-\eta$, it holds true that $\|\Delta y^{(t)} - \Delta y^\star\|_2 \leq \varepsilon$ after $t = O(\log(\|\Delta y^\star\|_2/\varepsilon))$ iterations.*

Based on Lemma 2.10, we present next Lemma 2.11 showing that at each iteration k of Nys-IP-PMM, Nyström PCG can return a solution with small residual after a few iterations, when the Newton system (2.19) is solved by the normal equations. For simplicity of presentation, we drop the iteration superindex k in the statement and proof of the lemma.

Lemma 2.11. *Let $N = A(Q + \Theta^{-1} + \mu I_n)^{-1}A^T$. Given Assumptions 2.3 and 2.4, suppose the regularized normal equations* (2.35) *is solved by PCG using the randomized Nyström preconditioner*

(2.13) *with* $\delta = \mu$ *and sketch size*

$$\ell \geq 8\left(\sqrt{d_{\mathit{eff}}(N,\mu)} + \sqrt{8\log(16/\eta)}\right)^2, \tag{2.39}$$

where $d_{\mathit{eff}}(\cdot)$ *is the effective dimension of* N *defined in* (2.15). *Let* $\{\Delta y^{(t)}\}_{t\geq 1}$ *denote the iterates generated by Nyström PCG. Then, with probability at least* $1-\eta$, *after* $t = O\left(\log(\frac{n}{\varepsilon\mu})\right)$ *iterations, the residual satisfies*

$$\left\|(N+\mu I)\Delta y^{(t)} - \xi\right\|_2 \leq \varepsilon. \tag{2.40}$$

Proof. We abbreviate the regularized matrix as $N_\mu = N + \mu I$ and let $\Delta y^\star$ denote the true solution of (2.6) (with $\rho = \delta = \mu$). Let $r^{(t)} = N_\mu \Delta y^{(t)} - \xi$ denote the residual at t-th iteration. It follows from norm consistency and $\|N_\mu\|_2 = \lambda_{\max}(N_\mu)$ that $\|r^{(t)}\|_2 \leq \lambda_{\max}(N_\mu)\|\Delta y^{(t)} - \Delta y^\star\|_2$. Hence, we would have $\|r^{(t)}\|_2 \leq \varepsilon$ if

$$\|\Delta y^{(t)} - \Delta y^\star\|_2 \leq \varepsilon/\lambda_{\max}(N_\mu). \tag{2.41}$$

By Lemma 2.10, with probability at least $1-\eta$, inequality (2.41) holds true after

$$t = O\left(\log\left(\frac{\lambda_{\max}(N_\mu)\|\Delta y^\star\|_2}{\varepsilon}\right)\right) \tag{2.42}$$

PCG iterations. Assumption 2.4 and Lemma 2.5 respectively guarantee that

$$\lambda_{\max}(N_\mu) = \left\|A(Q+\Theta+\mu I)^{-1}A^T\right\|_2 + \mu = O\left(\frac{1}{\mu}\right) \text{ and } \|\Delta y^\star\|_2 = O\left(n^3\right).$$

Therefore, the required number of iterations (2.42) becomes $t = O\left(\log(\frac{n}{\varepsilon\mu})\right)$. □

Now, we are ready to establish the following convergence results for `Nys-IP-PMM`.

Theorem 2.12. *Let* $\epsilon \in (0,1)$ *be a given error tolerance. Instate the assumptions of Theorem 2.7 and suppose the sketch size in* `Nys-IP-PMM` *is taken to be*

$$\ell_k \geq 8\left(\sqrt{d_{\mathit{eff}}(N_k,\mu_k)} + \sqrt{32\log(16(k+2))}\right)^2. \tag{2.43}$$

Then,

(i) with probability at least 0.9, Nyström PCG runs at most $O\left(\log\frac{n}{\epsilon}\right)$ *iterations in each* `Nys-IP-PMM` *iteration;*

(ii) after $k = O(n^4\log(\frac{1}{\epsilon}))$ `Nys-IP-PMM` *iterations, the duality measure satisfies* $\mu_k \leq \epsilon$.

Hence, the total number of matvecs with A *required is at most* $O\left(n^4\log(\frac{1}{\epsilon})\log(\frac{n}{\epsilon})\right)$.

Proof. To guarantee the convergence, the residuals of `IP-PMM` have to satisfy Assumption 2.2, which holds true if, by Proposition 2.9, the residuals ϑ^k in (2.35) satisfies

$$\|\vartheta^k\|_2 \leq \frac{\sigma_{\min}}{4\mu_0} \min\{C_N, \gamma_{\mathcal{A}}\rho\}\, \mu_k. \tag{2.44}$$

At each iteration k of `Nys-IP-PMM`, we take the sketch size ℓ_k as in (2.43) and $\eta_k := \frac{1}{(k+2)^4}$ for all $k \geq 0$, so that (2.39) in Lemma 2.11 is satisfied, which guarantees, with probability at least $1-\eta_k$, that Nyström PCG satisfies (2.44) with order of iterations given by $O(\log \frac{n}{C\mu_k^2}) = O(\log(n/\varepsilon))$, where $C := \frac{\sigma_{\min}}{4\mu_0} \min\{C_N, \gamma_{\mathcal{A}}\rho\}$. To get this, we use the fact that $\frac{1}{\mu_k} = O(\frac{1}{\varepsilon})$ before `Nys-IP-PMM` terminates. By intersecting all these events across iteration count k, we conclude: with probability at least $1-\sum_{k=0}^{\infty} \frac{1}{(k+2)^4} \approx 0.91$, Nyström PCG terminates within $t = O(\log \frac{n}{\varepsilon})$ steps and the residual error of PCG satisfies (2.44) for all $k \geq 0$, which proves ((i)).

Proposition 2.9 then guarantees that $(\Delta x^k, \Delta y^k, \Delta z^k)$ satisfies the inexact Newton system (2.19) in which the residuals satisfy Assumption 2.2. Now, we are under the assumptions of Theorem 2.7, so we have $\mu_k \leq \varepsilon$ after $k = O(n^4 \log \frac{1}{\varepsilon})$ iterations of `Nys-IP-PMM`, as required in ((ii)). □

The sketch size ℓ_k in Theorem 2.12 grows with k to ensure the number of inner iterations of PCG is sublinear in n. In any given iteration, if ℓ_k is chosen smaller than the bound (2.43), item 1) above may fail, but item 2) still holds: in particular, `Nys-IP-PMM` is still guaranteed to converge. Our practical observations in Section 2.4 indicate that a fixed sketch size typically performs well.

2.3.2 Implementation of `Nys-IP-PMM`

Many implementation details deviate slightly from the theory to improve performance, following [Pougkakiotis and Gondzio, 2021]; see supplement for details. Pseudocode for `Nys-IP-PMM` appears in Algorithm 4.

Algorithm 4 Randomized Nyström Preconditioned `IP-PMM` (`Nys-IP-PMM`)

Input: QP data A, b, c, u as in ($\tilde{\text{P}}$), feasibility and optimality tolerance tol
Output: approximate primal-dual solution $(\hat{x}, \hat{y}, \hat{z}, \hat{w}, \hat{s})$ to problem ($\tilde{\text{P}}$)–($\tilde{\text{D}}$)

1: Compute the initial point $(x^{(0)}, y^{(0)}, z^{(0)}, w^{(0)}, s^{(0)})$ as in (A.4).
2: Set initial PMM subproblem parameters: $\lambda^{(0)} = y^{(0)}$, $\zeta^{(0)} = x^{(0)}$, $\rho_0 = 8$, $\delta_0 = 8$.
3: **for** $k = 0, 1, 2, \ldots$ **do**
4: Check the termination criteria using [Pougkakiotis and Gondzio, 2021, Alg. TC].
5: Construct $N_k = A(Q + \Theta_k^{-1} + \rho_k I)^{-1} A^T$ as a linear operator.
6: Choose sketch size ℓ_k.
7: Construct inverse Nyström preconditioner P_k^{-1} by Algorithm 2.
8: Update the iterate and duality measure by Algorithm 6 with P_k^{-1}.
9: Update PMM parameters ρ_k, δ_k, λ^k, ζ^k by [Pougkakiotis and Gondzio, 2021, Alg. PEU].
10: **end for**

Free and box-constrained variables

`Nys-IP-PMM` accepts QPs with free variables and box-constrained variables:

$$
\begin{array}{lllll}
(\tilde{\mathrm{P}}): & \text{minimize} & \frac{1}{2}x^TQx + c^Tx & (\tilde{\mathrm{D}}): & \text{maximize} \quad -\frac{1}{2}x^TQx + b^Ty - u^Ts \\
& \text{subject to} & Ax = b & & \text{subject to} \quad -Qx + A^Ty + z - s = c, \\
& & x_{\mathcal{F}}\text{: free},\ x_{\mathcal{I}} \geq 0, & & y : \text{free},\ z_{\mathcal{I}} \geq 0,\ z_{\mathcal{J}} \geq 0, \\
& & 0 \leq x_{\mathcal{J}} \leq u_{\mathcal{J}}, & & s_{\mathcal{J}} \geq 0,
\end{array}
$$

where $\mathcal{F}$, $\mathcal{I}$, and $\mathcal{J}$ are, respectively, the sets of indices for free variables, non-negative variables, and box-constrained variables; and vector $u_{\mathcal{J}} \in \mathbb{R}^{|\mathcal{J}|}$ denotes the upper bounds. For the ease of presentation, we assume the vectors u, z, s all have length n and satisfy $u_{\mathcal{I}\cup\mathcal{F}} = 0$, $z_{\mathcal{F}} = 0$, and $s_{\mathcal{I}\cup\mathcal{F}} = 0$; and we use the notation $\mathcal{IJ} = \mathcal{I} \cup \mathcal{J}$ and $\mathcal{IF} = \mathcal{I} \cup \mathcal{F}$.

It is very important to directly model box-constrained variables, as encoding them as general inequality constraints increases the size of the normal equations from m to $m+|\mathcal{J}|$. Moreover, many practical problems do have bounds on problem variables.

The PMM subproblem to be solved at each `Nys-IP-PMM` iteration becomes:

$$
\begin{array}{ll}
\text{minimize} & \frac{1}{2}x^TQx + c^Tx + (\lambda^k)^T(b - Ax) + \frac{1}{2\delta_k}\|Ax - b\|_2^2 + \frac{\rho_k}{2}\|x - \zeta^k\|^2, \\
\text{subject to} & x_{\mathcal{F}}\text{: free},\ x_{\mathcal{I}} \geq 0,\ 0 \leq x_{\mathcal{J}} \leq u_{\mathcal{J}}.
\end{array}
\tag{2.45}
$$

We introduce a logarithmic barrier in the Lagrangian to enforce non-negativity and box constraints: $x_{\mathcal{I}} \geq 0$ and $0 \leq x_{\mathcal{J}} \leq u_{\mathcal{J}}$, and derive the corresponding linear system to determine the search direction (analogous to (2.6)–(2.8)) in the following five variables: primal variable x, Lagrange multiplier y (associated with $Ax = b$), dual variable $z_{\mathcal{IJ}}$ (associated with $x_{\mathcal{IJ}} \geq 0$), slack primal variable $w_{\mathcal{J}} \coloneqq u_{\mathcal{J}} - x_{\mathcal{J}}$, and dual variable $s_{\mathcal{J}}$ (associated with $x_{\mathcal{J}} \leq u_{\mathcal{J}}$). Pseudocode for the search direction appears in Algorithm 5, which takes RHS vectors (explicitly defined later in Section 2.3.2), current iterate, current parameters, and preconditioner as input. A detailed derivation for (2.46)–(2.50) is available in Appendix A.2.1 of the supplementary materials.

Initial point

Our choice of initial point is inspired by Mehrotra's initial point [Mehrotra, 1992] and that in [Pougkakiotis and Gondzio, 2021]. A candidate point is first constructed by ignoring the non-negative constraints and then is modified to ensure the positivity of $x_{\mathcal{IJ}}$, $z_{\mathcal{IJ}}$, $w_{\mathcal{J}}$, $s_{\mathcal{J}}$. See details in Appendix A.2.2 of the supplementary materials.

Algorithm 5 Solve Newton system

Input: RHS vectors r_p^k, r_d^k, r_u^k, r_{xz}^k, r_{ws}^k; current iterates x^k, z^k, w^k, s^k; parameters ρ_k, δ_k; inverse preconditioner P_k^{-1}

Output: directions Δx^k, Δy^k, Δz^k, Δw^k, Δs^k that satisfy (A.3)

1: Compute the auxiliary vectors ξ_{au}^k, $\xi^k \in \mathbb{R}^n$:

$$\left(\xi_{\text{au}}^k\right)_j = \begin{cases} 0, & \text{if } j \in \mathcal{F}; \\ -(x_j^k)^{-1}(r_{xz}^k)_j, & \text{if } j \in \mathcal{I}; \\ -(x_j^k)^{-1}(r_{xz}^k)_j + (w_j^k)^{-1}(r_{ws}^k)_j - (w_j^k)^{-1}s_j^k(r_u)_j, & \text{if } j \in \mathcal{J}, \end{cases}$$

$$\xi^k = r_p^k + A(Q + \Theta_k^{-1} + \rho_k I)^{-1}(r_d^k + \xi_{\text{au}}^k).$$

2: Use PCG with preconditioner P_k to solve Δy^k from the normal equations:

$$[A(Q + \Theta_k^{-1} + \rho_k I_n)^{-1}A^T + \delta_k I_m]\Delta y^k = \xi^k. \tag{2.46}$$

3: Compute Δx^k, Δz^k, Δw^k, Δs^k from closed-form formulae:

$$\Delta x^k = (Q + \Theta_k^{-1} + \rho_k I)^{-1}(A^T\Delta y^k - r_d^k - \xi_{\text{au}}^k); \tag{2.47}$$

$$\Delta w_{\mathcal{J}}^k = (r_u^k)_{\mathcal{J}} - \Delta x_{\mathcal{J}}^k, \quad \Delta w_{\mathcal{I}\mathcal{F}}^k = 0; \tag{2.48}$$

$$\Delta z_{\mathcal{I}\mathcal{J}}^k = \text{diag}(x_{\mathcal{I}\mathcal{J}}^k)^{-1}(r_{xz}^k)_{\mathcal{I}\mathcal{J}} - \text{diag}(z_{\mathcal{I}\mathcal{J}}^k)\Delta x_{\mathcal{I}\mathcal{J}}^k, \quad \Delta z_{\mathcal{F}}^k = 0; \tag{2.49}$$

$$\Delta s_{\mathcal{J}}^k = \text{diag}(w_{\mathcal{J}}^k)^{-1}(r_{ws}^k)_{\mathcal{J}} - \text{diag}(s_{\mathcal{J}}^k)\Delta w_{\mathcal{J}}^k, \quad \Delta s_{\mathcal{I}\mathcal{F}}^k = 0. \tag{2.50}$$

Algorithm 6 Mehrotra's predictor-corrector method

Input: current iterate x^k, y^k, z^k, w^k, s^k; PMM parameters λ^k, ζ^k, ρ_k, δ_k; inverse preconditioner P_k^{-1}

Output: new iterate x^{k+1}, y^{k+1}, z^{k+1}, w^{k+1}, s^{k+1} and new duality measure μ_{k+1}

1: Compute predictor step $(\Delta_p x^k, \Delta_p y^k, \Delta_p z^k, \Delta_p w^k, \Delta_p s^k)$ by Algorithm 5 with:

$$\begin{aligned} r_d^k &= c + Qx^k - A^T y^k - z^k + s^k + \rho_k(x^k - \zeta^k), \\ r_p^k &= b - Ax^k - \delta_k(y^k - \lambda^k), \\ (r_u^k)_{\mathcal{J}} &= u_{\mathcal{J}} - x_{\mathcal{J}}^k - w_{\mathcal{J}}^k, \quad (r_u^k)_{\mathcal{IF}} = 0, \\ (r_{xz}^k)_{\mathcal{IJ}} &= -\operatorname{diag}(x_{\mathcal{IJ}}^k)\operatorname{diag}(z_{\mathcal{IJ}}^k)\mathbb{1}_{|\mathcal{IJ}|}, \quad (r_{xz}^k)_{\mathcal{F}} = 0, \\ (r_{ws}^k)_{\mathcal{J}} &= -\operatorname{diag}(w_{\mathcal{J}}^k)\operatorname{diag}(s_{\mathcal{J}}^k)\mathbb{1}_{|\mathcal{J}|}, \quad (r_{ws}^k)_{\mathcal{IF}} = 0, . \end{aligned} \tag{2.51}$$

2: Find stepsizes $\bar{\alpha}_p$ and $\bar{\alpha}_d$ for predictor step using (A.5)–(A.6).

3: Compute the hypothetical measure for predictor step:

$$\mu_{\text{aff}} = \frac{(x_{\mathcal{IJ}}^k + \bar{\alpha}_p \Delta_p x_{\mathcal{IJ}}^k)^T (z_{\mathcal{IJ}}^k + \bar{\alpha}_d \Delta_p z_{\mathcal{IJ}}^k) + (w_{\mathcal{J}}^k + \bar{\alpha}_p \Delta_p w_{\mathcal{J}}^k)^T (s_{\mathcal{J}}^k + \bar{\alpha}_d \Delta_p s_{\mathcal{J}}^k)}{|\mathcal{IJ}| + |\mathcal{J}|}. \tag{2.52}$$

4: Compute the duality measure μ_k and $\tilde{\mu}^k$:

$$\mu_k = \frac{\left(x_{\mathcal{IJ}}^k\right)^T z_{\mathcal{IJ}}^k + \left(w_{\mathcal{J}}^k\right)^T s_{\mathcal{J}}^k}{|\mathcal{IJ}| + |\mathcal{J}|}, \quad \tilde{\mu}^k := \frac{\mu_{\text{aff}}^3}{\mu_k^2}.$$

5: Compute corrector step $(\Delta_c x^k, \Delta_c y^k, \Delta_c z^k, \Delta_c w^k, \Delta_c s^k)$ by Algorithm 5 with:

$$\begin{aligned} & r_d^k = 0, \quad r_p^k = 0, \quad (r_u^k)_{\mathcal{J}} = 0, \\ & (r_{xz}^k)_{\mathcal{IJ}} = \tilde{\mu}^k \mathbb{1}_{|\mathcal{IJ}|} - \operatorname{diag}(\Delta_p x_{\mathcal{IJ}}^k)\operatorname{diag}(\Delta_p z_{\mathcal{IJ}}^k)\mathbb{1}_{|\mathcal{IJ}|}, \quad (r_{xz}^k)_{\mathcal{F}} = 0, \\ & (r_{ws}^k)_{\mathcal{J}} = \tilde{\mu}^k \mathbb{1}_{|\mathcal{J}|} - \operatorname{diag}(\Delta_p w_{\mathcal{J}}^k)\operatorname{diag}(\Delta_p s_{\mathcal{J}}^k)\mathbb{1}_{|\mathcal{J}|}, \quad (r_{ws}^k)_{\mathcal{IF}} = 0. \end{aligned} \tag{2.53}$$

6: Compute the final search direction:

$$\begin{aligned} (\Delta x^k, \Delta w^k) &= (\Delta_p x^k + \Delta_c x^k, \Delta_p w^k + \Delta_c w^k) \\ (\Delta y^k, \Delta z^k, \Delta s^k) &= (\Delta_p y^k + \Delta_c y^k, \Delta_p z^k + \Delta_c z^k, \Delta_p s^k + \Delta_c s^k) \end{aligned}$$

7: Find stepsizes α_p and α_d for final search direction using (A.5)–(A.6).

8: Update the iterates:

$$\begin{aligned} (x^{k+1}, w^{k+1}) &= (x^k, w^k) + \alpha_p(\Delta x^k, \Delta w^k) \\ (y^{k+1}, z^{k+1}, s^{k+1}) &= (y^k, z^k, s^k) + \alpha_d(\Delta y^k, \Delta z^k, \Delta s^k) \end{aligned}$$

9: Update the duality measure: $\mu_{k+1} = \frac{\left(x_{\mathcal{IJ}}^{k+1}\right)^T z_{\mathcal{IJ}}^{k+1} + \left(w_{\mathcal{J}}^{k+1}\right)^T s_{\mathcal{J}}^{k+1}}{|\mathcal{IJ}| + |\mathcal{J}|}$

Mehrotra's predictor-corrector method

Our `Nys-IP-PMM` method determines the search direction by Mehrotra's predictor-corrector method [Mehrotra, 1992], which is considered the industry-standard approach in IPMs. The method automatically selects the centering parameter σ_k by solving two Newton systems: the first chooses σ_k,

and the second chooses the search direction. See Algorithm 6. Fortunately, the two systems differ only in their RHS vectors, so the preconditioner can be reused.

The first direction, called *predictor step* and also known as *affine-scaling direction*, aims at reducing the duality measure μ as much as possible and ignores centrality by setting $\sigma_k = 0$. Hence, the RHS vectors r_p, r_d, and r_u are taken to be the negative residuals of primal, dual, and upper-bounded constraints respectively; while r_{xz} and r_{ws} are negative residuals of complementary slackness (see (2.51)).

Given the predictor step and associated stepsizes, Mehrotra suggests the centering parameter $\sigma_k = (\mu_{\text{aff}}/\mu_k)^3$, where μ_{aff} is the hypothetical duality measure resulting from making the predictor step (see (2.52)). The second direction, called the *corrector step*, consists of centering component and correcting component that attempts to compensate for error in the linear approximation to complementary slackness. See (2.53) for the associated RHS vectors. The final search direction is the sum of predictor step and corrector step.

2.4 Numerical experiments for `Nys-IP-PMM`

We demonstrate the effectiveness of `Nys-IP-PMM` numerically in this section. We compare `Nys-IP-PMM` with other matrix-free `IP-PMM`s: `IP-PMM` using CG (CG-IP-PMM) and `IP-PMM` using PCG with the partial Cholesky preconditioner (Chol-IP-PMM). In Section 2.4.1, we show that matrix-free algorithms can handle a large-scale portfolio optimization problem that is too large for IPMs using a direct inner linear system solver. Among the matrix-free methods, `Nys-IP-PMM` is fastest. Section 2.4.2 compares these matrix-free methods on linear support vector machine (SVM) problems [Fine and Scheinberg, 2001, Woodsend and Gondzio, 2011, Gondzio and Grothey, 2009, Woodsend and Gondzio, 2009]. `Nys-IP-PMM` beats Chol-IP-PMM substantially. It also improves on CG-IP-PMM when the constraint matrix A is dense, and is never much slower.

All experiments were run on a server with 2x 64-core AMD EPYC 7763 @ 2.45GHz, 1 TB RAM. We only use 16 cores for any given experiment since the calls to BLAS are limited to 16 threads. The implementation uses Julia v1.9.1 [Bezanson et al., 2017]. Code is publicly available at `https://github.com/udellgroup/Nys-IP-PMM`. Input matrices Q and A are passed as linear operator objects using `LinearOperators.jl` [Orban and Siqueira, 2024], while `Krylov.jl` [Montoison and Orban, 2023] is selected, because of its efficiency, as the implementation for the conjugate gradient (CG) method. When entrywise access to A is available, our implementation for the partial Cholesky preconditioner provides an option to compute the diagonal of N_k more efficiently (see footnote 2). Our experiments use this option for Chol-IP-PMM. Hence the performance of Chol-IP-PMM is *better* than it would be in the matrix-free setting, understating the advantage of `Nys-IP-PMM`.

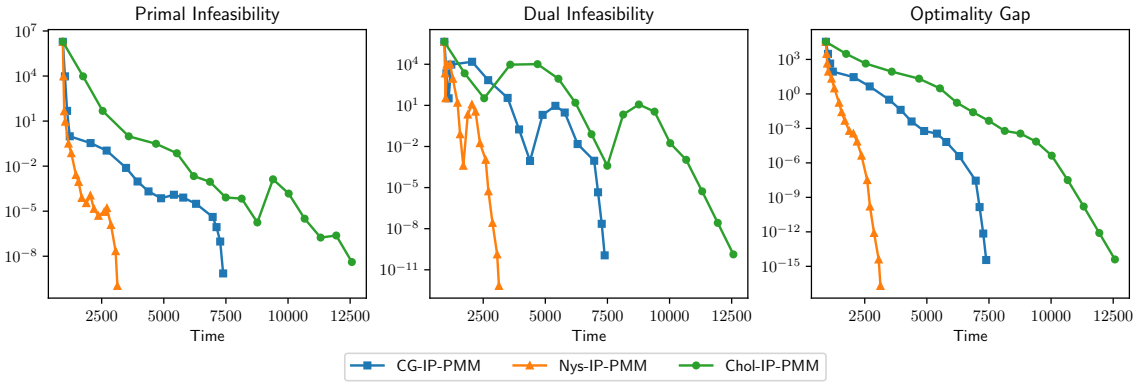


Figure 2.1: Relative primal/dual infeasibility and optimality gap versus cumulative time for portfolio optimization problem with $n = 80\,000$, $d = 50\,000$, and $s = 100$.

2.4.1 Large-scale portfolio optimization problem

This experiment showcases the use of `Nys-IP-PMM` for large-scale separable QPs. We construct a synthetic portfolio optimization problem, which aims at determining the asset allocation to maximize risk-adjusted returns while constraining correlation with market indexes or competing portfolios:

$$\begin{array}{ll} \text{minimize} & -r^T x + \gamma x^T \Sigma x \\ \text{subject to} & Mx \leq u,\ \mathbb{1}_n^T x = 1,\ x \geq 0, \end{array} \tag{2.54}$$

where variable $x \in \mathbb{R}^n$ represents the portfolio, $r \in \mathbb{R}^n$ denotes the vector of expected returns, $\gamma > 0$ denotes the risk aversion parameter, $\Sigma \in \mathbb{S}_n^+(\mathbb{R})$ represents the risk model covariance matrix, each row of $M \in \mathbb{R}^{d\times n}$ represents another portfolio, and $u \in \mathbb{R}^d$ upper bounds of the correlations. We assume a factor model for the covariance matrix $\Sigma = FF^T + D$, where $F \in \mathbb{R}^{n\times s}$ is the factor loading matrix and $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix representing asset-specific risk, and solve the equivalent reformulation of (2.54). See Appendix A.3.1 for details.

In our experiment, a true covariance matrix $\Sigma^\natural$ is generated with rapid spectral decay followed by slow decay; the rows of M are normally distributed; the expected returns r are sampled from standard normal; correlation upper bounds u are sampled from a uniform distribution, and γ is set to 1. To obtain the risk model Σ, we use the randomized rank-s Nyström approximation on $\Sigma^\natural$ to compute the factors F and define D by $\operatorname{diag}(D) = \operatorname{diag}(\Sigma - FF^T)$. The demonstrated problem instance has dimensions $n = 80\,000$, $d = 50\,000$, and $s = 100$. The normal equations to solve at each iteration have size $d + s + 1 = 50\,101$. The tolerance for relative primal-dual infeasibility and optimality gap μ is set as $\epsilon = 10^{-8}$. Figure 2.1 shows history of these three convergence measurements versus wallclock time. `Nys-IP-PMM` with sketchsize $\ell = 20$ converges 2x faster than CG-IP-PMM and more than 3x faster than Chol-IP-PMM.

2.4.2 Support vector machine (SVM) problem

The linear support vector machine (SVM) problem solves a binary classification task on n samples with d features [Deisenroth et al., 2020, Chap. 12]. It is well-known that dual linear SVM with ℓ_1-regularization can be formulated as a convex QP [Fine and Scheinberg, 2001, Woodsend and Gondzio, 2011, Gondzio and Grothey, 2009, Woodsend and Gondzio, 2009]. See the formulation we used in Appendix A.3.2. In this example, the constraint matrix A contains the feature matrix as a block and thus can be very dense. We use the medium-to-large sized real datasets from UCI [Kelly et al.] and LIBSVM [Chang and Lin, 2011] listed in Table 2.3.

Table 2.3: SVM Datasets information.

Datasets	Features d	Instances n	nnz %	Datasets	Features d	Instances n	nnz %
SensIT	100	98 528	100.0	Arcene	10 000	100	54.1
CIFAR10	3072	60 000	99.7	Dexter	20 000	300	0.5
STL-10	27 648	13 000	96.3	Sector	55 197	9619	0.3
RNASeq	20 531	801	85.8				

SVM results

Table 2.4: `Nys-IP-PMM` vs other preconditioners on SVM problem.

Datasets	`Nys-IP-PMM`			**CG-IP-PMM**			**Chol-IP-PMM**			
	`IP-PMM` Outer Iter.	Sum of Inner Iter.	Time (sec.)	`IP-PMM` Outer Iter.	Sum of Inner Iter.	Time (sec.)	`IP-PMM` Outer Iter.	Sum of Inner Iter.	Time (sec.)	Rank $\ell_k \equiv \ell$
CIFAR10	12	18 692	**1103.26**	*	*	*	*	*	*	200
RNASeq	14	4793	**8.40**	14	24 689	19.59	14	20 488	769.95	200
STL-10	12	119 760	**7691.41**	12	492 925	30 016.8	12	491 503	103 681.65	800
SensIT	15	1621	**5.69**	15	4569	14.12	15	1164	11.05	50
Arcene	5	386	**0.233**	5	649	0.278	5	6194	18.15	20
Dexter	4	344	0.202	4	333	**0.122**	4	3754	20.81	10
Sector	16	3570	**16.33**	16	4493	16.99	16	6940	681.96	20

This section evaluates the performance of `Nys-IP-PMM` on the SVM problem (A.9) using datasets listed in Table 2.3. We solve SVM problems with `Nys-IP-PMM`, CG-IP-PMM, and Chol-IP-PMM. All three methods are matrix-free, meaning they do not rely on direct access to the entries of $N_k + \delta_k I$. The results, presented in Table 2.4, highlight the effectiveness `Nys-IP-PMM`. In contrast, Chol-IP-PMM does not improve runtime as much, and in some cases, can even increase the required number of inner iterations (e.g., for Arcene, Dexter, and Sector). Its construction time can also be significant. Overall, Chol-IP-PMM is often much slower than CG-IP-PMM.

The randomized Nyström preconditioner accelerates `IP-PMM` more effectively on dense datasets such as SensIT, CIFAR10, STL-10, and RNASeq. As indicated in Table 2.2, the construction time of the Nyström preconditioner differs significantly from that of the partial Cholesky preconditioner

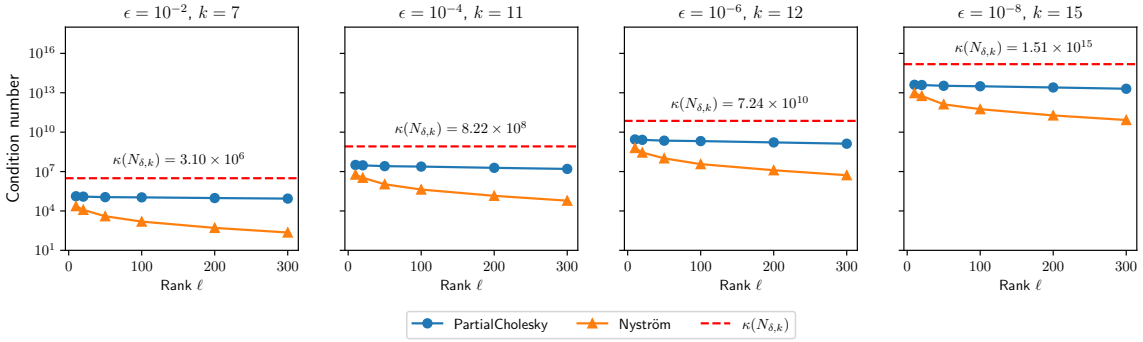


Figure 2.2: The condition numbers before and after preconditioning. The subplots represent distinct stages of `IP-PMM` convergence. The red dashed line shows the unpreconditioned condition number $\kappa(N_{\delta,k})$. Blue circles denote the condition number after partial Cholesky preconditioning, while orange triangles represent the condition number after Nyström preconditioning.

when A is dense. Datasets Arcene and Dexter are relatively easy, requiring few inner iterations even without a preconditioner.

In summary, the randomized Nyström preconditioner is a competitive choice for SVM problems. It effectively preconditions ill-conditioned instances without much slowdown for well-conditioned instances.

Condition numbers at different `IP-PMM` stages

We explore the impact of the preconditioner across various stages of `IP-PMM` by examining the condition number. We record the regularized normal equations (2.6) in the iteration of which `IP-PMM` attains primal-dual infeasibility and duality measure less than different tolerance levels $\epsilon \in \{10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}\}$. Then we compute the condition number of $N_\delta = N_k + \delta_k I$ before and after applying different preconditioners. For efficiency, this experiment concerns a smaller SVM problem formed from 1000 samples from CIFAR10; the normal equations have size 3073×3073.

Section 2.4.2 demonstrates the condition numbers of the preconditioned matrices with different choices $\ell \in \{10, 20, 50, 100, 200, 300\}$. The red dashed horizontal line denotes the condition number $\kappa(N_\delta)$ before preconditioning. The blue circles denote the condition numbers after partial Cholesky preconditioning, while the orange triangles denote those after Nyström preconditioning. As expected, the normal equation matrix N_δ becomes more and more ill-conditioned as `IP-PMM` converges. The preconditioners have similar effects at each stage of convergence. Both preconditioners reduce the condition number by 1–2 orders of magnitude. As the rank ℓ increases, the Nyström preconditioner improves the condition number by another 1–2 orders of magnitude, while larger ranks barely improve the effectiveness of partial Cholesky preconditioner.

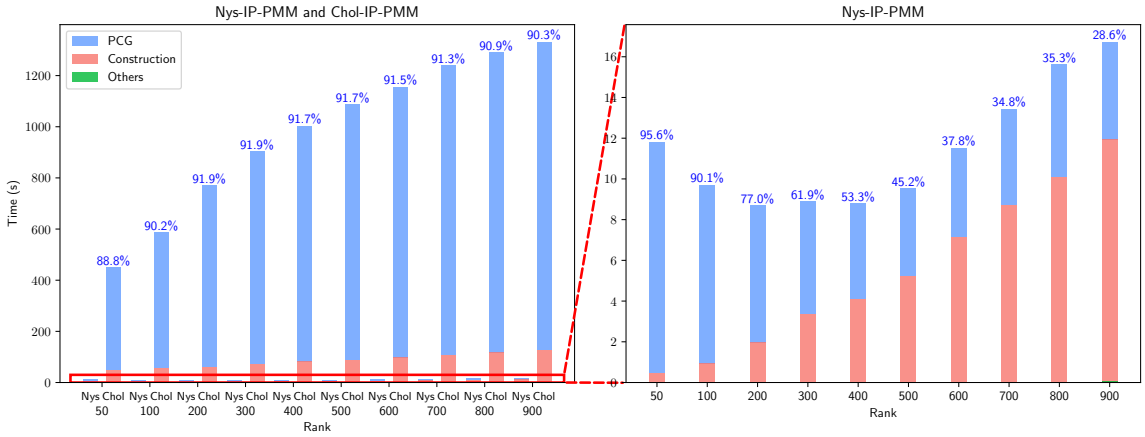


Figure 2.3: Runtime of `Nys-IP-PMM` and Chol-IP-PMM with varying rank ℓ on RNASeq dataset. Left plot compares the two methods; right plot zooms in on the bar chart for `Nys-IP-PMM`. Bar height show total runtime, which is broken into 1) PCG runtime, 2) construction time for preconditioner, and 3) other computation (which is negligible). Blue number above bar gives percentage PCG time.

Rank ℓ versus total/construction/PCG Time

We illustrate trade-offs between the rank ℓ and the wallclock time of `Nys-IP-PMM` and Chol-IP-PMM. Figure 2.3 shows the averaged wallclock time over 4 independent runs for SVM problem on the RNASeq dataset, where the normal equations have dimension $20\,532 \times 20\,532$.

The left subplot of Figure 2.3 shows that the Nyström preconditioner reduces overall runtime dramatically compared to the partial Cholesky preconditioner. The construction time for the partial Cholesky preconditioner increases with increased rank (as expected), while the PCG runtime does not decrease and even increases. We see that partial Cholesky is not an effective preconditioner for this problem.

Zooming in, the right subplot of Figure 2.3 shows the trade-off between the rank parameter ℓ and the total time for `Nys-IP-PMM`. The PCG runtime dominates for smaller ranks, while the time to construct the preconditioner dominates for higher ranks. The overall time exhibits a U-shape: excessively small or large ranks lead to slower computational times. Further improvements to `Nys-IP-PMM` might be achieved by tuning the rank for each problem and by reusing the preconditioner between iterations.

2.5 Concluding remarks

This chapter presents a new algorithm, Nys-IP-PMM, for large-scale separable QP by solving the normal equations in IP-PMM using randomized Nyström preconditioned conjugate gradient (PCG).

Nys-IP-PMM leverages matrix-free preconditioning for computational efficiency. Theoretical analysis and numerical experiments demonstrate superiority of Nys-IP-PMM over existing matrix-free IPM-based methods. Our open-source implementation provides a flexible basis for further exploration of matrix-free regularized IPMs, including new preconditioners, extensions to hardware accelerators such as GPUs, and IPMs for non-separable QP (which require using the augmented system) or other classes of optimization problems. By combining theoretical insights with empirical validation, our work advances the state-of-the-art in large-scale optimization, with potential applications in finance, engineering, and machine learning.

Chapter 3

Hypergradient Descent Method

In this chapter, we consider unconstrained smooth convex optimization problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex and L-smooth with $f^\star := \min_x f(x) > -\infty$. It is well-known that gradient descent with stepsize $1/L$ converges with iteration complexity $\mathcal{O}(\kappa \log(1/\varepsilon))$, where $\kappa = L/\mu$ is the condition number of the problem. Two major techniques have been developed in the literature to accelerate gradient descent. One is to improve the dependence on κ through Nesterov's fast gradient method [Necoara et al., 2019, Nesterov, 2013]; the other is through preconditioning: a positive definite matrix stepsize P_k, known as preconditioner, premultiplies the gradient to improve convergence:

$$x^{k+1} = x^k - P_k \nabla f(x^k). \tag{3.1}$$

Preconditioning has been a standard tool in convex optimization and numerical linear algebra to improve the convergence of gradient descent [Li, 2017, Maddison et al., 2021, Li et al., 2016, Frangella et al., 2022, 2023a] or other iterative methods [Saad, 2003], and it is closely related to the well-known adaptive gradient methods [Duchi et al., 2011, Kingma, 2014, Zhang et al., 2024], either for online learning or for a general optimization problem. Some recent results quantify the effect of adaptive methods on problem conditioning [Das et al., 2024]. In the context of machine learning, adaptively choosing a preconditioner is also relevant to hyperparameter tuning [Hospedales et al., 2021, Necoara et al., 2019], especially choosing a learning rate schedule [Defazio et al., 2024]. The choice of learning rate schedule strongly affects the performance of gradient descent in practice [Defazio et al., 2024, Agarwal et al., 2020]. Various stepsize selection strategies have been proposed to improve the convergence of gradient descent. Examples include line-search [Armijo, 1966], Polyak stepsize [Polyak, 1987], stepsize scheduling [Li et al., 2021, Wang and Yuan, 2023], hypergradient descent [Almeida et al., 1999, Rubio, 2017, Baydin et al., 2018] and the well-known adaptive stepsizes

[Orabona and Pál, 2016, Duchi et al., 2011, Kingma, 2014, Malitsky and Mishchenko, 2020, 2024].

Gao et al. [2024] proposed the online scaled gradient method (OSGM), a framework that adaptively updates the preconditioner P_k and accelerates gradient-based methods through online convex optimization. The main idea is to locally measure the quality of stepsize P_k by the contraction ratio of suboptimality at x^k:

$$r_{x^k}(P_k) := \frac{f(x^{k+1}) - f^\star}{f(x^k) - f^\star} = \frac{f(x^k - P_k \nabla f(x^k)) - f^\star}{f(x^k) - f^\star}.$$

The suboptimality after K iterations of preconditioned gradient descent (3.1) is the product of these ratios $r_{x^k}(P_k)$ and can be further bounded by their cumulative sum using the arithmetic-geometric mean inequality:

$$\frac{f(x^{K+1}) - f^\star}{f(x^1) - f^\star} = \prod_{k=1}^{K} \frac{f(x^{k+1}) - f^\star}{f(x^k) - f^\star} = \prod_{k=1}^{K} r_{x^k}(P_k) \leq \left(\frac{1}{K} \sum_{k=1}^{K} r_{x^k}(P_k) \right)^K. \tag{3.2}$$

Fast convergence is achieved if we prespecify stepsizes $\{P_k\}$ to minimize $\sum_{k=1}^{K} r_{x^k}(P_k)$, which is hard for a given instance f, as the function $r_{x^k}(P)$ depends on the previous sequence of stepsizes and the optimization landscape. However, this problem is ideally suited to online learning, which optimizes a cumulative sum of functions with provable regret guarantees even when the functions are adversarially chosen. For example, updating $\{P_k\}$ with online gradient descent $P_{k+1} = P_k - \eta \nabla r_{x^k}(P_k)$ guarantees sublinear regret with respect to any fixed stepsize $\hat{P}$ [Orabona, 2019]:

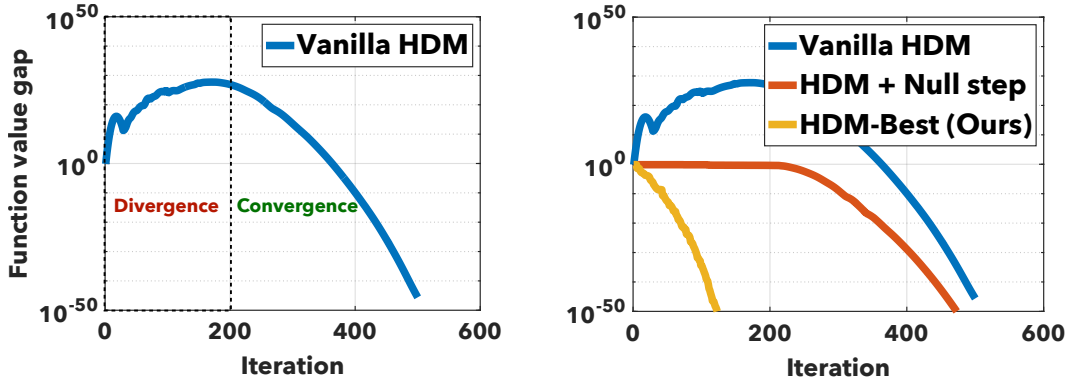
$$\frac{1}{K} \sum_{k=1}^{K} r_{x^k}(P_k) \leq \frac{1}{K} \sum_{k=1}^{K} r_{x^k}(\hat{P}) + \mathcal{O}(\frac{1}{\sqrt{K}}). \tag{3.3}$$

We may take $\hat{P}$ equal to the optimal preconditioner $P_r^\star$ that achieves the optimal condition number $\kappa^\star < \kappa$ and hence the ratio $r_{x^k}(P_r^\star) \leq 1 - \frac{1}{\kappa^\star}$. The regret guarantee (3.3) together with (3.2) implies a convergence rate

$$f(x^{K+1}) - f^\star \leq (f(x^1) - f^\star) \left(1 - \frac{1}{\kappa^\star} + \mathcal{O}\left(\frac{1}{\sqrt{K}} \right) \right)^K, \tag{3.4}$$

as well as an asymptotic iteration complexity $\mathcal{O}(\kappa^\star \log(1/\varepsilon))$. Online scaled gradient methods (OSGM) is a family of first-order methods that update the stepsize on the fly using online learning on a convergence measure.

This chapter focuses on the *hypergradient descent method* (HDM), which was initially proposed by Almeida et al. [1999] as a heuristic for stochastic optimization. It was later tested by Baydin et al. [2018] on modern machine learning problems and exhibited promising performance. In HDM,



(a) Two-phase behavior (b) Addressing instability

Figure 3.1: The convergence behavior of different `HDM` variants on a toy quadratic optimization problem. Figure 3.1a: two-phase convergence behavior of vanilla `HDM`. Figure 3.1b: effect of null step and our best variant `HDM-Best`.

the stepsize α_k is adjusted by another gradient descent update:

$$\begin{aligned}\alpha_{k+1} &= \alpha_k - \tilde{\eta}_k \frac{\mathrm{d}}{\mathrm{d}\alpha}[f(x^k - \alpha \nabla f(x^k))]\big|_{\alpha=\alpha_k} \\ &= \alpha_k - \eta_k \frac{-\langle \nabla f(x^{k+1}), \nabla f(x^k)\rangle}{\|\nabla f(x^k)\|^2},\end{aligned}$$

where the hypergradient stepsize $\tilde{\eta}_k$ is often set to be $\tilde{\eta}_k = \frac{\eta_k}{\|\nabla f(x^k)\|^2}$ for $\eta_k > 0$ to make the update invariant of the scaling of f. Gao et al. [2024] generalized `HDM` to update a matrix stepsize (preconditioner) $P_k \in \mathbb{R}^{n\times n}$ in *preconditioned* gradient descent through the iteration

$$x^{k+1} = x^k - P_k \nabla f(x^k), \tag{3.5}$$

$$P_{k+1} = P_k - \eta_k \frac{-\nabla f(x^{k+1}) \nabla f(x^k)^\top}{\|\nabla f(x^k)\|^2}, \tag{3.6}$$

where (3.6) follows from

$$\nabla_P [f(x^k - P\nabla f(x^k))]\big|_{P=P_k} = -\nabla f(x^{k+1}) \nabla f(x^k)^\top.$$

We call the update (3.5)-(3.6) *vanilla* `HDM` throughout the paper. In practice, matrix stepsize P_k is often set to be diagonal and the update (3.6) simplifies to

$$P_{k+1} = P_k - \eta_k \frac{-\operatorname{diag}(\nabla f(x^{k+1}) \circ \nabla f(x^k))}{\|\nabla f(x^k)\|^2},$$

where $\circ$ is the entry-wise product and $\operatorname{diag}(d)$ is the diagonal matrix with $d \in \mathbb{R}^n$ on the diagonal.

While vanilla HDM has been used heuristically in various applications [Chandra et al., 2022, Wang et al., 2023, Ozkara et al., 2024, Baydin et al., 2018], it can be unstable if the hypergradient stepsize η_k is not carefully tuned [Kunstner et al., 2024, Chandra et al., 2022, Rubio, 2017]. Figure 3.1a shows $f(x^k)$ can spike as high as 10^{30} in the early iterations of vanilla HDM, which would lead users to abandon the algorithm. Surprisingly, our analysis reveals that this behavior of HDM is not true divergence; instead, it can be understood as the warm-up phase of an online learning procedure, and is followed by fast convergence (Figure 3.1a). Moreover, we show in both theory and practice that the explosion of $f(x^k)$ can be circumvented by taking a *null step*, which skips the update whenever the new iterate fails to decrease the objective value, i.e., $f(x^k - P_k\nabla f(x^k)) \geq f(x^k)$. The null steps flatten the objective value curve in the warm-up phase of HDM but cannot shorten the warm-up (Figure 3.1b).

Our analysis exploits the online learning framework by Gao et al. [2024], in which the authors observe that the P-update (3.6) in vanilla HDM can be viewed as online gradient descent with respect to the online surrogate loss

$$h_x(P) := \frac{f(x - P\nabla f(x)) - f(x)}{\|\nabla f(x)\|^2}. \tag{3.7}$$

The function $h_x(P)$, called *hypergradient feedback* in this paper, is a function of preconditioner P and is well-defined for all non-stationary x. To see that (3.6) aligns with the online gradient descent update, notice $\nabla h_{x^k}(P_k) = -\frac{\nabla f(x^{k+1})\nabla f(x^k)^\top}{\|\nabla f(x^k)\|^2}$ so the update (3.6) sets

$$P_{k+1} = P_k - \eta_k \nabla h_{x^k}(P_k).$$

To ensure boundedness of P_k, P-update in [Gao et al., 2024] and in this chapter is projected onto a bounded closed convex candidate set $\mathcal{P}$:

$$P_{k+1} = \Pi_{\mathcal{P}}[P_k - \eta_k \nabla h_{x^k}(P_k)]$$

However, we do *not* require $\mathcal{P}$ to be a subset of positive (semi)definite cone and $\mathcal{P}$ is often taken to be a simple subset such as a ball or a box.

Algorithm 7 Hypergradient Descent Method (HDM)

1: **input** initial point $x^1, P_1 \in \mathcal{P}$ (not necessarily PSD)
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad x^{k+1} = \underset{x \in \{x^k,\, x^k - P_k\nabla f(x^k)\}}{\arg\min} f(x)$
4: $\quad P_{k+1} = \Pi_{\mathcal{P}}[P_k - \eta_k \nabla h_{x^k}(P_k)]$
5: **end for**

Vanilla HDM + null steps (Algorithm 7) was first considered by Gao et al. [2024] and guaranteed to converge globally. However, their analysis is not sufficient to explain the practical behavior of HDM

and provides no advice for how to design a practically efficient HDM. In this paper, we dive deeper into the convergence behavior of HDM (Algorithm 7), establishing sharper global convergence guarantees and conducting a local convergence analysis. Our findings offer new insights into (vanilla) HDM and serve as a foundation to design more efficient and practical variants of HDM. In this chapter, we

- provide the first rigorous convergence analysis for HDM, including both global and local convergence guarantees (Section 3.2) that show HDM can adapt to the local optimization landscape. Our analysis provides several new insights into how HDM adapts to optimization landscapes (Section 3.2.2), why vanilla HDM is unstable in practice (Section 3.2.3), and the connection between HDM and quasi-Newton methods (Section 3.2.4).
- develop and analyze an improved variant HDM + heavy-ball momentum (HDM-HB in Section 3.3), which has the same convergence rate as HDM but is faster than HDM in practice.
- develop a practically efficient variant HDM-Best (Figure 3.1b), which updates x^k by preconditioned gradient descent with heavy-ball momentum and jointly updates P_k and momentum parameter by AdaGrad. Our HDM-Best outperforms most adaptive first-order methods and performs on par with L-BFGS (with memory size 5 or 10) using *less memory* (memory size 1) (Section 4.5).

Notations. We denote Euclidean norm by $\|\cdot\|$ and Euclidean inner product by $\langle\cdot,\cdot\rangle$. The upper and lower case letters A, a respectively denote matrices and scalars. Denote the Frobenius norm by $\|A\|_F := \sqrt{\sum_{ij} a_{ij}^2}$. Define $[\cdot]_+ := \max\{\cdot, 0\}$. We use $\Pi_{\mathcal{C}}[\cdot]$ to denote the orthogonal projection onto a closed convex set $\mathcal{C}$ and use $\mathrm{dist}(x, \mathcal{C}) := \|x - \Pi_{\mathcal{C}}[x]\|$ to denote the distance between a point x and a closed convex set $\mathcal{C}$. Denote the optimal set of f by $\mathcal{X}^\star = \{x : f(x) = f(x^\star)\}$; and the α-sublevel set of f by $\mathcal{L}_\alpha := \{x : f(x) \le \alpha\}$. The condition number of an L-smooth and μ-strongly convex function is $\kappa := L/\mu$. For consistency of notation, a *stepsize* P in this paper always refers to a matrix applied in the gradient update. Define the set of scalar stepsizes $\mathcal{S} := \{P = \alpha I : \alpha \in \mathbb{R}\}$ and diagonal stepsizes $\mathcal{D} := \{P = \mathrm{diag}(d) : d \in \mathbb{R}^n\}$, for which the hypergradient feedback (3.7) simplifies to

$$h_x(\alpha) := \frac{f(x - \alpha\nabla f(x)) - f(x)}{\|\nabla f(x)\|^2} \quad \text{if } \mathcal{P} = \mathcal{S};$$
$$h_x(d) := \frac{f(x - d \circ \nabla f(x)) - f(x)}{\|\nabla f(x)\|^2} \quad \text{if } \mathcal{P} = \mathcal{D}.$$

3.1 Background: HDM and Online Learning

This section establishes the connection between HDM and online learning through the framework in [Gao et al., 2024]. We refer to the following assumptions in the paper.

A1: $f(x)$ is L-smooth and convex.

A2: $f(x)$ is μ-strongly convex with $\mu > 0$.

A3: Closed convex set $\mathcal{P}$ satisfies $0 \in \mathcal{P}, L^{-1}I \in \mathcal{P}$ and $\operatorname{diam}(\mathcal{P}) := \min_{P,Q\in\mathcal{P}} \|P-Q\|_F \le D < \infty$.

3.1.1 Descent Lemma and Hypergradient Feedback

Hypergradient feedback (3.7) is motivated by descent lemma:

$$f(x - \frac{1}{L}\nabla f(x)) - f(x) \le -\frac{1}{2L}\|\nabla f(x)\|^2.$$

The descent lemma states that, under the constant stepsize $P_k \equiv \frac{1}{L}I$, the function value progress of a gradient step is at least proportional to $\|\nabla f(x)\|^2$ with ratio $-1/(2L)$. When an (effective) preconditioner P_k is used, the *effective* smoothness constant decreases, and thus the ratio $h_x(P) = \frac{f(x-P\nabla f(x))-f(x)}{\|\nabla f(x)\|^2}$ is expected to become smaller than $-1/(2L)$, yielding faster convergence. Hence, the ratio $h_x(P)$ is a suitable feedback to measure the quality of a preconditioner. `HDM` uses this feedback to learn a good preconditioner with online gradient descent. The hypergradient feedback $h_x(P)$ has the following properties.

Lemma 3.1 (Extension of Proposition 6.1 in [Gao et al., 2024])**.** *For any $x \notin \mathcal{X}^\star$.*

- *Under* **A1**, *$h_x(P)$ is convex and L-smooth and $h_x(\frac{1}{L}I) \le -\frac{1}{2L}$. Moreover, if* **A2** *holds and $\mathcal{P} \subseteq \mathcal{S}$, then $h_x(\alpha)$ is μ-strongly convex.*
- *Under* **A1** *and* **A3**, *$h_x(P)$ is $(LD+1)$-Lipschitz. Moreover, if* **A2** *holds and $\mathcal{P} \subseteq \mathcal{D}$, then $h_x(d)$ is $\frac{\mu}{(1+LD)^2}$-exponential concave [Hazan et al., 2007].*

Proof. Consider the first property. Convexity and smoothness follow directly from Gao et al. [2024]. To verify strong convexity, note that for $h_x(\alpha) = \frac{f(x-\alpha\nabla f(x))-f(x^\star)}{\|\nabla f(x)\|^2}$

$$h_x''(\alpha) = \frac{\mathrm{d}}{\mathrm{d}\alpha}\left[\frac{\langle \nabla f(x-\alpha\nabla f(x)), \nabla f(x)\rangle}{\|\nabla f(x)\|^2}\right] = \left\langle \frac{\nabla f(x)}{\|\nabla f(x)\|}, \nabla^2 f(x)\frac{\nabla f(x)}{\|\nabla f(x)\|}\right\rangle \ge \mu$$

since $\nabla^2 f(x) \succeq \mu I$ and $x \notin \mathcal{X}^\star$. This completes the proof of the first property.

Next, we consider the second property. Lipschitz continuity also follows from Gao et al. [2024]. To verify exp-concavity, recall that a twice-differentiable function h is β-exp-concave if $\nabla^2 h(x) \succeq \beta\nabla h(x)\nabla h(x)^\top$ for some $\beta \ge 0$. By definition of $\mathcal{D}$,

$$\nabla h_x(P) = -\frac{\nabla f(x) \circ \nabla f(x - P\nabla f(x))}{\|\nabla f(x)\|^2} = -\frac{\operatorname{diag}(\nabla f(x))\nabla f(x - P\nabla f(x))}{\|\nabla f(x)\|^2}$$

and $\nabla^2 h_x(P) = \frac{\operatorname{diag}(\nabla f(x))\nabla^2 f(x-P\nabla f(x))\operatorname{diag}(\nabla f(x))}{\|\nabla f(x)\|^2}$. Using $\nabla^2 f(x - P\nabla f(x)) \succeq \mu I$, we deduce that

$$
\begin{aligned}
&\nabla^2 h_x(P) - \beta \nabla h_x(P) \nabla h_x(P)^\top \\
&= \frac{\operatorname{diag}(\nabla f(x))\nabla^2 f(x - P\nabla f(x))\operatorname{diag}(\nabla f(x))}{\|\nabla f(x)\|^2} - \beta \frac{\operatorname{diag}(\nabla f(x))\nabla f(x - P\nabla f(x))\nabla f(x - P\nabla f(x))^\top \operatorname{diag}(\nabla f(x))}{\|\nabla f(x)\|^4} \\
&= \operatorname{diag}\left(\frac{\nabla f(x)}{\|\nabla f(x)\|}\right)\left[\nabla^2 f(x - P\nabla f(x)) - \beta \frac{\nabla f(x - P\nabla f(x))}{\|\nabla f(x)\|}\frac{\nabla f(x - P\nabla f(x))^\top}{\|\nabla f(x)\|}\right]\operatorname{diag}\left(\frac{\nabla f(x)}{\|\nabla f(x)\|}\right) \\
&\succeq \operatorname{diag}\left(\frac{\nabla f(x)}{\|\nabla f(x)\|}\right)\left[\mu I - \beta \frac{\nabla f(x - P\nabla f(x))}{\|\nabla f(x)\|}\frac{\nabla f(x - P\nabla f(x))^\top}{\|\nabla f(x)\|}\right]\operatorname{diag}\left(\frac{\nabla f(x)}{\|\nabla f(x)\|}\right), \qquad (3.8)
\end{aligned}
$$

where (3.8) uses μ-strong convexity of $f(x)$. Now, it suffices to verify that

$$
\frac{\nabla f(x - P\nabla f(x))}{\|\nabla f(x)\|}\frac{\nabla f(x - P\nabla f(x))^\top}{\|\nabla f(x)\|} \preceq \frac{\mu}{\beta} I \qquad (3.9)
$$

for all $x \notin \mathcal{X}^\star$. Write $\frac{\nabla f(x-P\nabla f(x))}{\|\nabla f(x)\|} = \frac{\nabla f(x)}{\|\nabla f(x)\|} + \frac{\nabla f(x-P\nabla f(x)) - \nabla f(x)}{\|\nabla f(x)\|}$ and let $z := \nabla f(x - P\nabla f(x)) - \nabla f(x)$, we have, by L-smoothness, that $\|z\| \le L\|P\nabla f(x)\| \le LD\|\nabla f(x)\|$ and

$$
\begin{aligned}
\left\|\frac{\nabla f(x - P\nabla f(x))}{\|\nabla f(x)\|}\frac{\nabla f(x - P\nabla f(x))^\top}{\|\nabla f(x)\|}\right\| &= \left\|\left(\frac{\nabla f(x)}{\|\nabla f(x)\|} + \frac{z}{\|\nabla f(x)\|}\right)\left(\frac{\nabla f(x)}{\|\nabla f(x)\|} + \frac{z}{\|\nabla f(x)\|}\right)^\top\right\| \\
&= \left\|\frac{\nabla f(x)\nabla f(x)^\top}{\|\nabla f(x)\|^2} + \frac{z\nabla f(x)^\top}{\|\nabla f(x)\|^2} + \frac{\nabla f(x) z^\top}{\|\nabla f(x)\|^2} + \frac{zz^\top}{\|\nabla f(x)\|^2}\right\| \\
&\le 1 + \frac{2\|z\|}{\|\nabla f(x)\|} + \frac{\|z\|^2}{\|\nabla f(x)\|^2} = (1 + \frac{\|z\|}{\|\nabla f(x)\|})^2 \le (1 + LD)^2.
\end{aligned}
$$

Hence, for $\beta \le \frac{\mu}{(1+LD)^2}$ the relation (3.9) holds. We conclude that $h_x(P) = h_x(d)$ is $\frac{\mu}{(1+LD)^2}$-exponential concave. □

3.1.2 Online Learning Guarantees

Using convexity and Lipschitz continuity of $h_x(P)$, analysis in online learning [Orabona, 2019, Hazan et al., 2016] guarantees sublinear regret for online gradient descent.

Lemma 3.2 (Sublinear regret [Gao et al., 2024])**.** *Under* **A1** *and* **A3***, online gradient descent*

$$
P_{k+1} = \Pi_{\mathcal{P}}[P_k - \eta_k \nabla h_{x^k}(P_k)] \qquad (3.10)
$$

with stepsize $\eta_k \equiv \frac{D}{2(LD+1)\sqrt{K}}$ *or* $\eta_k = \frac{D}{2(LD+1)\sqrt{k}}$ *generates* $\{P_k\}$ *such that*

$$
\sum_{k=1}^{K} h_{x^k}(P_k) - \min_{P\in\mathcal{P}} \sum_{k=1}^{K} h_{x^k}(P) \le \rho_K := 8D(LD+1)\sqrt{K}. \qquad (3.11)
$$

Proof. We use Lipschitz continuity from Lemma 3.1 and (3.14) from Lemma 3.6 by taking $\gamma = 1+LD$ and $\eta = \frac{D}{(LD+1)\sqrt{K}}$. □

If strong convexity **A2** is further assumed and $P_k \in \mathcal{S}$, a different choice of hypergradient stepsize η_k in (3.10) improves the regret to $\log K$.

Lemma 3.3 (Logarithmic regret)**.** *Instate* **A1** *to* **A3** *and suppose* $\mathcal{P} \subseteq \mathcal{S}$*. Then online gradient descent* (3.10) *with* $\eta_k = 1/(k\mu)$ *generates* $\{P_k\}$ *such that* $\sum_{k=1}^K h_{x^k}(P_k) - \min_{P\in\mathcal{P}} \sum_{k=1}^K h_{x^k}(P) \le \frac{(LD+1)^2}{2} \log K$.

Proof. We use Lipschitz continuity and strong convexity from Lemma 3.1 and invoke Lemma 3.7 by taking $\gamma = 1 + LD$. □

Given exponential-concavity of h_x in Lemma 3.1, it is possible to apply online learning algorithms such as the online Newton method [Hazan et al., 2007].

Remark 3.4. The diameter D of candidate stepsize set $\mathcal{P}$ is measured in Frobenius norm (**A3**) and can incur dimension dependence in the regret bound when P_k are diagonal or full matrices. This introduces a trade-off between online regret and the best possible cumulative feedback $\min_{P\in\mathcal{P}} \sum_{k=1}^K h_{x^k}(P)$: full matrices may achieve smaller feedback but have larger dimension factor in the regret bound. Diagonal stepsize often strikes a good balance in practice.

3.1.3 Hypergradient Reduction and `HDM`

One major contribution of Gao et al. [2024] is a reduction that relates the minimization of cumulative hypergradient feedback $\sum_{k=1}^K h_{x^k}(P_k)$ to the function value gap. We provide a sharper version of this reduction.

Lemma 3.5 (Sharper version of Lemma 6.1 in Gao et al. [2024])**.** *Under* **A1***, the iterates generated by Algorithm 7 satisfy*

$$f(x^{K+1}) - f(x^\star) \le \min\left\{ \frac{\Delta^2}{K \max\{\frac{1}{K}\sum_{k=1}^K -h_{x^k}(P_k), 0\}}, f(x^1) - f(x^\star) \right\},$$

where $\Delta = \max_{x\in\mathcal{L}_{f(x^1)}} \min_{x^\star\in\mathcal{X}^\star} \|x - x^\star\|$*. Further, under* **A1** *and* **A2***,*

$$f(x^{K+1}) - f(x^\star) \le (f(x^1) - f(x^\star)) \left(1 - 2\mu \max\left\{\tfrac{1}{K}\textstyle\sum_{k=1}^K -h_{x^k}(P_k), 0\right\}\right)^K.$$

Proof. The proof resembles Gao et al. [2024] and uses a tighter analysis. Consider the optimality

measure $f(x^{K+1}) - f(x^\star)$, and we deduce that

$$\begin{aligned} f(x^{K+1}) - f(x^\star) &= \frac{1}{\frac{1}{f(x^{K+1})-f(x^\star)}} \\ &= \frac{1}{\sum_{k=1}^K \frac{1}{f(x^{k+1})-f(x^\star)} - \frac{1}{f(x^k)-f(x^\star)} + \frac{1}{f(x^1)-f(x^\star)}} \\ &= \frac{1}{\sum_{k=1}^K \frac{f(x^k)-f(x^{k+1})}{[f(x^{k+1})-f(x^\star)][f(x^k)-f(x^\star)]} + \frac{1}{f(x^1)-f(x^\star)}} \\ &= \frac{1}{\sum_{k=1}^K \frac{\max\{-h_{x^k}(P_k),0\}\|\nabla f(x^k)\|^2}{[f(x^{k+1})-f(x^\star)][f(x^k)-f(x^\star)]} + \frac{1}{f(x^1)-f(x^\star)}} \end{aligned}$$

Next, using $f(x) - f(x^\star) \le \|\nabla f(x)\| \cdot \|x - x^\star\|$,

$$\frac{\max\{-h_{x^k}(P_k),0\}\|\nabla f(x^k)\|^2}{[f(x^{k+1}) - f(x^\star)][f(x^k) - f(x^\star)]} \ge \frac{\max\{-h_{x^k}(P_k),0\}\|\nabla f(x^k)\|^2}{[f(x^k) - f(x^\star)]^2} \ge \frac{\max\{-h_{x^k}(P_k),0\}}{\operatorname{dist}(x^k, \mathcal{X}^\star)^2} \ge \frac{\max\{-h_{x^k}(P_k),0\}}{\Delta^2}.$$

Finally, we deduce that

$$\begin{aligned} f(x^{K+1}) - f(x^\star) &\le \frac{\Delta^2}{\sum_{k=1}^K \max\{-h_{x^k}(P_k),0\} + \frac{\Delta^2}{f(x^1)-f(x^\star)}} \\ &\le \frac{\Delta^2}{\max\{\sum_{k=1}^K -h_{x^k}(P_k),0\} + \frac{\Delta^2}{f(x^1)-f(x^\star)}} \\ &\le \min\left\{\frac{\Delta^2}{K\max\{\frac{1}{K}\sum_{k=1}^K -h_{x^k}(P_k),0\}}, f(x^1) - f(x^\star)\right\} \end{aligned}$$

and this completes the proof. □

According to Lemma 3.5, the negative average feedback $\frac{1}{K}\sum_{k=1}^K -h_{x^k}(P_k)$ determines the rate for sublinear/linear convergence of Algorithm 7: larger $\frac{1}{K}\sum_{k=1}^K -h_{x^k}(P_k)$ implies faster convergence. Given the objective $\frac{1}{K}\sum_{k=1}^K -h_{x^k}(P_k)$, `HDM` applies online gradient descent to generate a sequence of preconditioners $\{P_k\}$ that guarantee the following lower bound:

$$\frac{1}{K}\sum_{k=1}^K -h_{x^k}(P_k) \ge \max_{P\in\mathcal{P}} \frac{1}{K}\sum_{k=1}^K -h_{x^k}(P) + o(1). \tag{3.12}$$

Equation (3.12) follows from the sublinear regret $\rho_K = o(K)$ in Lemma 3.2 and Lemma 3.3, implying $\frac{\rho_K}{K} = o(1)$.

3.2 The Convergence Behavior of HDM

This section presents our main convergence results on HDM and consequent insights. All the analyses are based on the online learning framework established in Section 3.1, together with the known online learning results collected in Section 3.2.1.

3.2.1 Known Results in Online Learning Literature

Lemmas below collect auxiliary results known in online learning literature, which will be used later in the analysis of HDM.

Lemma 3.6 (Sublinear dynamic regret [Hazan et al., 2016])**.** *Given a family of convex and γ-Lipschitz losses $\{h_k\}$, online gradient descent $P_{k+1} = \Pi_{\mathcal{P}}[P_k - \eta \nabla h_k(P_k)]$ with constant stepsize $\eta > 0$ generates a sequence of scaling matrices $\{P_k\}$ such that*

$$\sum_{k=1}^{K} [h_k(P_k) - h_k(\hat{P}_k)] \leq \frac{D^2}{2\eta} + \frac{\eta}{2}\gamma^2 K + \frac{D}{2\eta}\mathsf{PL}(\{\hat{P}_k\}). \tag{3.13}$$

where $\{\hat{P}_k\}, \hat{P}_k \in \mathcal{P}$ are arbitrarily chosen competitors and $\mathsf{PL}(\{\hat{P}_k\}) := \sum_{k=1}^{K} \|\hat{P}_k - \hat{P}_{k+1}\|_F$ is the path length of the competitors. In particular, if $\hat{P}_k \equiv P$, then $\mathsf{PL}(\{\hat{P}_k\}) = 0$ and

$$\sum_{k=1}^{K} [h_k(P_k) - h_k(\hat{P}_k)] \leq \frac{D^2}{2\eta} + \frac{\eta}{2}\gamma^2 K. \tag{3.14}$$

Proof. The result follows from a standard dynamic regret analysis from online convex optimization literature, and we adapt the proof for our analysis. For any $P \in \mathcal{P}$, we deduce

$$\begin{aligned}
\|P_{k+1} - P\|_F^2 &= \|\Pi_{\mathcal{P}}[P_k - \eta \nabla h_k(P_k)] - P\|_F^2 \\
&\leq \|P_k - P - \eta \nabla h_k(P_k)\|_F^2 \qquad (3.15) \\
&\leq \|P_k - P\|_F^2 - 2\eta \langle \nabla h_k(P_k), P_k - P \rangle + \eta^2 \|\nabla h_k(P_k)\|_F^2 \\
&\leq \|P_k - P\|_F^2 - 2\eta [h_k(P_k) - h_k(P)] + \eta^2 \gamma^2, \qquad (3.16)
\end{aligned}$$

where (3.15) uses non-expansiveness of orthogonal projection; (3.16) applies convexity and γ-Lipschitz continuity of h_k. Now, let $P = \hat{P}_k$ and we re-arrange to get

$$\begin{aligned}
h_k(P_k) - h_k(\hat{P}_k) &\leq \frac{1}{2\eta}[\|P_k - \hat{P}_k\|_F^2 - \|P_{k+1} - \hat{P}_k\|_F^2] + \frac{\eta}{2}\gamma^2 \\
&= \frac{1}{2\eta}[\|P_k\|_F^2 - \|P_{k+1}\|_F^2 + 2\langle \hat{P}_k, P_{k+1} - P_k \rangle] + \frac{\eta}{2}\gamma^2 \\
&\leq \frac{D^2}{2\eta} + \frac{D}{2\eta}\|P_{k+1} - P_k\|_F + \frac{\eta}{2}\gamma^2, \qquad (3.17)
\end{aligned}$$

where (3.17) uses Cauchy's inequality $\langle \hat{P}_k, P_{k+1} - P_k \rangle \leq \|\hat{P}_k\| \cdot \|P_{k+1} - P_k\| \leq D\|P_{k+1} - P_k\|$. Telescoping,

$$\sum_{k=1}^{K} [h_k(P_k) - h_k(\hat{P}_k)] \leq \frac{D^2}{2\eta} + \frac{\eta}{2}\gamma^2 K + \frac{D}{2\eta}\sum_{k=1}^{K} \|\hat{P}_k - \hat{P}_{k+1}\|_F$$

and this completes the proof. □

Lemma 3.7 (Logarithmic static regret [Orabona, 2019])**.** *Given a family of μ-strongly convex and γ-Lipschitz losses $\{h_k\}$, online gradient descent $P_{k+1} = \Pi_{\mathcal{P}}[P_k - \eta_k \nabla h_k(P_k)]$ with stepsize $\eta_k = 1/(\mu k)$ generates a sequence of scaling matrices $\{P_k\}$ such that $\sum_{k=1}^{K} h_k(P_k) - h_k(P) \leq \frac{1}{2}\gamma^2 \log K$.*

Proof. Using strong convexity, we have $h_k(P) \geq h_k(P_k) + \langle \nabla h_k(P_k), P - P_k \rangle + \frac{\mu}{2}\|P - P_k\|_F^2$ and

$$\begin{aligned}
\|P_{k+1} - P\|_F^2 &\leq \|P_k - P\|_F^2 - 2\eta_k \langle \nabla h_k(P_k), P_k - P \rangle + \eta_k^2 \gamma^2 \\
&\leq \|P_k - P\|_F^2 - 2\eta_k [h_k(P_k) - h_k(P)] + \eta_k^2 \gamma^2 - \mu\eta_k \|P - P_k\|_F^2 \\
&= \frac{k-1}{k}\|P_k - P\|_F^2 - \frac{2}{k\mu}[h_k(P_k) - h_k(P)] + \frac{\gamma^2}{k^2\mu^2}, \qquad (3.18)
\end{aligned}$$

where (3.18) plugs in $\eta_k = 1/(\mu k)$. Re-arranging the terms,

$$h_k(P_k) - h_k(P) \leq \frac{\mu}{2}[(k-1)\|P_k - P\|_F^2 - k\|P_{k+1} - P\|_F^2] + \frac{\gamma^2}{2k\mu}$$

and telescoping gives $\sum_{k=1}^{K}[h_k(P_k) - h_k(P)] \leq \sum_{k=1}^{K} \frac{\gamma^2}{2k\mu} \leq \frac{\gamma^2}{2\mu}(\log K + 1)$, which completes the proof. □

3.2.2 HDM Adapts to the Local Landscape

Our first convergence result follows by combining Lemma 3.5 and Lemma 3.2:

Theorem 3.8 (Static adaptivity)**.** *Under* **A1** *and* (**A1** + **A2**) *respectively, Algorithm 7 satisfies*

$$f(x^{K+1}) - f(x^\star) \leq \min\left\{ \frac{\Delta^2}{K \max\{\gamma_K^\star - \frac{\rho_K}{K}, 0\}}, f(x^1) - f(x^\star) \right\} \qquad \text{(under } \mathbf{A1}\text{)}$$

$$f(x^{K+1}) - f(x^\star) \leq [f(x^1) - f(x^\star)]\left(1 - 2\mu \max\left\{\gamma_K^\star - \tfrac{\rho_K}{K}, 0\right\}\right)^K, \qquad \text{(under } (\mathbf{A1} + \mathbf{A2})\text{)}$$

where Δ is the same as defined in Lemma 3.5, ρ_K is defined in (3.11)*, and*

$$\gamma_K^\star := -\min_{P \in \mathcal{P}} \frac{1}{K}\sum_{k=1}^{K} h_{x^k}(P).$$

Proof. Plugging (3.11) from Lemma 3.2 into Lemma 3.5 completes the proof. □

Theorem 3.8 has two implications: 1) Since $\gamma_K^\star \geq -\frac{1}{K}\sum_{k=1}^K h_{x^k}(\frac{1}{L}I) \geq \frac{1}{2L}$ (by descent lemma) and $\frac{\rho_K}{K} = o(1)$, both upper bounds in Theorem 3.8 converge to 0 when K goes to infinity, guaranteeing global convergence of HDM. 2) More importantly, $\gamma_K^\star$ reflects the possibly improved convergence rate of HDM through the adaptive P-update, which depends on the local optimization landscape. To see this, when K is large and $\frac{\rho_K}{K}$ is negligible, the convergence of HDM is competitive with preconditioned gradient descent (3.1) with any *static* preconditioner. In particular, the optimal $P_K^\star := \arg\min_{P\in\mathcal{P}} \frac{1}{K}\sum_{k=1}^K h_{x^k}(P)$ achieves the rate $\frac{\Delta^2}{K\gamma_K^\star}$. Note that $\gamma_K^\star$ (or $P_K^\star$) depends only on the past trajectory $\{x^k\}_{k\leq K}$; and thus if the algorithm visits a local region with a smaller smoothness constant than the global constant L, one can expect $\gamma_K^\star \gg \frac{1}{2L}$. In summary, HDM *adapts to the local optimization landscape* and leads to faster convergence than standard gradient descent.

We borrow a standard dynamic regret argument in online convex optimization literature [Hazan et al., 2016] to provide an even stronger notion of adaptivity of HDM:

Theorem 3.9 (Dynamic adaptivity)**.** *Under* **A1** *and* (**A1** + **A2**) *respectively, Algorithm 7 satisfies*

$$f(x^{K+1}) - f(x^\star) \leq \min\left\{\frac{\Delta^2}{K\max\{\delta_K^\star - \frac{\rho_K}{K}, 0\}}, f(x^1) - f(x^\star)\right\}; \qquad \text{(under } \mathbf{A1}\text{)}$$

$$f(x^{K+1}) - f(x^\star) \leq (f(x^1) - f(x^\star))\left(1 - 2\mu\max\left\{\delta_K^\star - \tfrac{\rho_K}{K}, 0\right\}\right)^K, \qquad \text{(under } (\mathbf{A1} + \mathbf{A2})\text{)}$$

where Δ is the same as defined in Lemma 3.5,

$$\delta_K^\star := -\min_{\{\hat{P}_k\in\mathcal{P}\}}\left[\tfrac{1}{K}\textstyle\sum_{k=1}^K h_{x^k}(\hat{P}_k) + \mathsf{PL}(\{\hat{P}_k\})\right], \tag{3.19}$$

and $\mathsf{PL}(\{\hat{P}_k\}) := \frac{(LD+1)}{2\sqrt{K}}\sum_{k=1}^K \|\hat{P}_{k+1} - \hat{P}_k\|_F$.

Proof. Invoking Lipschitzness from Lemma 3.1 and Equation (3.13) from Lemma 3.6 with $\gamma = 1 + LD, \eta = \frac{D}{(LD+1)\sqrt{K}}$ gives

$$\sum_{k=1}^K\left[h_{x^k}(P_k) - h_{x^k}(\hat{P}_k)\right] \leq \rho_K + \frac{LD+1}{2}\sqrt{K}\sum_{k=1}^K\|\hat{P}_k - \hat{P}_{k+1}\|_F.$$

Plugging the relation into Lemma 3.5 completes the proof. □

Theorem 3.8 and Theorem 3.9 differ in the constants $\gamma_K^\star$ and $\delta_K^\star$, as the minimum in (3.19) searches over different optimal preconditioners for different h_{x^k}. Theorem 3.9 shows that, even if the sequence $\{x^k\}$ traverses different regions of the landscape, HDM automatically chooses $\hat{P}_k$ to adapt to the local region, at the price of an additional regret term $\mathsf{PL}(\{\hat{P}_k\})$. Note that the upper bounds in Theorem 3.9 holds for *any* benchmark path $\{\hat{P}_k\} \subset \mathcal{P}$. In particular, HDM is guaranteed to asymptotically achieve the performance of the optimal path that maximizes the algorithm progress. Adaptivity of HDM undergirds its good empirical performance.

Behavior of HDM and Static/Dynamic Adaptivity. We observe that the behavior of HDM can be divided into two stages: (1) when the iterates x^k are far from the optimum $x^\star$, the change in landscape is more drastic and we expect dynamic adaptivity to capture the convergence behavior; (2) when x^k is near the optimum $x^\star$, $f(x)$ locally behaves like a quadratic and P_k remains more stable, and thus static adaptivity describes the convergence behavior.

3.2.3 Online Regret and Instability

Though adaptive P-update underpins the strong performance of HDM, vanilla HDM is observed to be unstable in practice. This section identifies the source of instability in vanilla HDM (Figure 3.1) based on our analysis. We also propose two simple yet effective strategies to address the instability.

Divergence Behavior due to Regret. Recall from Theorem 3.8 that the optimality gap at x^{K+1} is bounded by $\frac{\Delta^2}{K[\gamma_K^\star - \frac{\rho_K}{K}]_+}$. This rate can be better than that of gradient descent when K is large and $\gamma_K^\star \gg \frac{1}{2L}$, but the analysis provides no guarantee on earlier iterates $\{x^k\}_{k\le K}$. In particular, the convergence rate makes sense only if $\gamma_K^\star > \frac{\rho_K}{K}$. That is, the progress $\sum_{k=1}^K h_{x^k}(P_k)$ accumulated by the online gradient descent outweighs its regret ρ_K. In other words, online gradient descent takes time to learn a good preconditioner, and the regret accumulated during this warm-up phase causes HDM to behave as if it is diverging until the progress $\sum_{k=1}^K h_{x^k}(P_k)$ outpaces the regret ρ_K. Since ρ_K grows sublinearly with the iteration count K, HDM will eventually converge. However, the objective value will usually explode and possibly be terminated by the user before convergence begins. Consequently, the two-phase convergence behavior (Figure 3.1a) is rarely observed.

Addressing Instability. While our analysis guarantees HDM will converge eventually, an algorithm that diverges up to 10^{30} before converging is not practical. We propose two simple but effective fixes based on our analysis:

- *Null step.* The x-update is skipped if new iterate increases the objective value (see the first line of Algorithm 7). The null step ensures a monotonic decrease as HDM learns a good preconditioner, although it requires an additional function value oracle call at each iteration. Even on iterations when x^k is not updated, the preconditioner P_k is updated using online gradient descent, so the algorithm is still making progress. In Figure 3.2, the null steps flatten the objective value curve in the divergence phase.
- *Advanced Learning Algorithms.* Better online learning algorithms with lower regret shorten the divergence phase. Figure 3.2 shows a significant speedup when the online gradient descent in Algorithm 7 is replaced by AdaGrad. In our experiments, AdaGrad often improves the robustness of HDM since it does not require pre-specifying algorithm parameters that depend on the total iteration count K and provides convergence guarantees for the earlier iterates $\{x^k\}_{k\le K}$.

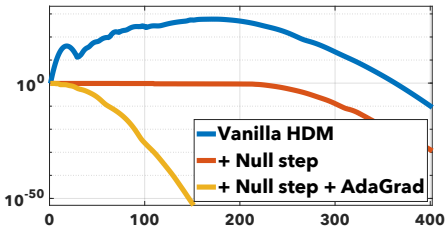


Figure 3.2: Addressing instability of HDM

3.2.4 Local Superlinear Convergence

Figure 3.1 shows HDM converges faster than the (linearly convergent) first-order methods. In fact, HDM exhibits local superlinear convergence on strongly convex objectives (Theorem 3.10 below). This subsection assume a strongly convex objective (**A2**) and Lipschitz Hessian (**A4**):

A4: $f(x)$ has H-Lipschitz Hessian.

Strongly Convex Quadratics. We develop intuition by considering a strongly convex quadratic. For $f(x) = \frac{1}{2}\langle x, Ax\rangle - \langle b, x\rangle$, we have $x^\star = x - [\nabla^2 f(x^\star)]^{-1}\nabla f(x)$. In other words, $P^\star = [\nabla^2 f(x^\star)]^{-1}$ is a universal minimizer of $h_x(P)$ that drives any non-optimal point $x \notin \mathcal{X}^\star$ to the optimum $x^\star$ in one step. When $[\nabla^2 f(x^\star)]^{-1} \in \mathcal{P}$, Theorem 3.8 guarantees the performance of HDM is competitive to $[\nabla^2 f(x^\star)]^{-1}$. Therefore, we expect the descent curve to decrease more and more sharply, giving superlinear convergence (Figure 3.1a).

Local Superlinear Convergence. For general functions satisfying **A4**, $f(x)$ behaves like a quadratic near $x^\star$:

$$f(x) \approx f(x^\star) + \frac{1}{2}\langle x - x^\star, \nabla^2 f(x^\star)(x - x^\star)\rangle.$$

Therefore, local superlinear convergence is expected for HDM near $x^\star$. Theorem 3.10 formalizes this intuition.

Theorem 3.10 (Local superlinear convergence)**.** *Suppose* $[\nabla^2 f(x^\star)]^{-1} \in \mathcal{P}$ *and assume* **A1** *to* **A4**. *Then Algorithm 7 has local superlinear convergence:*

$$f(x^{K+1}) - f(x^\star) \le [f(x^1) - f(x^\star)] \min\left\{\frac{H^2\kappa^2}{4\mu^2 K}\sum_{k=1}^{K}\|x^k - x^\star\|^2 + \frac{2L\rho_K}{K}, 1\right\}^K. \tag{3.20}$$

where ρ_K *is the regret bound defined in* (3.11).

Proof. For Theorem 3.10 and Lemma 3.11 only, we will define the following modified feedback

function by replacing $f(x^k)$ in the numerator by $f(x^\star)$:

$$\hat{h}_x(P) := \frac{f(x - P\nabla f(x)) - f(x^\star)}{\|\nabla f(x)\|^2} \geq 0. \tag{3.21}$$

For a fixed x, $\hat{h}_x(P)$ only differs from the original hypergradient feedback by a constant; it has the same properties as the original feedback function, and the algorithm is exactly the same since only the gradient of $\hat{h}_x$ is considered in the algorithm update. Using (3.21), we deduce that

$$\begin{aligned}
\frac{f(x^{K+1}) - f(x^\star)}{f(x^1) - f(x^\star)} &= \prod_{k=1}^{K} \frac{f(x^{k+1}) - f(x^\star)}{f(x^k) - f(x^\star)} \\
&\leq \left(\frac{1}{K} \sum_{k=1}^{K} \frac{f(x^{k+1}) - f(x^\star)}{f(x^k) - f(x^\star)} \right)^K \\
&= \left(\frac{1}{K} \sum_{k=1}^{K} \min\{ \frac{\hat{h}_{x^k}(P_k)\|\nabla f(x^k)\|^2}{f(x^k) - f(x^\star)}, 1\} \right)^K \qquad (3.22) \\
&\leq \left(\frac{1}{K} \sum_{k=1}^{K} \min\{2L\hat{h}_{x^k}(P_k), 1\} \right)^K \qquad (3.23) \\
&\leq \left(\min\left\{ \frac{2L}{K} \sum_{k=1}^{K} \hat{h}_{x^k}(P_k), 1 \right\} \right)^K,
\end{aligned}$$

where (3.22) plugs in the definition of $\hat{h}_x$; (3.23) uses L-smoothness and that $\hat{h}_x$ is nonnegative. Using Lemma 3.2, we get $\sum_{k=1}^{K} \hat{h}_{x^k}(P_k) \leq \sum_{k=1}^{K} \hat{h}_{x^k}(P) + \rho_K$ for any $P \in \mathcal{P}$. Next, we consider the quantity $\hat{h}_x([\nabla^2 f(x^\star)]^{-1})$ and deduce that

$$\begin{aligned}
\hat{h}_x([\nabla^2 f(x^\star)]^{-1}) &= \frac{f(x - [\nabla^2 f(x^\star)]^{-1}\nabla f(x)) - f(x^\star)}{\|\nabla f(x)\|^2} \\
&\leq \frac{\frac{L}{2}\|x - [\nabla^2 f(x^\star)]^{-1}\nabla f(x) - x^\star\|^2}{\|x - x^\star\|^2} \frac{\|x - x^\star\|^2}{\|\nabla f(x)\|^2} \qquad (3.24) \\
&\leq \frac{L}{2\mu^2} \frac{\|x - [\nabla^2 f(x^\star)]^{-1}\nabla f(x) - x^\star\|^2}{\|x - x^\star\|^2}, \qquad (3.25)
\end{aligned}$$

where (3.24) uses L-smoothness $f(x) - f(x^\star) \leq \frac{L}{2}\|x - x^\star\|^2$ and (3.25) uses $\|\nabla f(x)\|^2 \geq \mu^2\|x - x^\star\|^2$. Then,

$$\begin{aligned}
x - [\nabla^2 f(x^\star)]^{-1}\nabla f(x) - x^\star &= x - x^\star - [\nabla^2 f(x^\star)]^{-1}\nabla f(x) \\
&= [\nabla^2 f(x^\star)]^{-1}[\nabla^2 f(x^\star)(x - x^\star) - (\nabla f(x) - \nabla f(x^\star))]
\end{aligned}$$

since $\nabla f(x^\star) = 0$. Plugging in $\nabla f(x) - \nabla f(x^\star) = \int_0^1 \nabla^2 f(x^\star + t(x - x^\star))(x - x^\star)\mathrm{d}t$, we deduce that

$$\begin{aligned}\|\nabla^2 f(x^\star)(x - x^\star) - (\nabla f(x) - \nabla f(x^\star))\| &= \left\|\nabla^2 f(x^\star)(x - x^\star) - \int_0^1 \nabla^2 f(x^\star + t(x - x^\star))(x - x^\star)\mathrm{d}t\right\| \\ &= \left\|\int_0^1 [\nabla^2 f(x^\star) - \nabla^2 f(x^\star + t(x - x^\star))](x - x^\star)\mathrm{d}t\right\| \\ &\leq \int_0^1 tH\|x - x^\star\|^2\mathrm{d}t = \frac{H}{2}\|x - x^\star\|^2, \end{aligned} \tag{3.26}$$

where (3.26) uses H-Lipschitz continuity of $\nabla^2 f(x)$ and, consequently,

$$\begin{aligned}&\|x - [\nabla^2 f(x^\star)]^{-1}\nabla f(x) - x^\star\| \\ &= \|[\nabla^2 f(x^\star)]^{-1}[\nabla^2 f(x^\star)(x - x^\star) - (\nabla f(x) - \nabla f(x^\star))]\| \leq \frac{H}{2\mu}\|x - x^\star\|^2 \end{aligned} \tag{3.27}$$

since $\nabla^2 f(x^\star) \succeq \mu I$ due to strong convexity. Plugging the relation back, we get

$$\hat{h}_x([\nabla^2 f(x^\star)]^{-1}) \leq \frac{L}{2\mu^2}\frac{\frac{H^2}{4\mu^2}\|x - x^\star\|^4}{\|x - x^\star\|^2} = \frac{H^2\kappa}{8\mu^3}\|x - x^\star\|^2. \tag{3.28}$$

Since $[\nabla^2 f(x^\star)]^{-1} \in \mathcal{P}$ by assumption,

$$\sum_{k=1}^K \hat{h}_{x^k}(P_k) \leq \sum_{k=1}^K \hat{h}_{x^k}([\nabla^2 f(x^\star)]^{-1}) + \rho_K \leq \frac{H^2\kappa}{8\mu^3}\sum_{k=1}^K \|x^k - x^\star\|^2 + \rho_K,$$

and we get

$$f(x^{K+1}) - f(x^\star) \leq [f(x^1) - f(x^\star)]\big(\min\big\{\frac{H^2\kappa^2}{4\mu^2 K}\sum_{k=1}^K \|x^k - x^\star\|^2 + \frac{2L\rho_K}{K}, 1\big\}\big)^K,$$

which completes the proof. □

Theorem 3.10 justifies our observation of superlinear convergence in Figure 3.1a: for strongly convex quadratics, the Hessian Lipschitz constant is zero ($H = 0$) and (3.20) guarantees the superlinear convergence at rate $\mathcal{O}((\frac{\rho_K}{K})^K) = \mathcal{O}((\frac{1}{\sqrt{K}})^K)$. For general strongly convex objectives, global linear convergence (Theorem 3.8) implies $\lim_{K\to\infty}\frac{1}{K}\sum_{k=1}^K \|x^k - x^\star\|^2 = 0$ when $\eta_k = \mathcal{O}(1/\sqrt{k})$. So eventually the first term in (3.20) vanishes, giving superlinear convergence. This superlinear convergence behavior demonstrates that `HDM` can perform significantly better than standard adaptive first-order methods and line search. `HDM` represents a new family of first-order methods that achieves superlinear convergence on strongly convex objectives, following the celebrated quasi-Newton (QN) family [Nocedal and Wright, 1999, Fletcher, 2000]. Table 3.1 summarizes the superlinear convergence rates of `HDM` and other QN methods.

Table 3.1: Recent Superlinear convergence rates

Algorithm	Rate
Greedy QN [Rodomanov and Nesterov, 2021]	$\mathcal{O}(e^{-\frac{1}{2}K^2})$
Broyden family [Rodomanov and Nesterov, 2022]	$\mathcal{O}(e^{-\frac{1}{2}K\log K})$
Online-learning guided QN [Jiang et al., 2023]	$\mathcal{O}(e^{-\frac{1}{2}K\log K})$
BFGS with line-search [Jin et al., 2024a,b]	$\mathcal{O}(e^{-K\log K})$
`HDM` (This chapter)	$\mathcal{O}(e^{-\frac{1}{2}K\log K})$

`HDM` Learns the Hessian at the Optimum. In fact, $\{P_k\}$ in `HDM` will converge to $[\nabla^2 f(x^\star)]^{-1}$ under an assumption similar to one studied in the quasi-Newton literature Conn et al. [1991], Nocedal and Wright [1999]. Lemma 3.11 quantifies the effect of learning the preconditioner through the distance $\|P_k - [\nabla^2 f(x^\star)]^{-1}\|_F$.

Lemma 3.11. *Under the same assumptions as Theorem 3.10, Algorithm 7 generates $\{P_k\}$ such that*

$$\begin{aligned}&\|P_{k+1} - [\nabla^2 f(x^\star)]^{-1}\|_F^2 \\ &\le \|P_k - [\nabla^2 f(x^\star)]^{-1}\|_F^2 - \frac{\mu(\eta - L\eta^2)}{2}\left\|(P_k - [\nabla^2 f(x^\star)]^{-1})\tfrac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\|^2 + (2\eta - L\eta^2)\frac{H^2\kappa}{4\mu^3}\|x^k - x^\star\|^2.\end{aligned} \tag{3.29}$$

Proof. Let $P^\star = [\nabla^2 f(x^\star)]^{-1}$ for brevity and let $\hat{h}_x(P)$ be the modified feedback defined in (3.21). The update of online gradient descent implies

$$\begin{aligned}
&\|P_{k+1} - P^\star\|_F^2 && (3.30)\\
&= \|\Pi_{\mathcal{P}}[P_k - \eta\nabla\hat{h}_{x^k}(P_k) - P^\star]\|_F^2 \\
&\le \|P_k - \eta\nabla\hat{h}_{x^k}(P_k) - P^\star\|_F^2 \\
&= \|P_k - P^\star\|_F^2 - 2\eta\langle\nabla\hat{h}_{x^k}(P_k), P_k - P^\star\rangle + \eta^2\|\nabla\hat{h}_{x^k}(P_k)\|_F^2 \\
&\le \|P_k - P^\star\|_F^2 - 2\eta[\hat{h}_{x^k}(P_k) - \hat{h}_{x^k}(P^\star)] + 2L\eta^2[\hat{h}_{x^k}(P_k) - \inf_{P\in\mathbb{R}^{n\times n}}\hat{h}_{x^k}(P)] && (3.31)\\
&= \|P_k - P^\star\|_F^2 - 2\eta[\hat{h}_{x^k}(P_k) - \hat{h}_{x^k}(P^\star)] + 2L\eta^2[\hat{h}_{x^k}(P_k) - \hat{h}_{x^k}(P^\star)] + 2L\eta^2\hat{h}_{x^k}(P^\star) && (3.32)\\
&= \|P_k - P^\star\|_F^2 - 2\eta(1-\eta L)[\hat{h}_{x^k}(P_k) - \hat{h}_{x^k}(P^\star)] + 2L\eta^2\hat{h}_{x^k}(P^\star), && (3.33)
\end{aligned}$$

where (3.31) uses L-smoothness and $\inf_{P\in\mathbb{R}^{n\times n}}\hat{h}_x(P) = 0$ for all $x \notin \mathcal{X}^\star$; (3.32) is a simple rearrangement.

Next we lower bound $\hat{h}_{x^k}(P_k) - \hat{h}_{x^k}(P^\star)$. Using strong convexity,

$$\begin{aligned}
&f(x^k - P_k\nabla f(x^k)) - f(x^k - P^\star\nabla f(x^k)) \\
&= f(x^k - P_k\nabla f(x^k)) - f(x^\star) + f(x^\star) - f(x^k - P^\star\nabla f(x^k)) \\
&\geq \frac{\mu}{2}\|x^k - x^\star - P_k\nabla f(x^k)\|^2 + f(x^\star) - f(x^k - P^\star\nabla f(x^k)), \qquad (3.34)
\end{aligned}$$

where (3.34) uses $f(x) - f(x^\star) \geq \frac{\mu}{2}\|x - x^\star\|^2$. The first term can be bounded as follows:

$$\begin{aligned}
&\|x^k - x^\star - P_k\nabla f(x^k)\|^2 \\
&= \|x^k - P^\star\nabla f(x^k) - x^\star + (P^\star - P_k)\nabla f(x^k)\|^2 \\
&= \|x^k - P^\star\nabla f(x^k) - x^\star\|^2 + 2\langle x^k - P^\star\nabla f(x^k) - x^\star, (P^\star - P_k)\nabla f(x^k)\rangle + \|(P^\star - P_k)\nabla f(x^k)\|^2 \\
&\geq \frac{1}{2}\|(P^\star - P_k)\nabla f(x^k)\|^2 - \|x^k - P^\star\nabla f(x^k) - x^\star\|^2,
\end{aligned}$$

where we use the inequality $2\langle a, b\rangle \geq -\theta\|a\|^2 - \theta^{-1}\|b\|^2$ with $\theta = 2$. Plugging the relation back into (3.34) and dividing both sides by $\|\nabla f(x^k)\|^2$,

$$\begin{aligned}
&\hat{h}_{x^k}(P_k) - \hat{h}_{x^k}(P^\star) \\
&= \frac{f(x^k - P_k\nabla f(x^k)) - f(x^\star) + f(x^\star) - f(x^k - P^\star\nabla f(x^k))}{\|\nabla f(x^k)\|^2} \\
&\geq \frac{\mu}{4}\left\|(P^\star - P_k)\tfrac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\|^2 - \frac{\mu}{2}\frac{\|x^k - P^\star\nabla f(x^k) - x^\star\|^2}{\|\nabla f(x^k)\|^2} + \frac{f(x^\star) - f(x^k - P^\star\nabla f(x^k))}{\|\nabla f(x^k)\|^2} \\
&= \frac{\mu}{4}\left\|(P^\star - P_k)\tfrac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\|^2 - \frac{\mu}{2}\frac{\|x^k - P^\star\nabla f(x^k) - x^\star\|^2}{\|\nabla f(x^k)\|^2} - \hat{h}_{x^k}(P^\star) \qquad (3.35) \\
&\geq \frac{\mu}{4}\left\|(P^\star - P_k)\tfrac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\|^2 - \frac{H^2}{8\mu}\frac{\|x^k - x^\star\|^4}{\|\nabla f(x^k)\|^2} - \hat{h}_{x^k}(P^\star) \qquad (3.36) \\
&\geq \frac{\mu}{4}\left\|(P^\star - P_k)\tfrac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\|^2 - \frac{H^2\kappa}{8\mu^3}\|x^k - x^\star\|^2 - \hat{h}_{x^k}(P^\star), \qquad (3.37)
\end{aligned}$$

where (3.35) uses the definition of $\hat{h}_{x^k}$; (3.36) applies the relation $\|x - P^\star\nabla f(x) - x^\star\| \leq \frac{H}{2\mu}\|x - x^\star\|^2$ from (3.27); (3.37) again uses the fact $\|\nabla f(x)\|^2 \geq \mu^2\|x - x^\star\|^2$. Putting the relations back into

(3.33) and assuming $\eta \leq \frac{1}{2L}$,

$$
\begin{aligned}
&\|P_{k+1} - P^\star\|_F^2 \\
&\leq \|P_k - P^\star\|_F^2 - \frac{\mu(\eta - L\eta^2)}{2} \left\|(P_k - P^\star)\tfrac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\|^2 \\
&\quad + \frac{H^2\kappa(\eta - L\eta^2)}{4\mu^3}\|x^k - x^\star\|^2 + 2(\eta - L\eta^2)\hat{h}_{x^k}(P^\star) + 2L\eta^2\hat{h}_{x^k}(P^\star) \\
&= \|P_k - P^\star\|_F^2 - \frac{\mu(\eta - L\eta^2)}{2} \left\|(P_k - P^\star)\tfrac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\|^2 + \frac{H^2\kappa(\eta - L\eta^2)}{4\mu^3}\|x^k - x^\star\|^2 + 2\eta\hat{h}_{x^k}(P^\star) \\
&\overset{(3.28)}{\leq} \|P_k - P^\star\|_F^2 - \frac{\mu(\eta - L\eta^2)}{2} \left\|(P_k - P^\star)\tfrac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\|^2 + \frac{H^2\kappa(\eta - L\eta^2)}{4\mu^3}\|x^k - x^\star\|^2 + 2\eta\frac{H^2\kappa}{8\mu^3}\|x^k - x^\star\|^2 \\
&= \|P_k - P^\star\|_F^2 - \frac{\mu(\eta - L\eta^2)}{2} \left\|(P_k - P^\star)\tfrac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\|^2 + (2\eta - L\eta^2)\frac{H^2\kappa}{4\mu^3}\|x^k - x^\star\|^2.
\end{aligned}
$$

This completes the proof. □

Relation (3.29) consists of three terms: the distance $\|P_k - [\nabla^2 f(x^\star)]^{-1}\|_F^2$; a decrement in the distance; and an error term that converges to zero as $x^k \to x^\star$. The decrement is determined by the magnitude of $\left\|(P_k - [\nabla^2 f(x^\star)]^{-1})\frac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\|^2$, which measures the difference between the operators P_k and $[\nabla^2 f(x^\star)]^{-1}$ in the (unit) gradient direction $\frac{\nabla f(x^k)}{\|\nabla f(x^k)\|}$. To ensure fast convergence, it suffices for $P_k\nabla f(x^k)$ and $[\nabla^2 f(x^\star)]^{-1}\nabla f(x^k)$ to remain sufficiently close. If the set $\left\{\frac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\}$ spans the entire space over the iterations, P_k and $[\nabla^2 f(x^\star)]^{-1}$ should align in all directions, leading to convergence of $\{P_k\}$. Theorem 3.12 below formalizes the statement. The proof is long and is deferred to Appendix B.1.

Theorem 3.12 (Convergence of the preconditioner)**.** *Instate the same assumptions as in Lemma 3.11 and let* $\eta_k \equiv \eta \in (0, \frac{1}{2L(LD+1)^2\kappa}]$ *in online gradient descent* (3.10)*. Suppose the gradient directions* $\left\{\frac{\nabla f(x^k)}{\|\nabla f(x^k)\|}\right\}$ *are uniformly independent* [1]*. Then* $\lim_{k\to\infty}\|P_k - [\nabla^2 f(x^\star)]^{-1}\| = 0$.

The convergence of stepsize in `HDM` was observed experimentally by Baydin et al. [2018] for a scalar stepsize. ($\mathcal{P} \subseteq \mathcal{S}$). Our result theoretically justifies this observation.

`HDM` and Quasi-Newton Methods. Our results identify a similarity between `HDM` and quasi-Newton methods. Both learn the inverse Hessian operator $g \mapsto [\nabla^2 f(x^\star)]^{-1}g$ as the algorithm progresses, but through different properties of the operator. The quasi-Newton methods use the secant equation $x - y \approx [\nabla^2 f(x^\star)]^{-1}(\nabla f(x) - \nabla f(y))$ for x, y close to $x^\star$ and enforce this equation, replacing the inverse Hessian by P_k, to guide learning Jiang et al. [2023]. In contrast, `HDM` learns an optimal preconditioner for the function. Since the function is locally quadratic, this optimal preconditioner is the inverse Hessian. `HDM` uses the hypergradient feedback $h_x(P)$ to directly measure

[1] The formal definition of a uniformly independent sequence is given in Appendix B.1, which is adapted from quasi-Newton literature Conn et al. [1991], Nocedal and Wright [1999]

the quality of the preconditioner and can search for an optimal preconditioner in a given closed convex set $\mathcal{P}$, whereas quasi-Newton methods use the secant equation as an indirect proxy. Both approaches require a safeguard to prevent divergence in the warm-up phase, which is achieved by line-search in quasi-Newton and null step in HDM.

3.3 HDM with Heavy-Ball Momentum

In this section, we develop the variant of HDM with heavy-ball momentum Polyak [1964]. Heavy-ball method is a practical acceleration technique:

$$x^{k+1} = x^k - P_k \nabla f(x^k) + B_k(x^k - x^{k-1}). \tag{3.38}$$

The momentum parameter B_k is typically chosen as a scalar $B_k = \beta_k I$ with $\beta_k > 0$. HDM can learn a matrix momentum $B_k \in \mathcal{B} \subseteq \mathbb{R}^{n\times n}$ with convergence guarantees (Theorem 3.16) when $\mathcal{B}$ satisfies this assumption:

A5: Closed convex set $\mathcal{B}$ satisfies $\frac{1}{2}I \in \mathcal{B}$, $\operatorname{diam}(\mathcal{B}) \le D$.

HDM can *jointly* learn the pair (P_k, B_k) using the modified feedback function

$$h_{x,x^-}(P,B) := \frac{\psi(x^+(P,B),x) - \psi(x,x^-)}{\|\nabla f(x)\|^2 + \frac{\tau}{2}\|x - x^-\|^2}, \tag{3.39}$$

where ψ is the potential function for heavy-ball momentum defined by $\psi(x,x^-) := f(x) + \frac{\omega}{2}\|x - x^-\|^2$ Danilova et al. [2020]; $x^+(P,B) := x - P\nabla f(x) + B(x - x^-)$ updates x; and $\omega > 0$ and $\tau > 0$ are constants. Algorithm 8 presents the resulting method, HDM-HB, which uses HDM, heavy-ball momentum, and a null step to ensure decrease of the potential function ψ. Figure 3.3 compares non-adaptive heavy-ball ($P_k \equiv \alpha I, B_k \equiv \beta I$) against HDM-HB with full-matrix/diagonal preconditioner and scalar momentum.

Algorithm 8 HDM with heavy-ball momentum (HDM-HB)

input initial point $x^0 = x^1$, $\eta_p, \eta_b > 0$, P_1, B_1
for $k = 1, 2, \ldots$ **do**
  $x^{k+1/2} = x^k - P_k \nabla f(x^k) + B_k(x^k - x^{k-1})$
  $P_{k+1} = \Pi_{\mathcal{P}}[P_k - \eta_p \nabla_P h_{x^k, x^{k-1}}(P_k, B_k)]$
  $B_{k+1} = \Pi_{\mathcal{B}}[B_k - \eta_b \nabla_B h_{x^k, x^{k-1}}(P_k, B_k)]$
  $(x^{k+1}, x^k) = \underset{(x^+,x)\in\{(x^k,x^{k-1}),\,(x^{k+1/2},x^k)\}}{\arg\min} \psi(x^+, x)$
end for
output x^{K+1}

3.3.1 HDM + Heavy-ball Momentum (HDM-HB)

Algorithm 8 uses the following *heavy-ball feedback function* to guide the online learning for (P_k, B_k):

$$h_{x,x^-}(P,B) := \frac{\psi(x^+,x)-\psi(x,x^-)}{\|\nabla f(x)\|^2+\frac{\tau}{2}\|x-x^-\|^2} = \frac{[f(x^+)+\frac{\omega}{2}\|x^+-x\|^2]-[f(x)+\frac{\omega}{2}\|x-x^-\|^2]}{\|\nabla f(x)\|^2+\frac{\tau}{2}\|x-x^-\|^2},$$

where $\omega > 0$, $\tau > 0$, $x^+ = x - P\nabla f(x) + B(x-x^-)$, and $\psi(x,x^-) = f(x) + \frac{\omega}{2}\|x-x^-\|^2$. To show that online learning can be applied to $h_{x,x^-}(P,B)$ with regret guarantees, we need to verify the convexity and Lipschitz continuity of $h_{x,x^-}(P,B)$ with respect to the norm defined by

$$\|(P,B)\| := \sqrt{\|P\|_F^2+\|B\|_F^2}. \tag{3.40}$$

Lemma 3.13. *Under* **A1**, **A3**, *and* **A5**, *the heavy-ball feedback function $h_{x,x^-}(P,B)$ is jointly convex in (P,B) and c-Lipschitz with respect to the norm defined in* (3.40), *where $c := \sqrt{2}(1+\frac{2}{\tau})[1+2(1+\frac{2}{\tau})D(L+\omega)]$.*

Proof. Denote $x^+(P,B) := x - P\nabla f(x) + B(x-x^-)$. Recall that the feedback function is

$$h_{x,x^-}(P,B) = \frac{[f(x^+(P,\beta))+\frac{\omega}{2}\|x^+(P,\beta)-x\|^2]-[f(x)+\frac{\omega}{2}\|x-x^-\|^2]}{\|\nabla f(x)\|^2+\frac{\tau}{2}\|x-x^-\|^2}.$$

Since $x^+(P,B)$ is affine in (P,B) and f is convex, the term $f(x^+(P,\beta))+\frac{\omega}{2}\|x^+(P,\beta)-x\|^2$ is jointly convex as a function of (P,B). The other terms in the feedback function $h_{x,x^-}(P,B)$ are constants, so $h_{x,x^-}(P,B)$ is also jointly convex in (P,B).

To prove the Lipschitz continuity of $h_{x,x^-}(P,B)$, it suffices to show that the gradients of $h_{x,x^-}(P,B)$ are bounded. The gradients of $h_{x,x^-}(P,B)$ with respect to P and B are

$$\nabla_P h_{x,x^-}(P,B) = \frac{[-\nabla f(x^+(P,B))+\omega P\nabla f(x)-\omega B(x-x^-)]\nabla f(x)^\top}{\|\nabla f(x)\|^2+\frac{\tau}{2}\|x-x^-\|^2},$$
$$\nabla_B h_{x,x^-}(P,B) = \frac{[\nabla f(x^+(P,B))-\omega P\nabla f(x)+\omega B(x-x^-)](x-x^-)^\top}{\|\nabla f(x)\|^2+\frac{\tau}{2}\|x-x^-\|^2}.$$

Using the fact $\|ab^\top\|_F = \|a\|\cdot\|b\|$, the gradients have norms

$$\|\nabla_P h_{x,x^-}(P,B)\|_F = \frac{\|\nabla f(x^+(P,B))-\omega P\nabla f(x)+\omega B(x-x^-)\|\|\nabla f(x)\|}{\|\nabla f(x)\|^2+\frac{\tau}{2}\|x-x^-\|^2}, \tag{3.41}$$
$$\|\nabla_B h_{x,x^-}(P,B)\|_F = \frac{\|\nabla f(x^+(P,B))-\omega P\nabla f(x)+\omega B(x-x^-)\|\|x-x^-\|}{\|\nabla f(x)\|^2+\frac{\tau}{2}\|x-x^-\|^2}. \tag{3.42}$$

Using **A1**, we have the Lipschitz continuity of $\nabla f(x)$ and thus

$$\begin{aligned}
&\|\nabla f(x^+(P,B)) - \omega P\nabla f(x) + \omega B(x - x^-)\| \\
&\le \|\nabla f(x^+(P,B)) - \nabla f(x)\| + \|(I - \omega P)\nabla f(x)\| + \omega\|B\|\|x - x^-\| \\
&\le L\|P\nabla f(x) - B(x - x^-)\| + (1 + \omega\|P\|)\|\nabla f(x)\| + \omega\|B\|\|x - x^-\| \\
&\le LD(\|\nabla f(x)\| + \|x - x^-\|) + (1 + \omega D)\|\nabla f(x)\| + \omega D\|x - x^-\| \\
&= (1 + LD + \omega D)\|\nabla f(x)\| + (\omega + L)D\|x - x^-\|. \qquad (3.43)
\end{aligned}$$

Now, we bound the norms in (3.41)–(3.42) by the case analysis.

Case 1. If $\frac{\tau}{2}\|x - x^-\|^2 \le \|\nabla f(x)\|^2$, then together with (3.43), we have

$$\begin{aligned}
&\max\{\|\nabla_P h_{x,x^-}(P,B)\|_F, \|\nabla_B h_{x,x^-}(P,B)\|_F\} \\
&\le \frac{[(1 + LD + \omega D)\|\nabla f(x)\| + (\omega + L)D\|x - x^-\|] \max\left\{\sqrt{2\tau^{-1}}, 1\right\}\|\nabla f(x)\|}{\|\nabla f(x)\|^2} \\
&\le \max\left\{\sqrt{2\tau^{-1}}, 1\right\}\left[(1 + LD + \omega D) + \frac{\sqrt{2}D(\omega + L)}{\sqrt{\tau}}\right] \\
&= \max\left\{\sqrt{2\tau^{-1}}, 1\right\}(1 + D(L + \omega)(1 + \sqrt{2\tau^{-1}})).
\end{aligned}$$

Case 2. If $\frac{\tau}{2}\|x - x^-\|^2 \ge \|\nabla f(x)\|^2$, then

$$\begin{aligned}
&\max\{\|\nabla_P h_{x,x^-}(P,B)\|_F, \|\nabla_B h_{x,x^-}(P,B)\|_F\} \\
&\le \frac{[(1 + LD + \omega D)\|\nabla f(x)\| + (\omega + L)D\|x - x^-\|] \max\left\{\sqrt{\frac{\tau}{2}}, 1\right\}\|x - x^-\|}{\frac{\tau}{2}\|x - x^-\|^2} \\
&\le \max\{\sqrt{\tau/2}, 1\}\left[\frac{\sqrt{2}(1 + LD + \omega D)}{\sqrt{\tau}} + \frac{2D(\omega + L)}{\tau}\right] \\
&= \sqrt{2\tau^{-1}}\max\{\sqrt{2\tau^{-1}}, 1\}(1 + D(L + \omega)(1 + \sqrt{2\tau^{-1}})).
\end{aligned}$$

Combining the two cases, we have

$$\begin{aligned}
\max\{\|\nabla_P h_{x,x^-}(P,B)\|_F, \|\nabla_B h_{x,x^-}(P,B)\|_F\} &\le \max\left\{\frac{2}{\tau}, 1\right\}(1 + D(L + \omega)(1 + \sqrt{2\tau^{-1}})) \\
&\le (1 + \tfrac{2}{\tau})[1 + 2(1 + \tfrac{2}{\tau})D(L + \omega)].
\end{aligned}$$

Then the gradient of $h_{x,x^-}(P,B)$ under the norm defined in (3.40) is bounded by the constant $c := \sqrt{2}(1 + \frac{2}{\tau})[1 + 2(1 + \frac{2}{\tau})D(L + \omega)]$. □

The next lemma bounds the potential at the last iterate x^{K+1} from Algorithm 8 in terms of the

sum of feedback functions $h_{x^k,x^{k-1}}(P_k, B_k)$.

Lemma 3.14. *The sequence $\{x^k\}$ generated from Algorithm 8 satisfies*

$$f(x^{K+1}) - f(x^\star) + \frac{\omega}{2}\|x^{K+1} - x^K\|^2 \leq \frac{f(x^1) - f(x^\star)}{1 + \sum_{k=1}^K \max\{-h_{x^k,x^{k-1}}(P_k, B_k), 0\}V}, \tag{3.44}$$

where $V := \min\left\{\frac{f(x^1)-f(x^\star)}{4\Delta^2}, \frac{\tau}{4\omega}\right\}$ and $\Delta := \max_{x\in\mathcal{L}_{f(x^1)}} \min_{x^\star\in\mathcal{X}^\star} \|x - x^\star\|$.

Proof. The null step guarantees

$$\frac{\psi(x^{k+1}, x^k) - \psi(x^k, x^{k-1})}{\|\nabla f(x^k)\|^2 + \frac{\tau}{2}\|x^k - x^{k-1}\|^2} = \min\{h_{x^k,x^{k-1}}(P_k, B_k), 0\}.$$

Using the initial condition $x^1 = x^0$, we have

$$\begin{aligned}
\psi(x^{K+1}, x^K) - f(x^\star) &= \frac{1}{\frac{1}{\psi(x^{K+1},x^K)-f(x^\star)}} \\
&= \frac{1}{\sum_{k=1}^K \frac{1}{\psi(x^{k+1},x^k)-f(x^\star)} - \frac{1}{\psi(x^k,x^{k-1})-f(x^\star)} + \frac{1}{\psi(x^1,x^0)-f(x^\star)}} \\
&= \frac{1}{\sum_{k=1}^K \frac{\psi(x^k,x^{k-1})-\psi(x^{k+1},x^k)}{[\psi(x^{k+1},x^k)-f(x^\star)][\psi(x^k,x^{k-1})-f(x^\star)]} + \frac{1}{\psi(x^1,x^0)-f(x^\star)}} \\
&= \frac{1}{\sum_{k=1}^K \frac{\max\{-h_{x^k,x^{k-1}}(P_k,B_k),0\}[\|\nabla f(x^k)\|^2+\frac{\tau}{2}\|x^k-x^{k-1}\|^2]}{[\psi(x^{k+1},x^k)-f(x^\star)][\psi(x^k,x^{k-1})-f(x^\star)]} + \frac{1}{f(x^1)-f(x^\star)}}.
\end{aligned} \tag{3.45}$$

Then, by monotonicity, $\frac{\|\nabla f(x^k)\|^2+\frac{\tau}{2}\|x^k-x^{k-1}\|^2}{[\psi(x^{k+1},x^k)-f(x^\star)][\psi(x^k,x^{k-1})-f(x^\star)]} \geq \frac{\|\nabla f(x^k)\|^2+\frac{\tau}{2}\|x^k-x^{k-1}\|^2}{[\psi(x^k,x^{k-1})-f(x^\star)]^2}$.

Now we do case analysis to bound

$$\frac{\|\nabla f(x^k)\|^2 + \frac{\tau}{2}\|x^k - x^{k-1}\|^2}{[\psi(x^k, x^{k-1}) - f(x^\star)]^2} = \frac{\|\nabla f(x^k)\|^2 + \frac{\tau}{2}\|x^k - x^{k-1}\|^2}{[f(x^k) + \frac{\omega}{2}\|x^k - x^{k-1}\|^2 - f(x^\star)]^2}$$

Case 1. If $\frac{\omega}{2}\|x^k - x^{k-1}\|^2 \leq f(x^k) - f(x^\star)$, then

$$\frac{\|\nabla f(x^k)\|^2 + \frac{\tau}{2}\|x^k - x^{k-1}\|^2}{[f(x^k) + \frac{\omega}{2}\|x^k - x^{k-1}\|^2 - f(x^\star)]^2} \geq \frac{\|\nabla f(x^k)\|^2}{4[f(x^k) - f(x^\star)]^2} \geq \frac{1}{4\Delta^2},$$

where $\Delta := \max_{x\in\mathcal{L}_{f(x^1)}} \min_{x^\star\in\mathcal{X}^\star} \|x - x^\star\|$.

Case 2. If $\frac{\omega}{2}\|x^k - x^{k-1}\|^2 \geq f(x^k) - f(x^\star)$, then $\frac{\tau}{2}\|x^k - x^{k-1}\|^2 \geq \frac{\tau}{\omega}[f(x^k) - f(x^\star)]$ and

$$\frac{\|\nabla f(x^k)\|^2 + \frac{\tau}{2}\|x^k - x^{k-1}\|^2}{[f(x^k) + \frac{\omega}{2}\|x^k - x^{k-1}\|^2 - f(x^\star)]^2} \geq \frac{\frac{\tau}{2}\|x^k - x^{k-1}\|^2}{\omega^2\|x^k - x^{k-1}\|^4} = \frac{\tau}{2\omega^2}\frac{1}{\|x^k - x^{k-1}\|^2} \geq \frac{\tau}{4\omega}\frac{1}{f(x^1) - f(x^\star)}.$$

since $\frac{\omega}{2}\|x^k - x^{k-1}\|^2 \leq \psi(x^k, x^{k-1}) - f(x^\star) \leq \psi(x^1, x^0) - f(x^\star) = f(x^1) - f(x^\star)$.

In both cases, we have $\frac{\|\nabla f(x^k)\|^2 + \frac{\tau}{2}\|x^k - x^{k-1}\|^2}{[\psi(x^k, x^{k-1}) - f(x^\star)]^2} \geq \min\{\frac{1}{4\Delta^2}, \frac{\tau}{4\omega}\frac{1}{f(x^1)-f(x^\star)}\} = \frac{V}{f(x^1)-f(x^\star)}$, where the constant V is defined in the lemma. Finally, plugging in the definition of ψ, (3.45) gives

$$f(x^{K+1}) - f(x^\star) + \frac{\omega}{2}\|x^{K+1} - x^K\|^2 \leq \frac{f(x^1) - f(x^\star)}{1 + \sum_{k=1}^K \max\{-h_{x^k, x^{k-1}}(P_k, B_k), 0\}V}.$$

□

Next lemma shows that there exist hindsight $\bar{P}, \bar{B}$ such that $h_{x,x^-}(\bar{P}, \bar{B}) \leq -\theta < 0$ for some θ.

Lemma 3.15. *Let $\omega = 3L$ and $\tau = 16L^2$. Then for any $x, x^- \notin \mathcal{X}^\star$, we have $h_{x,x^-}(\frac{1}{4L}I, \frac{1}{2}I) \leq -\frac{1}{8L}$. In particular, if $\frac{1}{4L}I \in \mathcal{P}$, $\frac{1}{2}I \in \mathcal{B}$, and $\{x^k\}_{k=1}^K \cap \mathcal{X}^\star = \varnothing$, then*

$$\gamma_K^\star := -\min_{(P,B)\in\mathcal{P}\times\mathcal{B}} \frac{1}{K}\sum_{k=1}^K h_{x^k, x^{k-1}}(P, B) \geq \frac{1}{8L}.$$

Proof. When $P = \alpha I$ and $B = \beta I$ for some $\alpha, \beta > 0$, the classical analysis for the heavy-ball momentum [Danilova et al., 2020] gives

$$f(x^+) + \frac{1 - \alpha L}{2\alpha}\|x^+ - x\|^2 \leq f(x) + \frac{\beta^2}{2\alpha}\|x - x^-\|^2 - \frac{\alpha}{2}\|\nabla f(x)\|^2.$$

Let $\alpha = \frac{1}{4L}$ and $\beta = \frac{1}{2}$, we have

$$\begin{aligned} f(x^+) + \frac{3L}{2}\|x^+ - x\|^2 &\leq f(x) + \frac{L}{2}\|x - x^-\|^2 - \frac{1}{8L}\|\nabla f(x)\|^2 \\ &= f(x) + \frac{3L}{2}\|x - x^-\|^2 - \frac{1}{8L}\|\nabla f(x)\|^2 - L\|x - x^-\|^2 \\ &= f(x) + \frac{3L}{2}\|x - x^-\|^2 - \frac{1}{8L}[\|\nabla f(x)\|^2 + 8L^2\|x - x^-\|^2] \end{aligned}$$

and re-arranging the terms, we get

$$\frac{f(x^+) + \frac{3L}{2}\|x^+ - x\|^2 - [f(x) + \frac{3L}{2}\|x - x^-\|^2]}{\|\nabla f(x)\|^2 + 8L^2\|x - x^-\|^2} \leq -\frac{1}{8L}$$

and this completes the proof. □

Theorem 3.16 (Convergence of `HDM-HB`)**.** *Under* **A1**, **A3** *and* **A5**, *Algorithm 8 with* $\eta_b, \eta_p = \mathcal{O}(1/\sqrt{K})$ *or* $\mathcal{O}(1/\sqrt{k})$ *satisfies*

$$f(x^{K+1}) - f(x^\star) \leq \frac{f(x^1) - f(x^\star)}{KV \max\{\gamma_K^\star - \frac{\rho_K}{K}, 0\} + 1},$$

where $\gamma_K^\star := -\min_{(P,B)\in\mathcal{P}\times\mathcal{B}} \frac{1}{K}\sum_{k=1}^K h_{x^k,x^{k-1}}(P,B)$ *depends on the iteration trajectory* $\{x^k\}_{k\leq K}$*;* $\rho_K = \mathcal{O}(\sqrt{K})$ *is the regret with respect to feedback* (3.39)*;* $V := \min\left\{\frac{f(x^1)-f(x^\star)}{4\Delta^2}, \frac{\tau}{4\omega}\right\}$*;* Δ *is defined in Lemma 3.5.*

Proof. By Lemma 3.13, the heavy-ball feedback is convex and Lipschitz, and thus the same proof of Lemma 3.2 guarantees that online gradient descent

$$(P_{k+1}, B_{k+1}) = \Pi_{\mathcal{P}\times\mathcal{B}}[(P_k, B_k) - \eta\nabla h_{x^k,x^{k-1}}(P_k, B_k)]$$

(with $\eta_p = \eta_b = \eta$) gives the regret bound

$$\frac{1}{K}\sum_{k=1}^K -h_{x^k,x^{k-1}}(P_k, B_k) \geq \gamma_K^\star - \frac{\rho_K}{K}$$

for some sublinear regret $\rho_K = \mathcal{O}(\sqrt{K})$ and the constant $\gamma_K^\star$ as defined in Lemma 3.15. Using the inequality

$$\frac{1}{K}\sum_{k=1}^K \max\{-h_{x^k,x^{k-1}}(P_k, B_k), 0\} \geq \max\left\{\frac{1}{K}\sum_{k=1}^K -h_{x^k,x^{k-1}}(P_k, B_k), 0\right\} \geq \max\left\{\gamma_K^\star - \frac{\rho_K}{K}, 0\right\},$$

the desired result follows directly from (3.44) in Lemma 3.14. □

3.4 Practical Variant of `HDM` and Numerical Experiments

This section conducts numerical experiments to validate the empirical performance of hypergradient descent. We compare `HDM-Best` (see Section 3.4.1 below) with different adaptive optimization algorithms.

3.4.1 Efficient and Practical Variant: `HDM-Best`

This section highlights the major components of our most competitive variant `HDM-Best`. This variant is adapted from `HDM-HB`, with simplifications to reduce the implementation complexity. The algorithm is given in Algorithm 9.

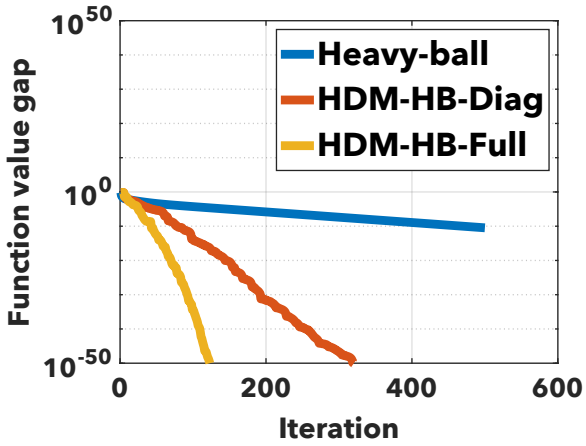


Figure 3.3: The convergence behavior of `HDM-HB` on a toy quadratic problem.

Diagonal Preconditioner and Heavy-ball Momentum. `HDM-Best` updates x by (3.38) with diagonal preconditioner $\mathcal{P} \subseteq \mathcal{D}$ and scalar momentum $\mathcal{B} = \{\beta I : \beta \in \mathbb{R}\}$. This choice balances practical efficiency and implementation complexity. Boundedness of $\mathcal{P}$ does not greatly impact the performance, while the bound on $\mathcal{B}$ can significantly change algorithm behavior. Two empirically robust ranges for $\mathcal{B}$ are $[0, 0.9995]$ and $[-0.9995, 0.9995]$.

`AdaGrad` for Online Learning. `HDM-Best` uses `AdaGrad` to shorten the warm-up phase for learning of (P_k, β_k) (see Section 3.2.3). `AdaGrad` usually yields faster convergence of `HDM` than online gradient descent at the cost of additional memory of size n. The implementation is available at `https://github.com/udellgroup/hypergrad`. We make several remarks about Algorithm 9.

- *Choice of online learning algorithm.* Unless $f(x)$ is quadratic, adaptive online learning algorithms such as `AdaGrad` often significantly outperform online gradient descent with constant stepsize. Note that `AdaGrad` introduces additional memory of size n to store the diagonal online learning preconditioner U.

- *Sensitivity of parameters.* The two stepsize parameters in `AdaGrad` are the most important algorithm parameters: η_p, η_b. According to the experiments, η_p should be set proportional to $1/L$, the smoothness constant, while an aggressive choice of $\eta_b \in \{1, 10, 100\}$ often yields fast convergence. A local estimator of the smoothness constant L can significantly enhance algorithm performance.

- *Heavy-ball feedback and null step.* In practice, it is observed that dropping the $\frac{\omega}{2}\|x^+(P,B) - x\|^2$ in the numerator of heavy-ball feedback (3.39) often does not affect algorithm performance. Therefore, in Algorithm 9 the hypergradient with respect to $\frac{\omega}{2}\|x^+(P,B) - x\|^2$ is ignored. On the other hand, the $\frac{\tau}{2}\|x^+(P,B) - x\|^2$ term in the denominator smoothes the update of β_k and can strongly affect convergence. The parameter τ should be taken to be proportional to L^2 according

Algorithm 9 `HDM-Best`

input starting point $x^0 = x^1$; $\mathcal{P} = \mathbb{S}^n_+ \cap \mathcal{D}$, $\mathcal{B} = [0, 0.9995]$; initial diagonal preconditioner $P_1 \in \mathbb{S}^n_+ \cap \mathcal{D}$; initial scalar momentum $\beta_1 = 0.95$; `AdaGrad` stepsizes $\eta_p, \eta_b > 0$; `AdaGrad` diagonal matrix $U_1 = 0$; `AdaGrad` momentum scalar $v_1 = 0$; $\tau > 0$
for $k = 1, 2, \ldots$ **do**
  $x^{k+1/2} = x^k - P_k \nabla f(x^k) + \beta_k (x^k - x^{k-1})$
  $\nabla_P h_{x^k, x^{k-1}}(P_k, \beta_k) = \dfrac{\operatorname{diag}(\nabla f(x^{k+1/2}) \circ \nabla f(x^k))}{\|\nabla f(x^k)\|^2 + \frac{\tau}{2}\|x^k - x^{k-1}\|^2}$ ▷ element-wise product
  $\nabla_\beta h_{x^k, x^{k-1}}(P_k, \beta_k) = \dfrac{\langle \nabla f(x^{k+1/2}),\, x^k - x^{k-1} \rangle}{\|\nabla f(x^k)\|^2 + \frac{\tau}{2}\|x^k - x^{k-1}\|^2}$ ▷ inner product
  $U_{k+1} = U_k + \nabla_P h_{x^k, x^{k-1}}(P_k, \beta_k) \circ \nabla_P h_{x^k, x^{k-1}}(P_k, \beta_k)$ ▷ diagonal matrix
  $v_{k+1} = v_k + \nabla_\beta h_{x^k, x^{k-1}}(P_k, \beta_k) \cdot \nabla_\beta h_{x^k, x^{k-1}}(P_k, \beta_k)$ ▷ scalar
  $P_{k+1} = \Pi_{\mathbb{R}^n_+ \cap \mathcal{D}} \big[P_k - \eta_p U_{k+1}^{-1/2} \nabla_P h_{x^k, x^{k-1}}(P_k, \beta_k) \big]$ ▷ diagonal matrix
  $\beta_{k+1} = \Pi_{[0,\, 0.9995]} \big[\beta_k - \eta_b v_{k+1}^{-1/2} \nabla_\beta h_{x^k, x^{k-1}}(P_k, \beta_k) \big]$
  $x^{k+1} = \arg\min_{x \in \{x^k,\, x^{k+1/2}\}} f(x)$
end for
output x^{K+1}

to the discussions in Section 3.3.1. The null step is taken with respect to the function value $f(x)$ instead of the heavy-ball potential function.

- *Memory usage.* The memory usage of `HDM-Best`, measured in terms of number of vectors of length n is $7n$: 1) three vectors store primal iterates x^-, x, x^+. 2) Two vectors store past and buffer gradients $\nabla f(x), \nabla f(x^+)$. 3) A vector stores the diagonal preconditioner P_k. 4) A vector stores the `AdaGrad` stepsize matrix U.

- *Computational cost.* The major additional computation cost arises from computing hypergradient ∇h, which involves one element-wise product and one inner product for vectors of size n. In addition, `HDM-Best` needs to maintain a diagonal matrix for `AdaGrad`. The overall additional computational cost is several $\mathcal{O}(n)$ operations.

3.4.2 Dataset and Testing Problems

We test `HDM-Best` on deterministic convex problems. We adopt two convex optimization tasks in machine learning: support vector machine Lee and Mangasarian [2001] and logistic regression Hastie [2009]. The testing datasets are obtained from `LIBSVM` Chang and Lin [2011].

3.4.3 Experiment Setup

Algorithm Benchmark. We benchmark the following algorithms.

- `GD`. Vanilla gradient descent.

- `GD-HB`. Gradient descent with heavy-ball momentum.
- `AGD-CVX`. The smooth convex version of accelerated gradient descent (Nesterov momentum).
- `AGD-SCVX`. The smooth strongly convex version of accelerated gradient descent.
- `Adam`. Adaptive momentum estimation.
- `AdaGrad`. Adaptive (sub)gradient method.
- `BFGS`. `BFGS` from `scipy`.
- `L-BFGS-Mk`. `L-BFGS` with memory size `k` in `scipy`.
- Practical variant `HDM-Best` uses as memory 7 vectors of size n, comparable to memory for `L-BFGS-M1`.

Algorithm Configuration.

- For `HDM-Best`, we search for the optimal η_p within $\{0.1/L, 1/L, 10/L, 100/L\}$ and $\eta_b \in \{1, 3, 5, 10, 100\}$.
- Stepsize in `GD`, `GD-HB`, `AGD-CVX`, and `AGD-SCVX` are all set to $1/L$.
- The momentum parameter in `GD-HB` is chosen within the set $\{0.1, 0.5, 0.9, 0.99\}$.
- The `Adam` stepsize is chosen within the set $\{1/L, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$. $\beta_1 = 0.9, \beta_2 = 0.999$.
- The `AdaGrad` stepsize is chosen within the set $\{1/L, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$.
- `BFGS`, `L-BFGS-Mk` use default parameters in `scipy`.

Testing Configurations.

1) *Maximum oracle access.* We allow a maximum of 1000 gradient oracles for each algorithm.

2) *Initial point*. All the algorithms are initialized from the same starting point generated from normal distribution $\mathcal{N}(0, I_n)$ and normalized to have unit length.

3) *Stopping criterion.* Algorithms stop if $\|\nabla f\|_\infty \leq 10^{-4}$.

For each algorithm, we record the number of successfully solved instances ($\|\nabla f\|_\infty \leq 10^{-4}$ within 1000 gradient oracles). Table 3.2 summarizes the detailed statistics. The number of instances solved by `HDM-Best` is comparable to that of `L-BFGS-M10`.

Table 3.2: Number of solved problems for each algorithm.

Algorithm/Problem	SVM (33 ↑)	Logistic Regression (33 ↑)
GD	5	2
GD-HB	9	7
AGD-CVX	8	3
AGD-SCVX	7	6
Adam	26	11
AdaGrad	9	8
L-BFGS-M1	13	11
L-BFGS-M3	20	14
L-BFGS-M5	26	16
L-BFGS-M10	31	18
BFGS	32	26
HDM-Best	32	21

Support Vector Machine. Figures 3.4 and 3.5 show the function value gap and gradient norm plots on sample test instances on support vector machine problems. The optimal value for each instance is obtained by running BFGS until $\|\nabla f\|_\infty \leq 10^{-4}$. We see that the practical variant of HDM-Best achieves a significant speedup over other adaptive first-order methods. In particular, HDM-Best often matches L-BFGS-M5 and L-BFGS-M10, while its memory usage is closer to L-BFGS-M1. Notably, Adam also achieves competitive performance in several instances.

Logistic Regression. In logistic regression (Figures 3.6 and 3.7), HDM-Best still compares well with L-BFGS-M5 and is significantly faster than other adaptive first-order methods.

Overall, HDM-Best demonstrates superior performance on deterministic convex problems and is comparable with the mature L-BFGS family. We believe that further development of HDM will fully unleash its potential for a broad range of optimization tasks.

3.5 Concluding Remarks

This paper addresses the long-standing challenge of establishing convergence of the hypergradient descent heuristic. We provide the first rigorous theoretical foundation for hypergradient descent and introduce a novel online learning perspective that extends to other first-order methods with adaptive hyperparameter updates. Our theoretical advances support effective and scalable enhancements that allow the (first-order) HDM to achieve superlinear convergence with guarantees that resemble quasi-Newton methods. Building on these results, we propose HDM-Best, an efficient variant of HDM that performs competitively with the widely used L-BFGS method on convex problems. This empirical success positions HDM as a compelling alternative for modern machine learning. Extending the theory of HDM to stochastic and nonconvex optimization is a crucial next step to understanding its potential

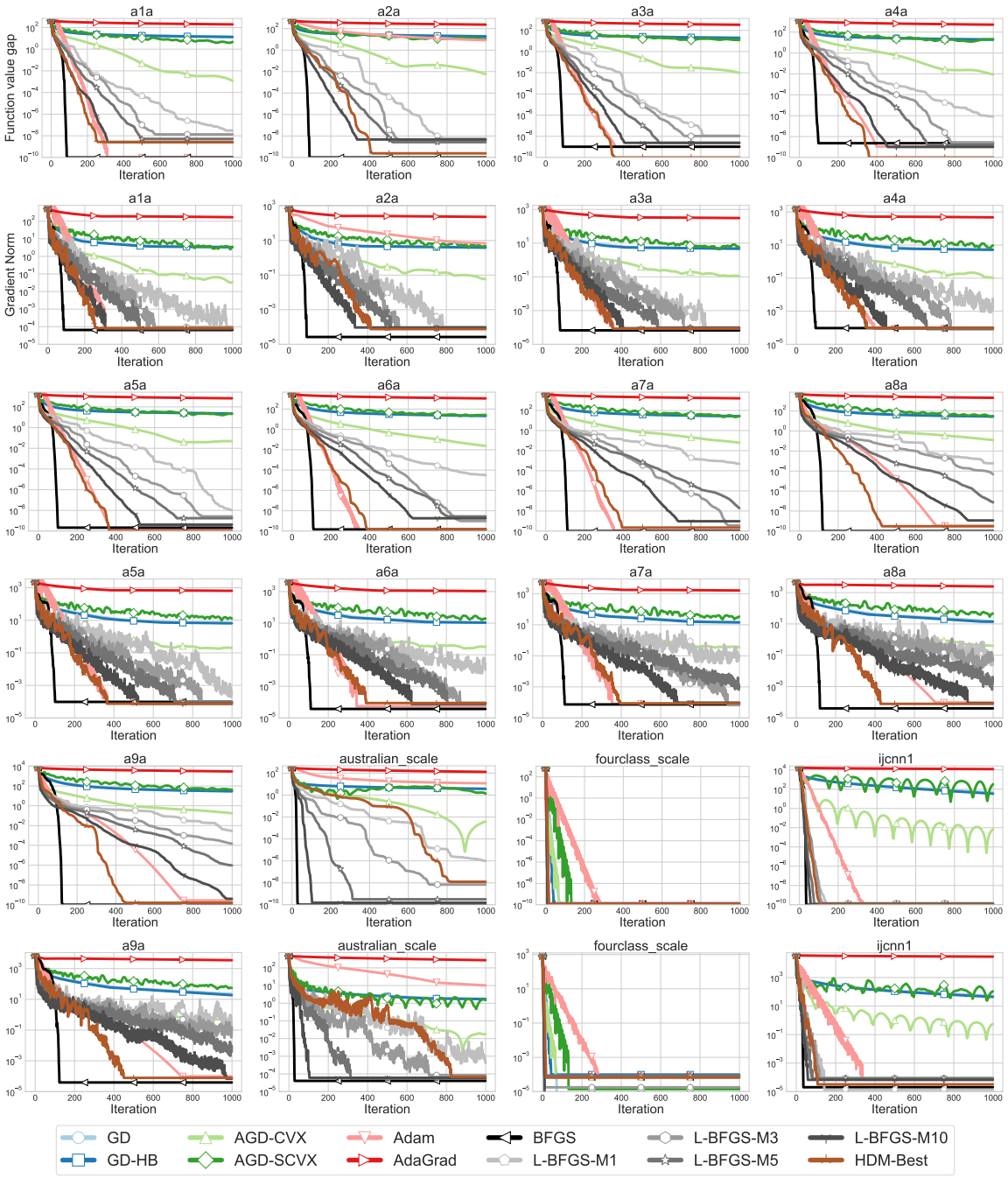


Figure 3.4: Experiments on support vector-machine problem (Part I). Odd rows: function value gap. Even rows: gradient norm

to speed up the training of large-scale models.

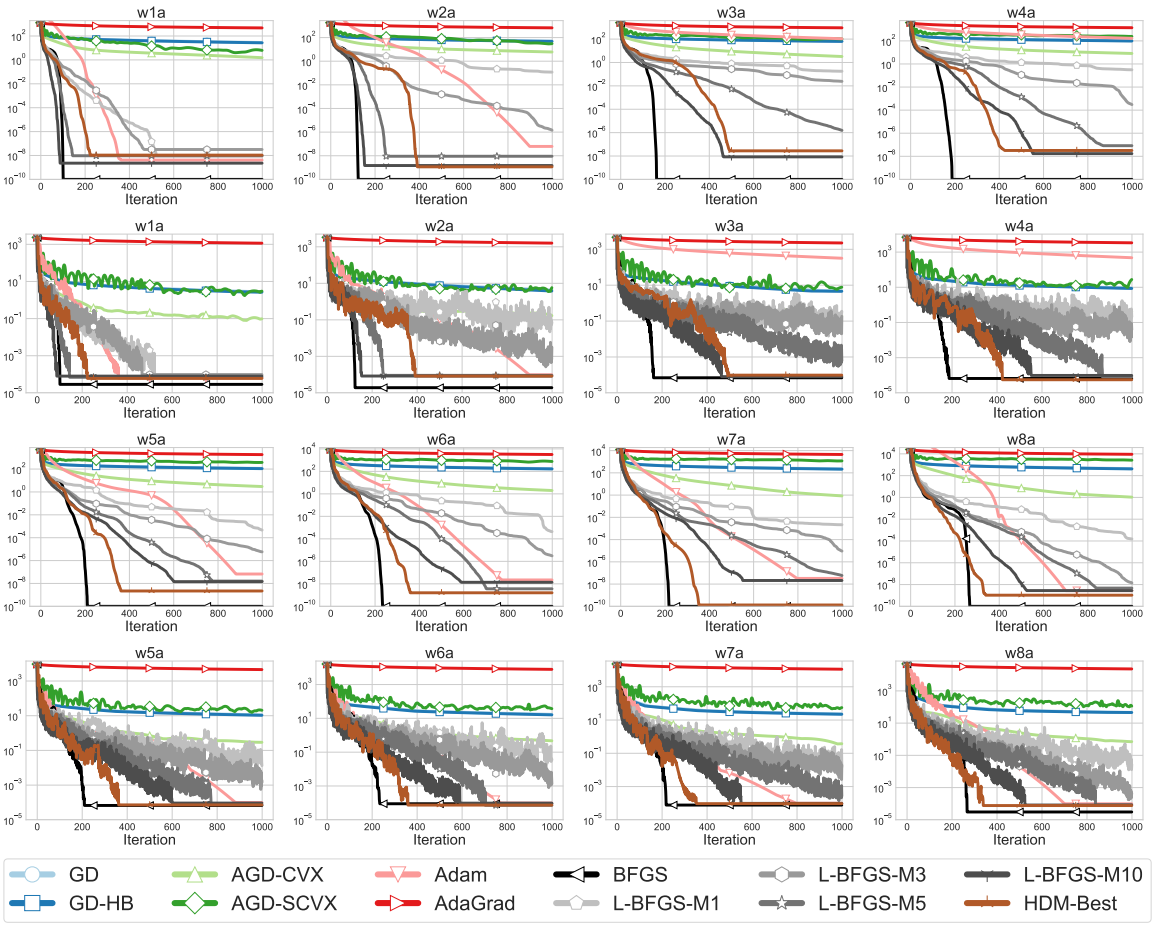


Figure 3.5: Experiments on support vector-machine problem (Part II). Odd rows: function value gap. Even rows: gradient norm

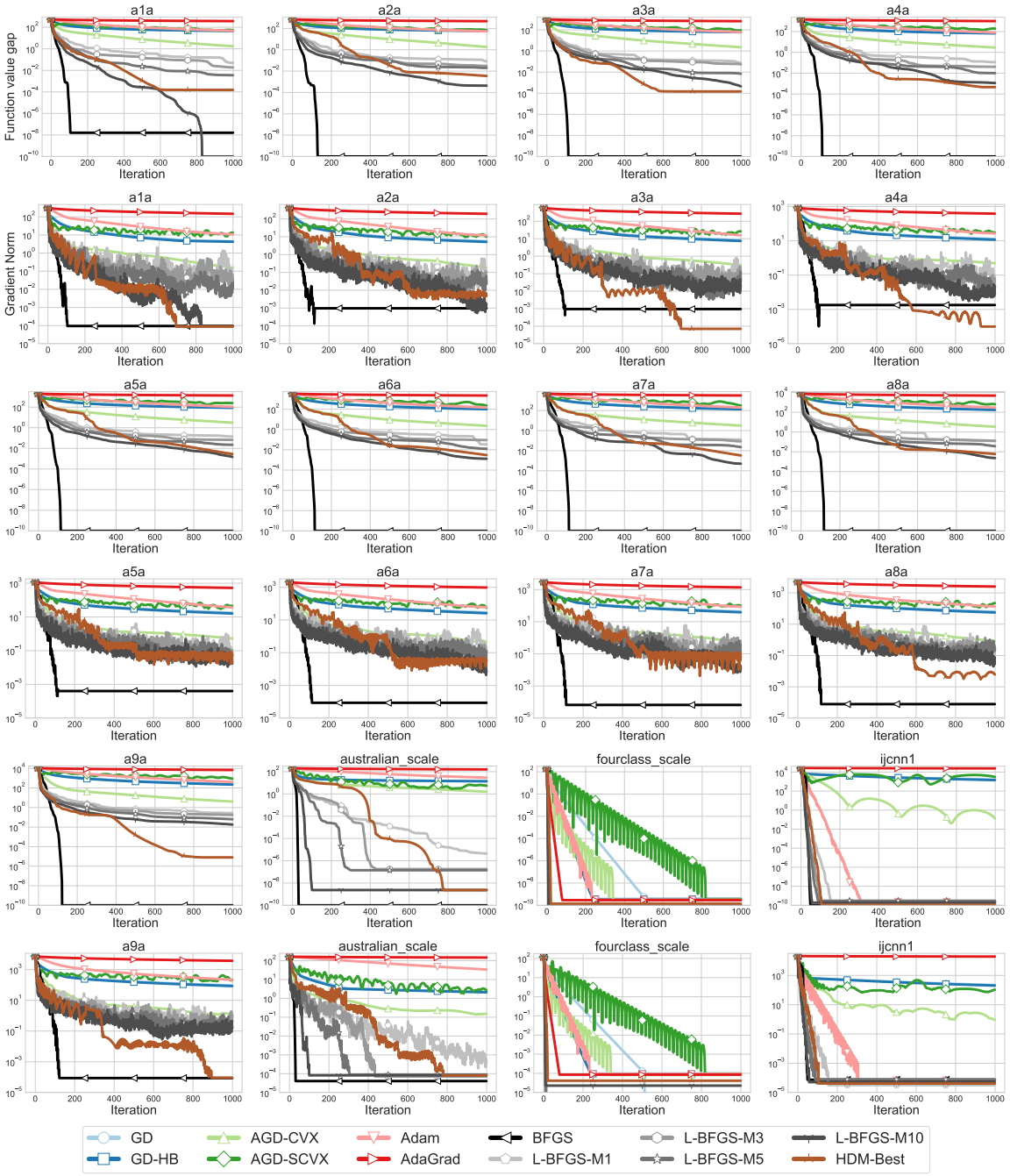


Figure 3.6: Experiments on logistic regression problems (Part I). Odd rows: function value gap. Even rows: gradient norm

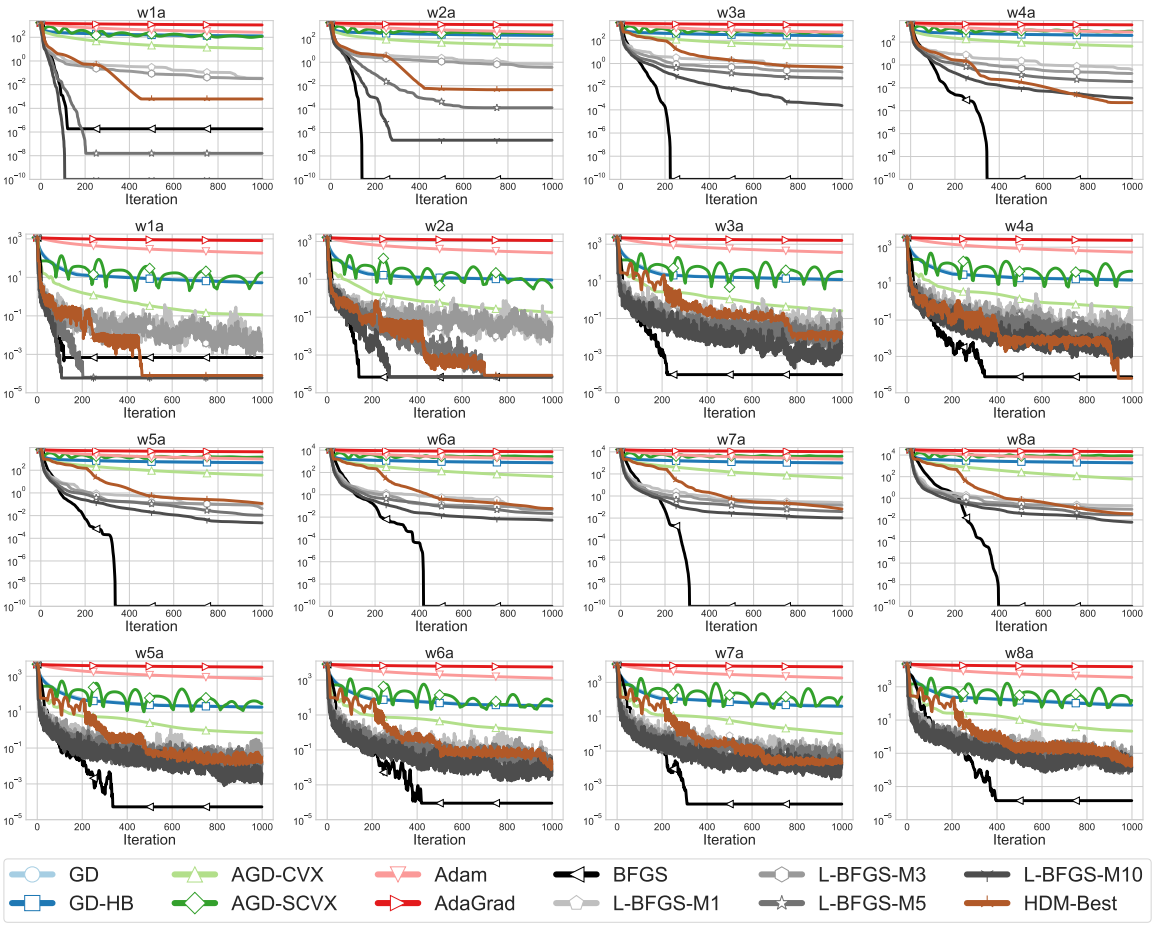


Figure 3.7: Experiments on logistic regression problems (Part II). Odd rows: function value gap. Even rows: gradient norm

Chapter 4

Repair Layers for Neural Networks with Hard Constraints

Deep learning (DL) models have emerged as powerful tools across diverse applications, from computational biology Jumper et al. [2021], Angermueller et al. [2016] and drug discovery Gómez-Bombarelli et al. [2018], Wu et al. [2018] to robotics Levine et al. [2016] and autonomous systems Grigorescu et al. [2020]. Their success stems from an ability to learn complex patterns from data and make accurate predictions in high-dimensional spaces. Increasingly, neural networks (NNs) are being deployed as fast surrogate models that can replace or supplement traditional computational methods. For instance, in domains where the same type of problem must be solved frequently under real-time constraints—such as power systems operation Pan [2021] and robotic control Williams et al. [2017]—or where individual solutions require expensive computation—such as weather forecasting Lam et al. [2023], Bonev et al. [2025] and fluid dynamics simulations Kochkov et al. [2021]—NNs offer orders-of-magnitude speedups compared to conventional solvers while maintaining competitive accuracy.

Despite versatility of DL models, standard DL architectures deliver unconstrained outputs. While simple constraints like simplex constraints and coordinate-wise bounds can be readily enforced through architecture design, it is challenging to enforce complex constraints, even for linear constraints. Research on *constrained neural networks* is growing in the past few years Amos and Kolter [2017], Lu et al. [2021], Iftakher et al. [2025], driven by several compelling needs.

First, many critical applications require constrained outputs to ensure validity and safety. In learning-based control policies, outputs must satisfy safety constraints and physical actuator limits to prevent catastrophic failures Ames et al. [2016]. In protein structure prediction, generated structures must obey fundamental physical constraints such as proper bond geometries, tetrahedral atomic chirality, and the absence of steric clashes to represent physically valid molecular conformations

Chen et al. [2025]. In applications such as optimal power flow in electrical grids or resource allocation in supply chains, outputs must satisfy operational constraints including power balance equations, capacity limits, and other problem-specific requirements that define an executable solution. Violating these constraints can lead to solutions that are not only suboptimal but potentially dangerous or nonsensical in practice.

Secondly, constraints provide a mechanism to inject domain knowledge and inductive biases into NNs, which can potentially improve generalization, particularly when training data is limited. In physics-informed neural networks Raissi et al. [2019], enforcing boundary conditions and physical laws as constraints guides the learning process toward more accurate solutions Li et al. [2024]. By encoding known structural properties of the problem into the network architecture, we guide the learning process toward physically plausible or operationally valid solutions.

The earliest and straightforward approach to handle constraints in NNs adds constraint violations as penalty terms in the loss function Platt and Barr [1987], Zhang and Constantinides [1992], Márquez-Neila et al. [2017], known as *soft constraint* approaches. However, this approach provides no guarantees on constraint satisfaction at inference time. Typical gradient-based training with soft constraints produces constraint violations on the order of 10^{-2} or worse Donti et al. [2021], leading to physically nonsensical or operationally infeasible solutions. Perhaps more surprisingly, our numerical experiments reveal that soft constraint training can be counterproductive even as a warm-start strategy (see Appendix C.4 for numerical demonstration). These observations underscore the necessity for hard constraint enforcement mechanisms that provide both feasibility and efficient training.

In this chapter, we introduce `SnareNet` (Figure 4.1), a novel framework to enforce nonlinear, input-dependent constraints on NNs while preserving their approximation capabilities and enabling end-to-end training. Our key contributions are:

- *A practical framework:* We propose `SnareNet`, which can enforce nonlinear, input-dependent constraints on neural network outputs.
- *Adaptive relaxation training paradigm:* We introduce a novel training strategy that progressively tightens constraint satisfaction during training, balancing feasibility and optimality while avoiding the pathological behaviors often observed in projection-based methods.
- *Feasibility control:* `SnareNet` enables practitioners to explicitly control the desired level of constraint satisfaction, allowing for application-specific trade-offs between strict feasibility and solution quality.
- *Broad applicability:* We demonstrate `SnareNet`'s effectiveness on two problem classes, optimization learning tasks and neural control policies. `SnareNet` consistently produces feasible solutions with better objective values compared to state-of-the-art baselines.

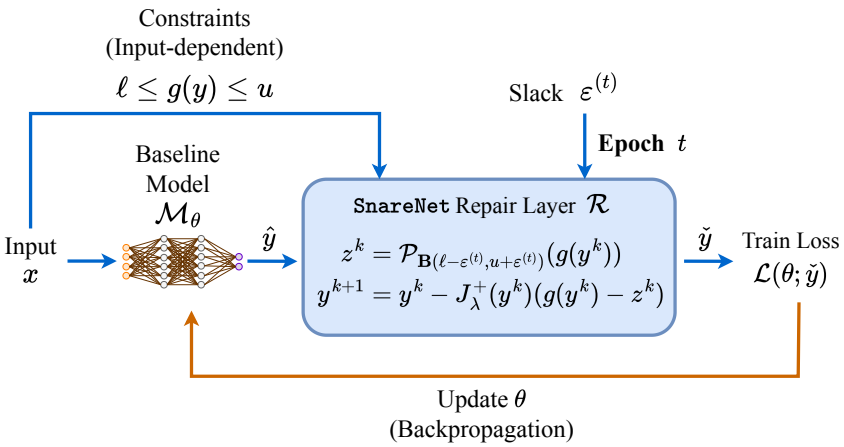


Figure 4.1: Architecture design of `SnareNet`.

The remainder of this paper is organized as follows. Section 4.1 introduces the problem of constrained NN learning and review the literature. Section 4.2 motivates `SnareNet` from linear constraints. Section 4.3 presents our `SnareNet` for nonlinear constraints and our adaptive relaxation training paradigm. Section 4.5 demonstrates our experiments on `SnareNet`. Section 4.6 concludes with a discussion of future directions. We interleave related literature review into Section 4.1 and Section 4.3 since literature were integral to the development of `SnareNet`.

4.1 Constrained Neural Networks

The fundamental task in deep learning applications is to learn complex input-output mappings $\Phi : \mathcal{X} \to \mathcal{Y}$ from data. Given input $x \in \mathcal{X}$, the deep learning model $\mathcal{M}_\theta : \mathcal{X} \to \mathcal{Y}$ parametrized by weights θ is expected to produce an output $\hat{y} = \mathcal{M}_\theta(x)$ that approximates $y = \Phi(x)$.

These deep learning models are typically trained by minimizing an empirical risk over a finite training dataset $\mathcal{X}_{\text{train}} \subset \mathcal{X}$ defined by $\mathcal{L}(\theta) = \frac{1}{|\mathcal{X}_{\text{train}}|} \sum_{x \in \mathcal{X}_{\text{train}}} \ell(\theta; x)$, where $\ell(\theta; x)$ denotes a suitable loss function measuring the discrepancy between predictions and targets for input x under model parameter θ. For simplicity, we consider the setting where $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}^n$ admit vector representations.

In this paper, we consider the task to impose input-dependent (i.e., x-dependent) constraints on the output y, which take the form:

$$\ell_x \le g_x(y) \le u_x, \quad \text{(abbrev. } \ell \le g(y) \le u) \tag{4.1}$$

where $g : \mathbb{R}^n \to \mathbb{R}^m$ is differentiable, $\ell, u \in (\mathbb{R} \cup \{\pm\infty\})^m$ are lower and upper bounds respectively, and all g, ℓ, u are parametrized by input x. For notational simplicity, we suppress the dependence of

g, ℓ, u on x throughout the paper. The formulation in (4.1) encompasses equality constraints when $\ell_i = u_i$ for index i, as well as one-sided constraints when $\ell_i = -\infty$ or $u_i = \infty$. Throughout the paper, we assume the feasible set $\mathcal{C} := \{y \in \mathcal{Y} \mid \ell \leq g(y) \leq u\}$ is non-empty for any $x \in \mathcal{X}$ of interest.

4.1.1 Soft-Constraint Methods

To encourage the model $\mathcal{M}_\theta$ to produce feasible outputs, early approaches augment the loss function with penalty terms that discourage constraint violations by the output[1] $\hat{y} = \hat{y}(\theta, x) := \mathcal{M}_\theta(x)$, also called a *soft loss*:

$$\ell_{\text{soft}}(\theta; x) = \ell(\theta; x) + \mu_u \|\text{ReLU}(g(\hat{y}) - u)\|^2 + \mu_\ell \|\text{ReLU}(\ell - g(\hat{y}))\|^2, \tag{4.2}$$

where $\mu_u > 0$ and $\mu_\ell > 0$ are penalty weights that determine the strength of the penalty, and $\text{ReLU}(\cdot) := \max(0, \cdot)$ is applied elementwise. The soft loss allows the use of standard unconstrained optimization techniques during training. This soft constraint methodology is commonly used for data-parameterized constrained optimization problems Van Hentenryck [2025] and for solving partial differential equations Raissi et al. [2019], Dener et al. [2020]. While penalty methods are straightforward to implement and broadly applicable, models trained with soft loss generally violate constraints on unseen problem instances. Strict constraint satisfaction requires setting μ_u and μ_ℓ to infinity (or a very large number), which however makes the soft loss infinitely ill-conditioned and hard to solve Rathore et al. [2024].

4.1.2 Hard-Constraint Methods

To impose the constraints (4.1) on the output of a baseline model $\mathcal{M}_\theta$, a more sophisticated approach appends a *repair module*[2] $\mathcal{R} : \mathbb{R}^n \to \mathbb{R}^n$ that maps any potentially infeasible output $\hat{y}$ to a feasible one $\check{y}$. We use $\hat{y} = \mathcal{M}_\theta(x)$, pronounced "y-hat", to denote the approximate solution from the baseline model, which is potentially infeasible; and we use $\check{y} = \mathcal{R}(\hat{y})$, pronounced "y-check", to denote the solution after applying the repair module, which is enforced to be feasible. Two paradigms have appeared in the literature to couple repair modules to a baseline neural net, post-processing and end-to-end training.

Post-Processing for Feasibility. The post-processing approach applies the repair module only during inference, but does not allow differentiable training for model parameters θ. The simplest

[1] We denote the output by $\hat{y}$ for simplicity when the dependence on θ and x is clear from the context.

[2] We borrow the word "repair" from the term *repair layer* in Van Hentenryck [2025], but we call $\mathcal{R}$ a *layer* only if it is trainable. The same concept has also been referred to as a *projection layer* Min and Azizan [2024], Grontas et al. [2025], Liang et al. [2024] or a *feasibility layer* Li et al. [2023], Nguyen and Donti [2025] in the literature when the repair module is trainable.

post-processing approach is projection onto the feasible set. Given any test instance x_{test} and a trained model $\mathcal{M}_\theta$, the repair module projects the baseline output onto the feasible set, i.e.,

$$\mathcal{R}(\hat{y}) := \arg\min_{y \in \mathcal{C}} \|y - \mathcal{M}_\theta(x_{\text{test}})\|^2. \tag{4.3}$$

When $\mathcal{C}$ is convex, problem (4.3) is a convex program that typically can be reliably solved by existing optimization solvers. When $\mathcal{C}$ is non-convex, optimization solvers still reliably find feasible (but not always optimal) solutions to Equation (4.3). Alternatively, Liang et al. [2024] propose to use an auxiliary deep learning model that learns a homeomorphic mapping from the feasible set to the unit ball. However, to project an infeasible solution $\hat{y}$ to the feasible set, they require binary search, which precludes end-to-end training of the combined system.

The post-processing approach can lead to several critical problems as demonstrated in [Grontas et al., 2025, Appendix C.4]: models trained without active constraints may diverge on unbounded objectives. Even when they converge, solution quality may be poor after projection since the baseline model fails to anticipate the repair operation.

Trainable Layers for Feasibility. On the other hand, the end-to-end approach activates the repair layer during both training and inference, allowing gradients of model weights θ to flow through the repair operation. The baseline model $\mathcal{M}_\theta$ can therefore adapt to the repair operation. To enable gradient-based optimization of the model parameters θ via backpropagation, the repair layer must use only operations that are differentiable almost everywhere. State-of-the-art trainable repair layers typically employ a fixed number of iterative updates derived from classical optimization algorithms, such as `DC3` Donti et al. [2021] and `ΠNet` Grontas et al. [2025]. `DC3` predicts an initial solution using a soft penalty formulation and moves it toward the feasible region via equality completion and gradient descent on the distance to feasibility. `ΠNet` decomposes an affine feasible set as the intersection of two higher-dimensional convex sets and applies Douglas-Rachford algorithm to find a feasible solution, in which the computational efficiency is improved by restricting the framework to linear and box constraints. While these iterative approaches improve constraint satisfaction on test instances compared to the baseline model $\mathcal{M}_\theta$, backpropagating through these iterations requires intensive GPU memory. In contrast, `HardNet` Min and Azizan [2024] uses a closed-form repair layer that guarantees exact feasibility up to machine precision and enables efficient backpropagation. However, `HardNet` is limited to linear constraints and often produces suboptimal solutions.

4.2 Motivation from Linear Constraints

This section presents a new perspective on `HardNet` Min and Azizan [2024], which enforces linear constraints via a closed-form repair layer. We reinterpret `HardNet`'s repair as a two-step procedure that first projects in image space and then maps the correction back to the domain space. This

preimage perspective is one contribution of the present work and enable a connection to nonlinear constraints that would not be apparent without it.

4.2.1 Closed-Form Repair Layer for Linear Constraints

`HardNet` Min and Azizan [2024] demonstrated that iterative repair layers are unnecessary by introducing a closed-form repair layer that strictly enforces all linear constraints $\ell \leq Ay \leq u$ when $A \in \mathbb{R}^{m \times n}$ has full row rank:

$$\mathcal{R}(\hat{y}) := \hat{y} + A^{+}\boldsymbol{\delta}(A\hat{y}; \ell, u), \quad \text{where } \boldsymbol{\delta}(z; \ell, u) := \mathrm{ReLU}(\ell - z) - \mathrm{ReLU}(z - u) \tag{4.4}$$

and A^{+} is the pseudoinverse of A. We call $\boldsymbol{\delta}(z; \ell, u) \in \mathbb{R}^m$ the *correction vector* to the box $\mathbf{B}(\ell, u) := \{z \in \mathbb{R}^m \mid \ell_i \leq z_i \leq u_i,\ \forall i \in [m]\}$, since

$$\mathcal{P}_{\mathbf{B}(\ell,u)}(z) := \begin{cases} z_i, & \text{if } \ell_i \leq z_i \leq u_i; \\ \ell_i, & \text{if } z_i \leq \ell_i; \\ u_i, & \text{if } z_i \geq u_i; \end{cases} \qquad = z + \boldsymbol{\delta}(z; \ell, u).$$

That is, $\boldsymbol{\delta}(z; \ell, u)$ corrects the vector z to the box $\mathbf{B}(\ell, u)$ by elementwise projection. [Min and Azizan, 2024, Theorem 2] shows that if A has full row rank, then the repair layer (4.4) strictly enforces linear feasibility: for each constraint i (row a_i of A), the repaired output satisfies

$$a_i^{\top}\mathcal{R}(\hat{y}) = \begin{cases} \ell_i, & \text{if } a_i^{\top}\hat{y} < \ell_i; \\ u_i, & \text{if } a_i^{\top}\hat{y} > u_i; \\ a_i^{\top}\hat{y}, & \text{if } \ell_i \leq a_i^{\top}\hat{y} \leq u_i. \end{cases} \tag{4.5}$$

4.2.2 Preimage Perspective

Building on `HardNet`, we observe that the feasible set $\mathcal{C}$ can be expressed as the preimage of $\mathbf{B}(\ell, u)$ under the map A:

$$\begin{aligned} \mathcal{C} &:= \{y \in \mathbb{R}^n \mid \ell \leq Ay \leq u\} \\ &= \{y \in \mathbb{R}^n \mid Ay \in \mathbf{B}(\ell, u)\} \\ &= \text{Preimage of } \mathbf{B}(\ell, u) \text{ under } A. \end{aligned}$$

`HardNet` chooses a particular feasible solution $\check{y} = \mathcal{R}(\hat{y})$ that lies in the preimage of the special point $\mathcal{P}_{\mathbf{B}(\ell,u)}(A\hat{y})$. When A has full row rank (i.e., A is surjective), every vector z admits a non-empty preimage. In particular, $A^{+}z$ given by the pseudoinverse always lies in the preimage of z.

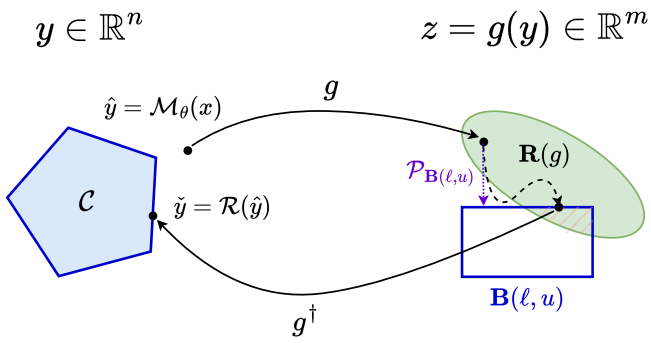


Figure 4.2: The infeasible point $\hat{y} = \mathcal{M}_\theta(x)$ is mapped to an image point $g(\hat{y}) \in \mathbb{R}^m$ that lies outside the box $\mathbf{B}(\ell, u)$. The box projection $\mathcal{P}_{\mathbf{B}_{(\ell,u)}}(g(\hat{y}))$ might have no preimage under non-linear g since it might not lie in joint numerical range $\mathbf{R}(g)$. `SnareNet` finds a path to approach an image point in the intersection $\mathbf{R}(g) \cap \mathbf{B}(\ell, u)$, the preimage of which will be feasible.

In this interpretation, the `HardNet` update 1) decomposes

$$\mathcal{P}_{\mathbf{B}(\ell,u)}(A\hat{y}) = A\hat{y} + \boldsymbol{\delta}(A\hat{y}; \ell, u);$$

and then 2) adjusts $\hat{y}$ by the vector $A^+\boldsymbol{\delta}(A\hat{y}; \ell, u)$, which lies in the preimage of the correction vector. The linearity of A ensures $A^+\boldsymbol{\delta}(A\hat{y}; \ell, u)$ serves as a feasibility correction:

$$A(\hat{y} + A^+\boldsymbol{\delta}(A\hat{y}; \ell, u)) = A\hat{y} + \boldsymbol{\delta}(A\hat{y}; \ell, u) = \mathcal{P}_{\mathbf{B}(\ell,u)}(A\hat{y}).$$

4.2.3 Challenges From Linear to Non-Linear

`HardNet`'s particular choice of $\check{y}$ does not directly extend to nonlinear constraints. Example 1 illustrates the challenge:

Example 1. Constrain $y \in \mathbb{R}^2$ to the intersection of two disks of radius $3/2$, centered at $(-1, 0)$ and $(1, 0)$:

$$g(y) = \begin{bmatrix} g_1(y) \\ g_2(y) \end{bmatrix} := \begin{bmatrix} (y_1+1)^2 + y_2^2 \\ (y_1-1)^2 + y_2^2 \end{bmatrix} \leq u := \begin{bmatrix} 9/4 \\ 9/4 \end{bmatrix}.$$

Now, consider the infeasible prediction $\hat{y} = (-1, 0)$, which violates the second constraints but satisfies the first one as $g_1(\hat{y}) = 0$ and $g_2(\hat{y}) = 4$. The box projection of $g(\hat{y})$ is $\mathcal{P}_{\mathbf{B}(-\infty,u)}(g(\hat{y})) = (0, \frac{9}{4})$.

However, there is *no* point $y \in \mathbb{R}^2$ such that $g(y) = (0, \frac{9}{4})$. That is, the system

$$\begin{cases} g_1(y_1, y_2) = (y_1 + 1)^2 + y_2^2 = 0 \\ g_2(y_1, y_2) = (y_1 - 1)^2 + y_2^2 = \frac{9}{4} \end{cases}$$

has no solution and the preimage of $(0, \frac{9}{4})$ under g is empty.

This example reveals the fundamental difficulty: unlike linear mappings, the *joint numerical range* $\mathbf{R}(g) := \{g(y) \in \mathbb{R}^m \mid y \in \mathbb{R}^n\}$ of a non-linear function g is typically a proper subset of $\mathbb{R}^m$. Consequently, projecting $g(\hat{y})$ onto the box $\mathbf{B}(\ell, u)$ may produce a point that has no preimage under g. To guarantee feasibility, we must instead move towards the intersection $\mathbf{R}(g) \cap \mathbf{B}(\ell, u)$, which characterizes points that are both (i) in the range of g (ensuring a preimage exists) and (ii) within the feasible box (ensuring constraint satisfaction). See Figure 4.2.

4.3 `SnareNet`: Flexible Repair Layer for Hard Constraints

This section introduces `SnareNet`, which efficiently finds a feasible point satisfying constraints (4.1).

4.3.1 Adaptive Newton Update

Given an image point $z \in \mathbf{R}(g) \cap \mathbf{B}(\ell, u)$, Newton's method can find a feasible solution $\check{y} \in \mathcal{C}$ by solving the non-linear system $g(y) = z$. `SnareNet` uses the box projection of $g(y^k)$ as the initial z^k at each Newton iteration:

$$z^k = \mathcal{P}_{\mathbf{B}(\ell,u)}(g(y^k)) \tag{4.6}$$

$$y^{k+1} = \arg\min_y \|J_g(y^k)(y - y^k) + g(y^k) - z^k\|^2 \tag{4.7}$$

$$= y^k - J_g(y^k)^+(g(y^k) - z^k) \tag{4.8}$$

where $J_g(y) \in \mathbb{R}^{m \times n}$ is the Jacobian of g. When $g = A$ is linear and A has full row rank, z^k must lie in $\mathbf{R}(A) \cap \mathbf{B}(\ell, u) = \mathbf{B}(\ell, u)$, so Newton's method converges in one iteration and the update (4.8) reduces to (4.4) in `HardNet`.

4.3.2 Levenberg–Marquardt Regularization

Newton's method (4.7) linearizes g at y_k and minimizes the residual of the linearized system. The linearization approximates g only locally, so Newton's method requires safeguards to ensure convergence. `SnareNet` uses Levenberg-Marquardt (LM) regularization, replacing (4.7) by

$$\min_y \|J_g(y^k)(y - y^k) + g(y^k) - z^k\|^2 + \lambda \|y - y^k\|^2. \tag{4.9}$$

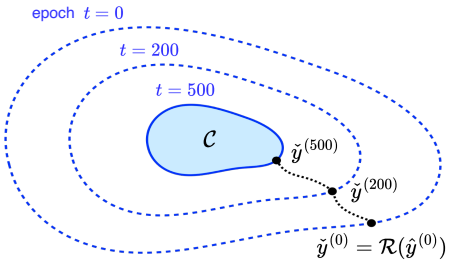


Figure 4.3: Illustration of adaptive constraints relaxation. The figure illustrates a schedule in which $\varepsilon^{(t)} = 0$ for $t \geq 500$. At epoch t, the repair layer $\mathcal{R}$ enforces the output $\check{y}^{(t)}$ lies in the relaxed constraint set $\mathcal{C}_{\varepsilon^{(t)}} = \{y \in \mathbb{R}^n \mid \ell - \varepsilon^{(t)} \leq g(y) \leq u + \varepsilon^{(t)}\}$.

The regularization term $\lambda\|y - y^k\|^2$ with $\lambda > 0$ ensures the new point is close to y^k, at which g is linearized. The regularized algorithm takes the form

$$y^{k+1} = y^k - J_\lambda^\dagger(y^k)(g(y^k) - z^k), \tag{4.10}$$

where $J_\lambda^\dagger(y^k) := [J_g(y^k)^\top J_g(y^k) + \lambda I]^{-1} J_g(y^k)^\top$. `SnareNet` terminates when the update magnitude or constraint violation falls below a prescribed tolerance or the iteration limit is reached.

4.3.3 Adaptive Relaxation

Enforcing strict constraints during early-stage training can harm performance. Min and Azizan [2024] observes that immediate enforcement prevents models from reaching better-optimized final states, and solves the problem by disabling the repair layer for the first few epochs. These "soft-epochs" instead penalize the constraint in the objective. Nguyen and Donti [2025] show that randomly initialized networks produce large violations that force the repair layer to work overtime and reduce the quality of the final solution.

`SnareNet` instead uses a new *adaptive relaxation* training paradigm that snares the neural network at initialization and shrinks it into the feasible set throughout the training process. Rather than strictly enforcing the constraints $\ell \leq g(y) \leq u$ from the beginning of training, we repair the approximate solution $\hat{y}$ towards a relaxed feasible set

$$\mathcal{C}_{\varepsilon^{(t)}} := \{y \in \mathbb{R}^n \mid \ell - \varepsilon^{(t)} \leq g(y) \leq u + \varepsilon^{(t)}\},$$

parametrized by a slack $\varepsilon^{(t)} \geq 0$ at epoch t. This slack progressively decreases to zero over the training epochs $t = 1, 2, \ldots, T$, allowing the model to explore a broader solution space initially while gradually tightening the constraints (see Figure 4.3). We ensure exact feasibility during the last few training epochs by setting $\varepsilon^{(t)} = 0$. `SnareNet` initializes the slack at epoch $t = 0$ to ensure the untrained model $\mathcal{M}_\theta$ satisfies the relaxed constraints, and linearly decays $\varepsilon^{(t)}$ to zero over a preset

Algorithm 10 Repair layer in `SnareNet`

assume $\hat{y} = \mathcal{M}_\theta(x)$ and constraints $\ell \leq g(y) \leq u$
function $\mathcal{R}(\hat{y}, \lambda, \varepsilon)$
    init $y^0 = \hat{y}$
    for $k = 0, 1, 2, \ldots$ until convergence **do**
        compute z^k using Equation (4.11) with ε
        update y^{k+1} using Equation (4.10) with λ
    end for
    return $\check{y} = y^{k+1}$
end function

decay horizon $T_d < T$.

At epoch t, `SnareNet` projects the image point $g(y^k)$ of every LM-iterate y^k to the relaxed box with slack $\varepsilon^{(t)}$:

$$z^k := \mathcal{P}_{\mathbf{B}(\ell - \varepsilon^{(t)}, u + \varepsilon^{(t)})}(g(y^k)) \tag{4.11}$$

Algorithm 10 provides the pseudocode for this repair layer and Algorithm 11 is the complete pseudocode for `SnareNet`'s training algorithm.

Algorithm 11 Training paradigm of `SnareNet`

function TRAIN($\mathcal{X}_{\text{train}}$)
    init neural network $\mathcal{M}_\theta : \mathbb{R}^d \to \mathbb{R}^n$ and $\lambda \geq 0$
    init constraint violation $\varepsilon_x^{(0)}$ for $x \in \mathcal{X}_{\text{train}}$
    set a relaxation parameter schedule $\{\varepsilon_x^{(t)}\}_{t=1}^T$
    for epoch $t = 1, 2, \ldots, T$ **do**
        for mini-batch $\mathcal{B} \subset \mathcal{X}_{\text{train}}$ **do**
            compute approximate solution $\hat{y}_i = \mathcal{M}_\theta(x_i)$ for all $x_i \in \mathbf{B}$
            repair to relaxed feasible solution $\check{y}_i = \mathcal{R}(\hat{y}_i, \lambda, \varepsilon_{x_i}^{(t)})$ for all $x_i \in \mathbf{B}$
            compute batch loss $\mathcal{L}_\mathcal{B}(\theta)$
            update θ using $\nabla_\theta \mathcal{L}_\mathcal{B}(\theta)$
        end for
    end for
end function

4.4 Guarantees and Computational Remarks

Section 4.4.1 shows that the repair layer of `SnareNet` converges to a feasible point if the constraints are convex. Section 4.4.2 discusses the computational aspects of `SnareNet`.

4.4.1 Convergence Guarantees

The key observation which enables a convergence analysis is that the `SnareNet`'s update is *equivalent to preconditioned gradient descent* on the constraint violation function

$$F(y) = \frac{1}{2}\|g(y) - \mathcal{P}_{\mathbf{B}(\ell,u)}(g(y))\|^2. \tag{4.12}$$

Proposition 4.1 below shows that F is twice differentiable almost everywhere, despite the presence of the seemingly non-smooth projection operator $\mathcal{P}_{\mathbf{B}(\ell,u)}$. In addition, when the feasible set $\mathcal{C}$ has a convex representation:

$$\mathcal{C} := \{y \in \mathbb{R}^n \mid h(y) \leq 0,\, \ell_A \leq Ay \leq u_A\} \tag{4.13}$$

with convex $h : \mathbb{R}^n \to \mathbb{R}^{m_h}$, matrix $A \in \mathbb{R}^{m_A \times n}$, and bounds $\ell_A, u_A \in \mathbb{R}^{m_A}$, the function F is in fact convex.

Proposition 4.1 (Differentiability & Convexity)**.** *Let $g : \mathbb{R}^n \to \mathbb{R}^m$ be a twice differentiable function. Then $F(y)$ defined in* (4.12) *is twice differentiable almost everywhere and its gradient is*

$$\nabla F(y) = J_g(y)^\top (g(y) - \mathcal{P}_{\mathbf{B}(\ell,u)}(g(y))). \tag{4.14}$$

In particular, if the feasible set $\mathcal{C}$ takes the convex form (4.13)*, the constraint violation function $F(y)$ is convex.*

Proof. Observe that $F(y) = R(g(y))$ a composition of g and a separable function $R : \mathbb{R}^m \to \mathbb{R}$ defined by

$$R(z) := \frac{1}{2}\|z - \mathcal{P}_{\mathbf{B}(\ell,u)}(z)\|^2 = \frac{1}{2}\sum_{i=1}^{m}(z_i - \mathcal{P}_{[\ell_i,u_i]}(z_i))^2, \tag{4.15}$$

where $\mathcal{P}_{[\ell_i,u_i]}$ being the one-dimensional projection operator onto the interval $[\ell_i, u_i]$.

- **Almost everywhere twice differentiability.** It suffices to show that $r_{[\ell,u]}(a) := \frac{1}{2}(a - \mathcal{P}_{[\ell,u]}(a))^2$ with $a \in \mathbb{R}$ is twice differentiable almost everywhere since $R(z) = \sum_{i=1}^{m} r_{[\ell_i,u_i]}(z_i)$ is the sum of such functions and $F(y) = R(g(y))$ is a composition of R and a twice differentiable g. The piecewise definition of $r(a)$ and its derivatives $r'(a)$ and $r''(a)$ are

$$r_{[\ell,u]}(a) = \begin{cases} \frac{1}{2}(a-\ell)^2, & \text{if } a < \ell; \\ 0, & \text{if } \ell \leq a \leq u; \\ \frac{1}{2}(a-u)^2, & \text{if } a > u, \end{cases} \qquad r'_{[\ell,u]}(a) = \begin{cases} a-\ell, & \text{if } a < \ell; \\ 0, & \text{if } \ell \leq a \leq u; \\ a-u, & \text{if } a > u, \end{cases}$$

$$r''_{[\ell,u]}(a) = \begin{cases} 0, & \text{if } \ell < a < u; \\ 1, & \text{if } a < \ell \text{ or } a > u. \end{cases}$$

Note that $r'_{[\ell,u]}(a) = a - \mathcal{P}_{[\ell,u]}(a)$ is continuous everywhere, and the second derivative $r''_{[\ell,u]}(a)$ is well-defined for all $a \in \mathbb{R}$ except at the two points $a = \ell$ and $a = u$. Thus, $r_{[\ell,u]}(a)$ is twice differentiable almost everywhere, and so is $F(y)$.

- **Gradient expression.** Note that $\nabla R(z) = z - \mathcal{P}_{\mathbf{B}(\ell,u)}(z)$ since $r'_{[\ell,u]}(a) = a - \mathcal{P}_{[\ell,u]}(a)$. The gradient of $F(y)$ follows by the chain rule:

$$\nabla F(y) = J_g(y)^\top \nabla R(g(y)) = J_g(y)^\top (g(y) - \mathcal{P}_{\mathbf{B}(\ell,u)}(g(y))).$$

- **Convexity of F when $\mathcal{C}$ is convex.** When $\mathcal{C}$ takes the form (4.13), the constraint function g and bounds ℓ, u in `SnareNet` are

$$g(y) := \begin{bmatrix} h(y) \\ Ay \end{bmatrix}, \quad \ell := \begin{bmatrix} -\infty \\ \ell_A \end{bmatrix}, \quad u := \begin{bmatrix} 0 \\ u_A \end{bmatrix},$$

so F takes the form

$$F(y) = \frac{1}{2}\| \max(0, h(y))\|^2 + \frac{1}{2}\|Ay - \mathcal{P}_{\mathbf{B}(\ell_A,u_A)}(Ay)\|^2.$$

The first term is convex since h is convex and $a \mapsto \frac{1}{2}(\max(0,a))^2$ is convex nondecreasing applied componentwise. For the second term, observe that $R_A(z) := \frac{1}{2}\|z - \mathcal{P}_{\mathbf{B}(\ell_A,u_A)}(z)\|^2 = \sum_{i=1}^{m_A} r_{[\ell_{A,i},u_{A,i}]}(z_i)$ is a sum of convex functions since each $r_{[\ell_{A,i},u_{A,i}]}$ is convex. Thus R_A is convex. As $R_A(Ay)$ is a composition of a convex function R_A and a linear function Ay, the second term is convex. Thus F is convex as a sum of convex functions.

□

Given $\nabla F(y)$ in (4.14), `SnareNet`'s repair update (4.10) with $\lambda > 0$ can be expressed as

$$\begin{aligned} y^{k+1} &= y^k - (J_g(y^k)^\top J_g(y^k) + \lambda I)^{-1} J_g(y^k)^\top (g(y^k) - \mathcal{P}_{\mathbf{B}(\ell,u)}(g(y^k))) \\ &= y^k - (J_g(y^k)^\top J_g(y^k) + \lambda I)^{-1} \nabla F(y^k). \end{aligned} \tag{4.16}$$

Equation (4.16) can be interpreted as a PGD step on $F(y)$ with preconditioner $P_k := (J_g(y^k)^\top J_g(y^k) + \lambda I)^{-1}$. To show the convergence under the choice P_k, we first establish an upper bound on $F(y)$ in Lemma 4.2 that is analogous to the quadratic upper bound for smooth functions.

Lemma 4.2 (Upper bound on F)**.** *Let $g : \mathbb{R}^n \to \mathbb{R}^m$ be a twice differentiable function with an L_g-Lipschitz continuous Jacobian. Then for any $x, y \in \mathbb{R}^n$, the constraint violation function $F(y)$*

defined in (4.12) *satisfies the bound*

$$\begin{aligned} F(y) \le F(x) + \langle \nabla F(x), y - x\rangle + \frac{1}{2}(y-x)^\top J_g(x)^\top J_g(x)(y-x) \\ + \frac{L_g}{2}\left[\|\nabla R(g(x))\| + \|J_g(x)(y-x)\| + \frac{L_g}{4}\|y-x\|^2\right]\|y-x\|^2. \end{aligned} \tag{4.17}$$

In particular, if $g(y) = Ay$ *is linear with* $A \in \mathbb{R}^{m\times n}$*, the bound in* (4.17) *reduces to*

$$F(y) \le F(x) + \langle \nabla F(x), y - x\rangle + \frac{1}{2}(y-x)^\top A^\top A(y-x). \tag{4.18}$$

Proof. We first show that $R(z)$ defined in (4.15) is 1-smooth, or equivalently, its gradient $\nabla R(z) = z - \mathcal{P}_{\mathbf{B}(\ell,u)}(z)$ is 1-Lipschitz continuous: for any $z, z' \in \mathbb{R}^m$, we have

$$\begin{aligned} \|\nabla R(z) - \nabla R(z')\|^2 &= \|(z - \mathcal{P}_{\mathbf{B}(\ell,u)}(z)) - (z' - \mathcal{P}_{\mathbf{B}(\ell,u)}(z'))\|^2 \\ &= \|z-z'\|^2 - 2\langle \mathcal{P}_{\mathbf{B}(\ell,u)}(z) - \mathcal{P}_{\mathbf{B}(\ell,u)}(z'), z - z'\rangle + \|\mathcal{P}_{\mathbf{B}(\ell,u)}(z) - \mathcal{P}_{\mathbf{B}(\ell,u)}(z')\|^2 \\ &\le \|z-z'\|^2 - \|\mathcal{P}_{\mathbf{B}(\ell,u)}(z) - \mathcal{P}_{\mathbf{B}(\ell,u)}(z')\|^2 \\ &\qquad\qquad \text{(by firmly non-expansiveness of } \mathcal{P}_{\mathbf{B}(\ell,u)}) \\ &\le \|z-z'\|^2. \end{aligned}$$

Apply the 1-smoothness of $R(z)$ to upper bound $F(y) = R(g(y))$ by

$$\begin{aligned} F(y) = R(g(y)) &\le R(g(x)) + \langle \nabla R(g(x)), g(y) - g(x)\rangle + \frac{1}{2}\|g(y) - g(x)\|^2 \\ &= F(x) + \langle \nabla R(g(x)), g(y) - g(x)\rangle + \frac{1}{2}\|g(y) - g(x)\|^2. \end{aligned} \tag{4.19}$$

Let $e_x(y) := g(y) - g(x) - J_g(x)(y-x)$ denote the error of linear approximation. Using the fundamental theorem of calculus and the L_g-Lipschitz continuity of J_g, the error can be bounded by

$$\|e_x(y)\| = \left\|\int_0^1 [J_g(x + t(y-x)) - J_g(x)](y-x)\,dt\right\| \le \int_0^1 L_g t\|y-x\|^2\,dt = \frac{L_g}{2}\|y-x\|^2. \tag{4.20}$$

Cauchy-Schwartz inequality and (4.20) imply

$$\begin{aligned} \langle \nabla R(g(x)), g(y) - g(x)\rangle &= \langle \nabla R(g(x)), g(y) - g(x) - J_g(x)(y-x)\rangle + \langle \nabla R(g(x)), J_g(x)(y-x)\rangle \\ &\le \|\nabla R(g(x))\|\|g(y) - g(x) - J_g(x)(y-x)\| + \langle J_g(x)^\top \nabla R(g(x)), y - x\rangle \\ &\le \frac{L_g\|\nabla R(g(x))\|}{2}\|y-x\|^2 + \langle \nabla F(x), y - x\rangle. \end{aligned} \tag{4.21}$$

Equation (4.20) also implies

$$\begin{aligned}\|g(y)-g(x)\|^2 &= \|e_x(y)+J_g(x)(y-x)\|^2 \\ &= \|J_g(x)(y-x)\|^2 + 2\langle J_g(x)(y-x), e_x(y)\rangle + \|e_x(y)\|^2 \\ &\le \frac{1}{2}(y-x)^\top J_g(x)^\top J_g(x)(y-x) + L_g\|J_g(x)(y-x)\|\|y-x\|^2 + \frac{L_g^2}{4}\|y-x\|^4.\end{aligned} \tag{4.22}$$

Combine (4.19), (4.21), and (4.22) to conclude (4.17). When $g(y) = A(y)$ is linear, the Jacobian of g is constant $J_g(y) = A$ for all $y \in \mathbb{R}^n$ and thus $L_g = 0$. The bound reduces to (4.18). □

If $\mathcal{C}$ contains only linear constraints (i.e., $g(y) = Ay$) and A has full row rank, then F satisfies the Polyak–Łojasiewicz (PL) condition (see Lemma 4.3).

Lemma 4.3 (PL-condition in preconditioned norm)**.** *Suppose $J_g(y)$ has full-row rank with minimum singular value $\sigma_{\min}(J_g(y)) > 0$ and $\mathcal{C} := \{y \in \mathbb{R}^n \mid \ell \le g(y) \le u\}$ is non-empty. The constraint violation function $F(y)$ defined in (4.12) satisfies the Polyak-Lojasiewicz (PL) condition under the norm $\|\cdot\|_{(J_g(y)^\top J_g(y)+\lambda I)^{-1}}$:*

$$\frac{1}{2}\|\nabla F(y)\|^2_{(J_g(y)^\top J_g(y)+\lambda I)^{-1}} \ge \frac{\sigma^2_{\min}(J_g(y))}{\sigma^2_{\min}(J_g(y))+\lambda}[F(y) - \min_{y\in\mathbb{R}^n} F(y)] = \frac{\sigma^2_{\min}(J_g(y))}{\sigma^2_{\min}(J_g(y))+\lambda}F(y).$$

Proof. Let $d(y) := g(y) - \mathcal{P}_{\mathbf{B}}(g(y))$. Since $\nabla F(y) = J_g(y)^\top d(y)$, the gradient under preconditioned norm is

$$\|\nabla F(y)\|^2_{(J_g(y)^\top J_g(y)+\lambda I)^{-1}} = d(y)^\top J_g(y)[J_g(y)^\top J_g(y) + \lambda I]^{-1} J_g(y)^\top d(y).$$

Let $\sigma_i(J_g(y))$ denote the i-th singular value of $J_g(y)$. Then the singular values of $J_g(y)[J_g(y)^\top J_g(y)+\lambda I]^{-1}J_g(y)^\top$ are $\frac{\sigma_i^2(J_g(y))}{\sigma_i^2(J_g(y))+\lambda}$ for $i = 1, \dots, m$ with minimum $\frac{\sigma^2_{\min}(J_g(y))}{\sigma^2_{\min}(J_g(y))+\lambda}$. Since $F(y) = \frac{1}{2}\|d(y)\|^2$ and non-empty $\mathcal{C}$ guarantees $\min_{y\in\mathbb{R}^n} F(y) = 0$, we conclude

$$\frac{1}{2}\|\nabla F(y)\|^2_{(J_g(y)^\top J_g(y)+\lambda I)^{-1}} \ge \frac{\sigma^2_{\min}(J_g(y))}{2(\sigma^2_{\min}(J_g(y))+\lambda)}\|d(y)\|^2 = \frac{\sigma^2_{\min}(J_g(y))}{\sigma^2_{\min}(J_g(y))+\lambda}F(y).$$

□

The properties of F in Lemmas 4.2 and 4.3 ground the convergence guarantees of `SnareNet` in Theorem 4.4 and Theorem 4.5. The choice of preconditioner H_λ enables a fast linear convergence rate when $g(y) = Ay$; and a sublinear convergence rate for general convex constraints.

Theorem 4.4 (Linear Convergence for Full Row Rank Linear Constraints)**.** *Let $g(y) = Ay$ and assume A has full row rank with minimum singular value $\sigma_{\min} > 0$. Suppose the feasible set $\mathcal{C} := \{y \in \mathbb{R}^n \mid \ell \le Ay \le u\}$ is non-empty. Then `SnareNet`'s update (4.6)+(4.10) with $\lambda > 0$ converges*

linearly to a feasible point at rate

$$F(y^k) \leq \left(1 - \frac{\sigma_{\min}^2}{\sigma_{\min}^2 + \lambda}\right)^k F(y^0).$$

Proof. Denote $H_\lambda := A^\top A + \lambda I$. Then $y^{k+1} = y^k - H_\lambda^{-1}\nabla F(y^k)$ and $A^\top A \prec H_\lambda$. Substitute $y = y^{k+1}$ and $x = y^k$ into (4.18) to obtain

$$\begin{aligned} F(y^{k+1}) &\leq F(y^k) + \langle \nabla F(y^k), y^{k+1} - y^k \rangle + \frac{1}{2}(y^{k+1} - y^k)^\top A^\top A (y^{k+1} - y^k) \\ &\leq F(y^k) + \langle \nabla F(y^k), y^{k+1} - y^k \rangle + \frac{1}{2}(y^{k+1} - y^k)^\top H_\lambda (y^{k+1} - y^k) \\ &= F(y^k) - \nabla F(y^k)^\top H_\lambda^{-1} \nabla F(y^k) + \frac{1}{2}\nabla F(y^k)^\top H_\lambda^{-1}\nabla F(y^k) \\ &\leq F(y^k) - \frac{1}{2}\|\nabla F(y^k)\|^2_{H_\lambda^{-1}}, \end{aligned} \tag{4.23}$$

proving the descent property for `SnareNet`'s update. Since A has full row rank, Lemma 4.3 guarantees $F(y)$ satisfies PL-condition and thus

$$F(y^{k+1}) \leq F(y^k) - \frac{\sigma_{\min}^2}{\sigma_{\min}^2 + \lambda} F(y^k) = \left(1 - \frac{\sigma_{\min}^2}{\sigma_{\min}^2 + \lambda}\right) F(y^k).$$

□

Theorem 4.4 suggests a small λ for faster convergence and the choice of $\lambda = 0$ matches one step convergence as in `HardNet`. However, the empirical results in Figure 4.6 show that a small λ can lead to worse final model performance.

Theorem 4.5 (Convergence for Convex Constraints)**.** *Suppose $\mathcal{C}$ in* (4.13) *is non-empty and $h : \mathbb{R}^n \to \mathbb{R}^{m_h}$ is a twice differentiable convex function with L-Lipschitz continuous Jacobian. Let $y^0 \in \mathbb{R}^n$ be the initial point, $\mathcal{D} := \{y \in \mathbb{R}^n \mid F(y) \leq F(y^0)\}$ be the sublevel set, $D := \sup_{y \in \mathcal{D}} \inf_{y^* \in \mathcal{C}} \|y - y^*\|$, and $\sigma_{\max}^2(J_h(y)) + \sigma_{\max}^2(A) \leq \sigma^2$ for all $y \in \mathcal{D}$. For $\lambda > 0$ satisfying $\lambda^2 - \sqrt{2F(y^0)}L\lambda + \frac{L^2 F(y^0)^2}{16} \geq 0$, `SnareNet` converges to a feasible point at rate*

$$F(y^k) \leq \frac{F(y^0)}{1 + \frac{F(y^0)}{2D^2(\sigma^2+\lambda)}k}. \tag{4.24}$$

Proof. Observe that F is convex under the convex constraints represented in (4.13) by Proposition 4.1. Using `SnareNet`'s notation, the constraint function g, the bounds ℓ, u, and the Jacobian J_g are (defined) as

$$g(y) := \begin{bmatrix} h(y) \\ Ay \end{bmatrix}, \ \ell := \begin{bmatrix} -\infty \\ \ell_A \end{bmatrix}, \ u := \begin{bmatrix} 0 \\ u_A \end{bmatrix}, \ J_g(y) := \begin{bmatrix} J_h(y) \\ A \end{bmatrix}.$$

The Jacobian of g is L-Lipschitz continuous since $J_h(y)$ is L-Lipschitz continuous and A is constant.

Now, we show that our choice of λ guarantees $y^k \in \mathcal{D}$ for all $k \geq 0$ and also the descent property $F(y^{k+1}) \leq F(y^k)$. We prove by induction. Note that $y^0 \in \mathcal{D}$ by definition. Suppose $y^k \in \mathcal{D}$ by induction hypothesis. Substitute $y = y^{k+1}$ and $x = y^k$ into (4.17) to obtain

$$\begin{aligned} F(y^{k+1}) \leq F(y^k) + \langle \nabla F(y^k), y^{k+1} - y^k \rangle + \tfrac{1}{2}(y^{k+1} - y^k)^\top J_g(y^k)^\top J_g(y^k)(y^{k+1} - y^k) \\ + \tfrac{L_g}{2}\left[\|\nabla R(g(y^k))\| + \|J_g(y^k)(y^{k+1} - y^k)\| + \tfrac{L_g}{4}\|y^{k+1} - y^k\|^2 \right] \|y^{k+1} - y^k\|^2. \end{aligned} \tag{4.25}$$

Since $y^k \in \mathcal{D}$, we have $\|\nabla R(g(y^k))\| = \|g(y^k) - \mathcal{P}_{\mathbf{B}(\ell,u)}(g(y^k))\| = \sqrt{2F(y^k)} \leq \sqrt{2F(y^0)} =: M$ and thus

$$\|J_g(y^k)(y^{k+1} - y^k)\| \leq \left\|J_g(y^k)(J_g(y^k)^\top J_g(y^k) + \lambda I)^{-1} J_g(y^k)^\top\right\| \left\|\nabla R(g(y^k))\right\| \leq M, \tag{4.26}$$

$$\|y^{k+1} - y^k\|^2 = \left\|(J_g(y^k)^\top J_g(y^k) + \lambda I)^{-1} J_g(y^k)^\top\right\|^2 \left\|\nabla R(g(y^k))\right\|^2 \leq \tfrac{M^2}{4\lambda}. \tag{4.27}$$

Let $H_k := J_g(y^k)^\top J_g(y^k) + \lambda I$. Then `SnareNet`'s update reads $y^{k+1} = y^k - H_k^{-1}\nabla F(y^k)$ and (4.25) becomes

$$\begin{aligned} F(y^{k+1}) &\leq F(y^k) + \langle \nabla F(y^k), y^{k+1} - y^k \rangle + \tfrac{1}{2}(y^{k+1} - y^k)^\top J_g(y^k)^\top J_g(y^k)(y^{k+1} - y^k) \\ &\quad + \left(L_g M + \tfrac{L_g^2 M^2}{32\lambda}\right)\|y^{k+1} - y^k\|^2 && \text{(by (4.26) and (4.27))} \\ &\leq F(y^k) + \langle \nabla F(y^k), y^{k+1} - y^k \rangle + \tfrac{1}{2}(y^{k+1} - y^k)^\top H_k(y^{k+1} - y^k) \\ &&& \text{(by } \lambda^2 - L_g M\lambda - \tfrac{L_g^2 M^2}{32} \geq 0\text{)} \\ &= F(y^k) - \tfrac{1}{2}\|\nabla F(y^k)\|^2_{H_k^{-1}}, && (4.28) \end{aligned}$$

which guarantees $F(y^{k+1}) \leq F(y^k)$ and thus $y^{k+1} \in \mathcal{D}$. The claim that $y^k \in \mathcal{D}$ for all $k \geq 0$ is proved by induction and the descent property is established.

Next, we prove the sublinear convergence using the convexity of F and the descent property (4.28). Since F is convex and the feasible set $\mathcal{C}$ is non-empty, we have $\min_{y\in\mathbb{R}^n} F(y) = 0$ and

$$F(y^k) = F(y^k) - \min_{y\in\mathbb{R}^n} F(y) \leq \|\nabla F(y^k)\|_{H_k^{-1}} \|y^k - \mathcal{P}_{\mathcal{C}}(y^k)\|_{H_k}. \tag{4.29}$$

To bound $\|y^k - \mathcal{P}_{\mathcal{C}}(y^k)\|_{H_k}$, observe that $\|H_k\| \leq \sigma^2_{\max}(J_g(y^k)) + \lambda \leq \sigma^2_{\max}(J_h(y^k)) + \sigma^2_{\max}(A) + \lambda \leq \sigma^2 + \lambda$ since $y^k \in \mathcal{D}$, and thus

$$\|y^k - \mathcal{P}_{\mathcal{C}}(y^k)\|^2_{H_k} \leq \|H_k\| \|y^k - \mathcal{P}_{\mathcal{C}}(y^k)\|^2 \leq (\sigma^2 + \lambda) D^2. \tag{4.30}$$

Combine (4.28), (4.29), and (4.30) to obtain

$$F(y^{k+1}) \leq F(y^k) - \frac{F(y^k)^2}{2D^2(\sigma^2+\lambda)}. \tag{4.31}$$

If $F(y^{k+1}) = 0$, the desired bound follows immediately. Otherwise, (4.31) implies

$$\frac{1}{F(y^{k+1})} - \frac{1}{F(y^k)} \geq \frac{F(y^k) - F(y^{k+1})}{F(y^k)F(y^{k+1})} \geq \frac{F(y^k)}{2D^2(\sigma^2+\lambda)F(y^{k+1})} \geq \frac{1}{2D^2(\sigma^2+\lambda)}.$$

The telescoping sum yields

$$\frac{1}{F(y^{k+1})} \geq \frac{1}{F(y^0)} + \frac{k+1}{2D^2(\sigma^2+\lambda)},$$

which implies the desired sublinear convergence bound (4.24). □

Convex constraints requires sufficiently large λ for convergence while too large λ can lead to slow convergence and increase the computational burden of repair layer.

4.4.2 Computational Remarks

The primary limitation of `SnareNet` is the computational cost to apply $J_\lambda^\dagger(y^k) = [J_g(y^k)^\top J_g(y^k) + \lambda I]^{-1} J_g(y^k)^\top$. By push-through identity (from Woodbury matrix identity)

$$\begin{aligned}&[J_g(y^k)^\top J_g(y^k) + \lambda I]^{-1} J_g(y^k)^\top \\ &= J_g(y^k)^\top [J_g(y^k) J_g(y^k)^\top + \lambda I]^{-1},\end{aligned}$$

single `SnareNet` iteration costs $\mathcal{O}(\min\{n,m\}^3)$, where m is output dimension of g. `SnareNet` is more suitable in the regime $m \ll n$. See Appendix C.3 for supporting experiments on the scaling of computational resources with respect to m and n. To scale to large-scale problems, `SnareNet` can apply the inverse by solving the linear system using iterative methods like conjugate gradient, which avoids forming the Jacobian explicitly and only requires vector-Jacobian products (VJP). One may leverage recently developed automatic differentiation frameworks such as `torch.autograd.functional.vjp` or `jax.vjp` for more efficient VJP computation. To verify the validity of `SnareNet` on constraint enforcement, we apply the inverse directly by `torch.linalg.solve` in our experiments.

4.5 Experiments

This section demonstrates the effectiveness of `SnareNet` on optimization learning and neural control policies.[3]

[3]Code for experiments is available at `https://github.com/miniyachi/SnareNet`.

4.5.1 Optimization Learning

The *optimization learning* Van Hentenryck [2025] task seeks a fast surrogate neural solver for a family of optimization problems parametrized by input $x \in \mathcal{X}$:

$$\min_{y \in \mathbb{R}^n} f_x(y) \;\text{ subject to }\; \ell_x \leq g_x(y) \leq u_x, \tag{4.32}$$

Both the objective $f_x : \mathbb{R}^n \to \mathbb{R}$ and feasible set $\ell_x \leq g_x(y) \leq u_x$ are parametrized. Problems of the form (4.32) can be non-linear and non-convex, and can be slow to solve with traditional optimization solvers. The goal of optimization learning is to learn a model $\mathcal{M}_\theta$ that approximates the solution map $\Phi : \mathcal{X} \to \mathbb{R}^n$, which maps the instance parameter x to optimal solution $y^\star = \Phi(x)$ of (4.32). In particular, $\mathcal{M}_\theta$ must produce a feasible solution for all $x \in \mathcal{X}$.

Families of Problems. We consider two families of parametric optimization problems, each of which consists of 10000 problem instances and is split into train/valid/test set in the ratio 8:1:1. One family has linear constraints while the other has nonlinear constraints:

1. *Linearly-Constrained Non-Convex Programs (NCPs):* This family of problems, also considered by Donti et al. [2021] and Min and Azizan [2024], take the form

$$\begin{array}{ll} \min\limits_{y \in \mathbb{R}^n} & \frac{1}{2} y^T Q y + p^T \sin(y) \\ \text{s.t.} & Ay \leq b,\; Cy = x, \end{array}$$

   where $Q \in \mathbb{R}^{n \times n} \succ 0, p \in \mathbb{R}^n, A \in \mathbb{R}^{n_{\text{ineq}} \times n}, C \in \mathbb{R}^{n_{\text{eq}} \times n}$, and $b \in \mathbb{R}^{n_{\text{ineq}}}$ are constants, and $x \in \mathbb{R}^{n_{\text{eq}}}$ is the input.

2. *Quadratically Constrained Quadratic Programs (QCQPs):* This family of problems minimizes a convex quadratic objective subject to convex quadratic constraints and linear equality constraints:

$$\begin{array}{ll} \min\limits_{y \in \mathbb{R}^n} & \frac{1}{2} y^T Q y + p^T y \\ \text{s.t.} & y^T H_i y + g_i^T y \leq h_i,\; i = 1, \ldots, n_{\text{ineq}} \\ & Cy = x, \end{array}$$

   where $x \in \mathbb{R}^{n_{\text{eq}}}$ is the input and the other problem data are constant within the problem family.

Baselines. We compare `SnareNet` to three baselines:

1. `DC3` Donti et al. [2021] strictly enforces equality constraints by predicting free variables and solving dependent variables from equality constraints. The inequality constraint violation are iteratively reduced by a fixed number of gradient descent steps on the free variables.

2. `HardNet` Min and Azizan [2024] uses soft-epochs and strictly enforces linear constraints by (4.4). We compare to `HardNet` on NCPs.

3. `HProj` Liang et al. [2024] trains in two stages, rather than end-to-end: the first stage trains a homeomorphic map and the second stage trains a soft-constraint NN. At inference time, `HProj` uses bisection to project onto the feasible set.

Evaluation. Table 4.1 summarizes the six evaluation metrics in our experiments. Figures 4.4 to 4.5 presented below follow the same 2-by-3 layout. All experiments are run for 5 random seeds and the metrics are averaged over the seeds. The optimality gap is assessed relative to Gurobi Gurobi Optimization, LLC [2024]. The complete test results for the experiments in this section are provided in Appendix C.2.

	Optimality Gap	**Inequality Violation**	**Equality Violation**
Geometric Mean	$\underset{x\in\mathcal{X}_{\text{test}}}{\text{gmean}}(f_x(\hat{y}) - f_x^\star)$	$\underset{x\in\mathcal{X}_{\text{test}}}{\text{gmean}}\ \underset{j\in[n_{\text{ineq}}]}{\text{gmean}}$ (j-th inequality/equality violation)	
Maximum	$\max_{x\in\mathcal{X}_{\text{test}}} (f_x(\hat{y}) - f_x^\star)$	$\max_{x\in\mathcal{X}_{\text{test}}}\ \max_{j\in[n_{\text{ineq}}]}$ (j-th inequality/equality violation)	

Table 4.1: Six evaluation metrics and their layout. The function gmean$(\cdot)$ denotes the geometric mean over a set of numbers.

Feasibility Control and Low Variation along Training. Figure 4.4 shows training dynamics for end-to-end training methods (i.e., exclude `HProj`) on 833 validation instances of NCPs and QCQPs. `HardNet` exhibits larger seed-to-seed variability in optimality gap, while `DC3` shows large variability in inequality-constraint violation. In contrast, `SnareNet` is robust across random seeds after adaptive relaxation completes (at epoch 500, marked by vertical red dashed line). Earlier, variable violations are expected since `SnareNet` enforces relaxed constraints while the figures show violation of original constraints. Moreover, `SnareNet` successfully controls feasibility up to the specified tolerance.

Better Optimality. Figure 4.5 evaluates trained neural solvers on a separate test set for NCPs and QCQPs. `SnareNet` achieves optimality gaps at least one order of magnitude smaller than baseline methods while maintaining feasibilitywithin the prescribed tolerance `tol`. `HProj` exhibits substantial variation in optimality gap across random seeds because its projection step is applied post-training and is not integrated into the learning objective, which can degrade optimality.

λ Improves Optimality Even for Linear Constraints. `SnareNet` specializes to `HardNet` when $\lambda = 0$ and adaptive relaxation is replaced by soft epochs for training. Setting $\lambda > 0$ requires more

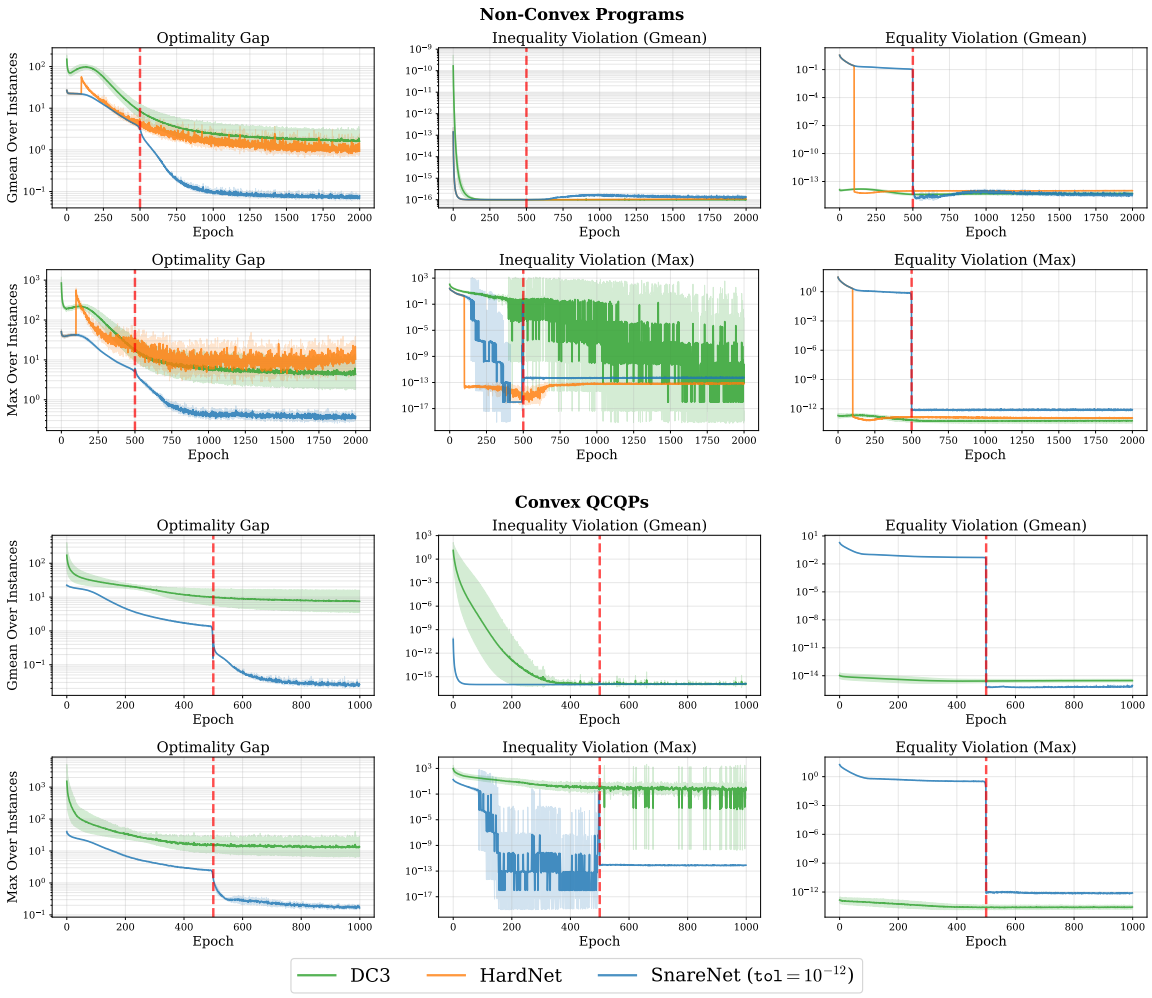


Figure 4.4: Training dynamics on 833 validation instances of NCPs and QCQPs. Shaded region indicate the standard deviation across seeds.

than one iteration to enforce feasibility but can improve the optimality gap by orders of magnitude. See Figure 4.6.

`SnareNet` Handles Many Constraints. `HardNet` requires the linear constraint matrix A to have full row rank and is thus limited to at most $2n$ constraints, where n is the output dimension. `SnareNet` overcomes this limitation through LM-regularization. Figure 4.7 presents test results on QCQPs with $n = 100$ variables, $n_{\text{eq}} = 50$ equality constraints, and $n_{\text{ineq}} \in \{10, 50, 100\}$ inequality constraints. `SnareNet` consistently produces feasible solutions within tolerance 10^{-4} across all test instances. In contrast, `DC3` achieves feasibility on only 80% of test instances on average and exhibits high sensitivity to random initialization. `HProj` shows substantial variation in feasibility rates for

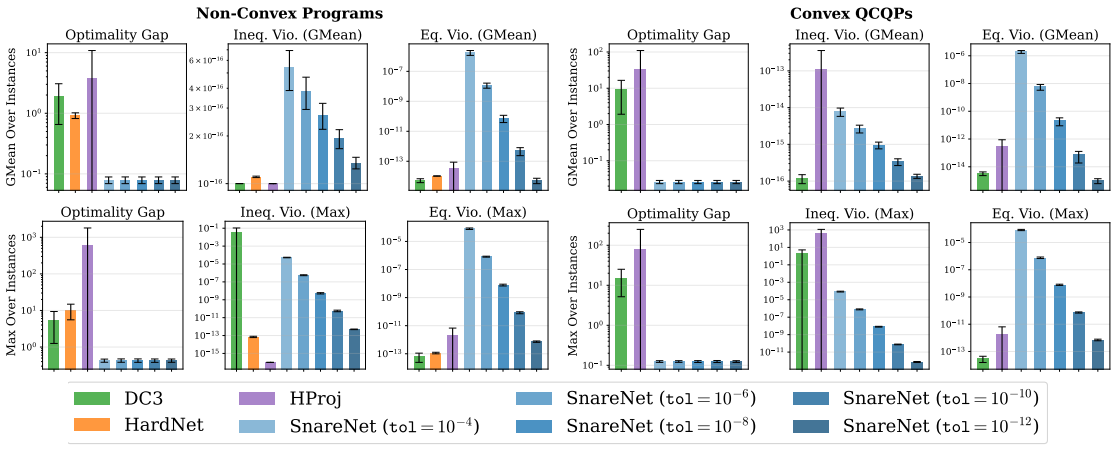


Figure 4.5: Evaluation metrics on 833 test instances of NCPs and QCQPs. Black error bars indicate the standard deviation across seeds.

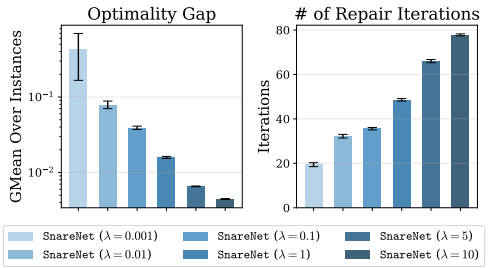


Figure 4.6: Optimality gap and repair iterations on 833 test instances of NCPs. Bars represent the mean across 5 random seeds, and black error bars indicate the standard deviation across seeds.

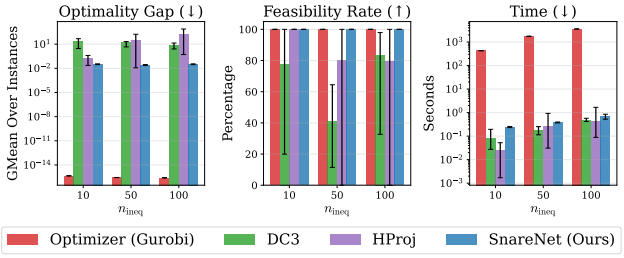


Figure 4.7: Performance comparison across methods for QCQPs with various number of inequality constraints. Bars represent the mean across 5 random seeds, and black error bars indicate the minimum and maximum values.

problems with $n_{\text{ineq}} = 50$ and 100 inequality constraints. Inference time for the classical solver Gurobi is significantly slower than neural solvers and scales rapidly with the number of inequality constraints.

4.5.2 Neural Control Policies

Neural control policies use deep neural networks to learn mappings from system states x to control actions u, either as standalone policies or to enhance traditional control methods. In this experiment, we train a neural network policy $\pi_\theta(x)$ to control a unicycle system, avoiding obstacle collisions by enforcing safety constraints Min and Azizan [2024], Tayal et al. [2024]. The neural policy is constructed as the sum of a nominal controller $\pi_{\text{nom}}(x)$, designed without obstacle awareness, and

a learned correction term $\mathcal{M}_\theta(x)$:

$$\pi_\theta(x) := \pi_{\text{nom}}(x) + \mathcal{M}_\theta(x).$$

The system state is $x(t) = (p_x(t), p_y(t), \theta(t), v(t), w(t)) \in \mathbb{R}^5$ at time t, where p_x, p_y are the coordinates, θ is the heading angle, v is the linear velocity, and w is the angular velocity. The control $u(t) = (a(t), \alpha(t)) \in \mathbb{R}^2$ includes linear acceleration a and angular acceleration α. The control changes the system state by the linear dynamics: $\dot{x} = F(x) + Gu$, where $F : \mathbb{R}^5 \to \mathbb{R}^5$ and $G \in \mathbb{R}^{5\times 2}$ are defined in Appendix C.1. Our neural policy is trained by minimizing a quadratic cost over a finite time horizon:

$$\Delta t \sum_{i=0}^{n_t} x(t_i)^T Q x(t_i) + c\|\pi_\theta(x(t_i))\|^2,$$

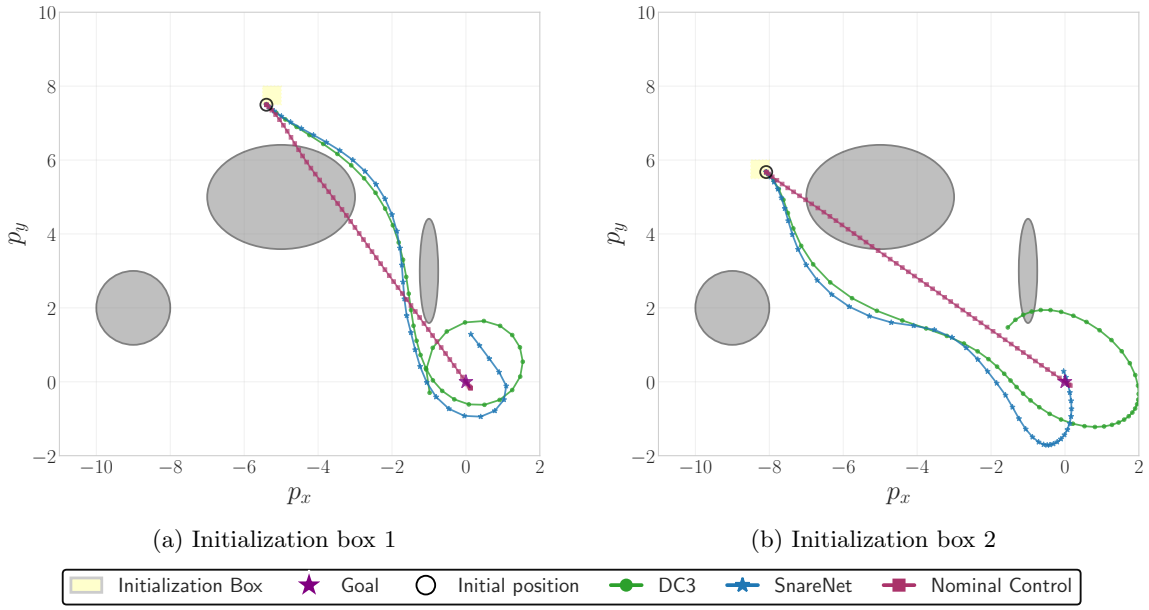
where $Q = \text{diag}(100, 100, 0, 0.1, 0.1)$ penalizes deviations from the target position $(0, 0)$ and velocity, $c = 0.1$ weights the control effort, and $n_t = 10$ steps for trajectories.

We use *control barrier functions* (CBFs) Ames et al. [2019], denoted by h, to parametrize the safe set: $h(x) \geq 0$ indicates x is a safe state. Each obstacle is represented by a distinct CBF. See Appendix C.1 for CBFs used in experiment. Collision-free trajectories are guaranteed by enforcing

$$\nabla h_j(x)^T (F(x) + G\pi_\theta(x)) \geq -\alpha h_j(x) \tag{4.33}$$

for each obstacle with CBF h_j and some $\alpha > 0$. The number of constraints scales linearly with the number of obstacles. `HardNet` can avoid at most two obstacles as it requires full row rank constraints and the system has only two controls, whereas `SnareNet` handles arbitrarily many obstacles.

For training, we uniformly sample initial states from two boxes in the state space, differing only in position: Box 1 with positions $(p_x, p_y) \in [-5.0, -5.5] \times [7.5, 8.0]$ and Box 2 with positions $(p_x, p_y) \in [-8.5, -8.0] \times [5.5, 6.0]$. Both boxes share the same heading angle range $\theta \in [-\frac{\pi}{4}, -\frac{\pi}{8}]$ and zero initial velocities ($v = w = 0$). The learned neural policies are shown in Figure 4.8 for test samples from both boxes. Table 4.2 shows the detailed evaluation statistics: `SnareNet` achieves a 99.4% feasibility rate on the test set compared to `DC3`'s 64.9% feasibility rate, demonstrating the effectiveness of `SnareNet` in enforcing safety constraints in neural control policies. Since we use a higher order CBF, a trajectory may violate a constraint without visually colliding with any obstacle. For instance, the DC3 trajectory in Figures 4.8a and 4.8b.



(a) Initialization box 1

(b) Initialization box 2

Figure 4.8: Simulated unicycle trajectories from a test sample inside the initialization region.

Table 4.2: Evaluation statistics on 84 CBF test instances.

Method	Dataset	Objective	# Feasible (1e-4)	Max Ineq. Error	GMean Ineq. Error
DC3	Box 1	3656	58	6.555	6.785×10^{-15}
SnareNet	Box 1	4060	84	4.21×10^{-8}	1.763×10^{-14}
DC3	Box 2	4269	51	4.17	1.913×10^{-14}
SnareNet	Box 2	5226	83	0.7483	4.038×10^{-14}

4.6 Concluding Remarks

We introduce `SnareNet`, a practical framework to enforce nonlinear input-dependent constraints in NNs. Our key innovations include regularization for stable optimization, Newton updates for efficient convergence, explicit infeasibility tolerances for controllable constraint enforcement, and adaptive relaxation that progressively tightens the constraints. Our empirical evaluation demonstrates that `SnareNet` consistently produces feasible solutions with better objective values across optimization learning tasks and neural control policies. An important direction for future work is to understand how hard constraints, relaxed constraints, and soft constraints fundamentally alter the learnability of deep neural networks.

Conclusions

Preconditioning can provably and empirically accelerate optimization algorithms, but the right preconditioner needs careful design tailored to the problem structure and algorithmic setting. `Nys-IP-PMM` constructs a randomized low-rank preconditioner via the Nystr"om approximation for the normal equation arising in interior point methods. When the data matrix exhibits rapid spectral decay, a small sketch captures the dominant curvature directions at a fraction of the cost of direct factorization. `HDM` casts preconditioner selection as an online learning problem, adaptively updating the preconditioner from carefully designed feedback computed during the iterate update. This formulation characterizes the warm-up instability of hypergradient methods as online learning regret rather than true divergence, and once the regret is overcome, the preconditioner adapts to local curvature, enabling faster convergence. `SnareNet` uses a Newton-motivated preconditioner for the constraint residual, enabling a differentiable repair layer to enforce feasibility in few iterations and effectively improve the optimality. These adaptive preconditioners share a common design philosophy: they should be computationally cheap, memory efficient, or well-adapted to local curvature. The convergence analysis serves not only to certify the effectiveness of the chosen preconditioner, but also to provide practitioners with actionable guidance for adapting each algorithm to specific problem.

Each method in this thesis also surfaces an open challenge for future work. Even with effective preconditioners, how to stabilize the iterative solver within `IPMs` at very large scale remains a challenge. Online learning based adaptive preconditioning requires a warm-up phase to adjust the preconditioner, and reducing this warm-up time is an important direction for future research. The key challenge for iterative repair layers is to reduce the cost per repair iteration while maintaining a high-quality preconditioner to enable fast convergence to a feasible solution. As optimization problems continue to grow in scale and complexity, algorithms need to be more customized for each application, and adaptive preconditioning remains a productive direction.

Appendix A

Supplement for Chapter 2

A.1 An example of QP with matrix-free constraint

This section provides a concrete example of a separable QP problem with a matrix-free constraint that arises in the context of filter design. The finite impulse response (FIR) filter design problem consists of determining a finite set of coefficients $h_0, \ldots, h_{N-1}$ that fully define the impulse response of the filter: $y[n] = \sum_{k=0}^{N-1} h_k s[n-k]$, where $s[n]$ is the input signal and $y[n]$ is the output signal. The values of the impulses are constrained within a bound

$$l \leq h_i \leq u \quad \text{for } i = 0, \ldots, N-1.$$

An energy-efficient filter minimizes the weighted energy $\frac{1}{2}\sum_{k=0}^{N-1} q_i h_k^2$ subject to the constraints on the frequency response. Linear-phase filters (i.e., $h_k = h_{N-k-1}$) are preferred in many signal processing applications since such filters prevent phase distortion. The frequency response of a linear-phase filter can be determined in terms of $\{h_k\}$ as

$$H(\omega) = \sum_{k=0}^{n-1} (2 - \delta_{k,n-1}) \cos((n-1-k)w) h_k = h_{n-1} + 2\sum_{j=1}^{n-1} \cos(jw) h_{n-1-j}, \tag{A.1}$$

where $n := \lfloor \frac{N-1}{2} \rfloor + 1$ is the number of coefficients decision variables, $\delta_{a,b}$ is the Kronecker delta (i.e., $\delta_{a,b} = 1$ if $a = b$; otherwise, $\delta_{a,b} = 0$), and $\omega \in \mathbb{R}$ is the angular frequency. By (A.1), the frequency response of the filter at frequencies $\omega_k = \frac{(2k+1)\pi}{2n}$ for $k = 0, \ldots, n-1$ can be expressed as the Discrete Cosine Transform (DCT) of the time-reversed sequence $x_j := h_{n-1-j}$ as

$$H(\omega_k) = x_0 + 2\sum_{j=1}^{n-1} \cos\left(\tfrac{\pi}{2n}(2k+1)j\right) x_j = 2Fx,$$

where $F \in \mathbb{R}^{n\times n}$ is the DCT matrix. The pass-band of the filter is the region in the frequency domain where we want the signal to pass through unaltered; the stop-band is the region where we want to suppress or eliminate the signal. The pass-band gain and stop-band gain of the filter are the values the frequency response $H(\omega)$ over the band, which is often normalized to 1.0 and 0 respectively. In practical design, small variations (or ripples) σ_p and σ_s around these nominal values are tolerated [Aksoy et al., 2014, Yli-Kaakinen and Saramaki, 2001, Gustafsson and Wanhammar, 2002]:

$$\begin{aligned} 1-\sigma_p \le H(\omega_k) \le 1+\sigma_p \qquad & \text{for } k \le m \text{ (pass-band)};\\ -\sigma_s \le H(\omega_k) \le \sigma_s \qquad & \text{for } k > m \text{ (stop-band)}, \end{aligned}$$

where we assume ω_k lies in pass-band for $k \le m$; and in stop-band for $k > m$. We arrive at a QP with matrix-free constraint and diagonal $Q = \operatorname{diag}(q_0, \ldots, q_{n-1})$:

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\textstyle\sum_{i=0}^{n-1} q_i x_i^2 \\ \text{subject to} \quad & 1-\sigma_p \le 2(Fx)_k \le 1+\sigma_p \quad \text{for } k \le m;\\ & -\sigma_s \le 2(Fx)_k \le \sigma_s \qquad\quad \text{for } k > m;\\ & l \le x \le u. \end{aligned}$$

Importantly, the DCT matrix F can be applied to a vector in $\mathcal{O}(n \log n)$ time using the fast cosine transform, which is faster than the time required to multiply by a generic $n \times n$ matrix. Hence, it is preferred to avoid instantiating the entries of F but instead access F through matrix-vector product only.

A.2 Implementation Details of `Nys-IP-PMM`

`Nys-IP-PMM` deviates from the theory in order to improve the computational efficiency, following [Pougkakiotis and Gondzio, 2021] in the following two main aspects: First, we do not limit to $\rho_k = \delta_k = \mu_k$ as theory does. The proximal parameters ρ_k and δ_k are independently updated following the suggestions from [Pougkakiotis and Gondzio, 2021, Algorithm PEU]. Second, the iterates of the method are not required to lie in the neighborhood as in (2.18) for efficiency.

The followings provide more details/explanations for the implementation of `Nys-IP-PMM`. Appendix A.2.1 derives the Newton system to be solved at each `Nys-IP-PMM` iteration for QP instance taking the form ($\tilde{\text{P}}$)–($\tilde{\text{D}}$) and the resulting normal equations. Appendix A.2.2 details the construction of practical initial point.

A.2.1 Derivations of Newton system and equations (2.46)–(2.50)

Given QP in the form ($\tilde{\text{P}}$)–($\tilde{\text{D}}$), `IP-PMM` solves a sequence of subproblems taking the form (2.45). Introducing the logarithmic barrier function to enforce the constraints $x^{\mathcal{I}} \geq 0$ and $0 \leq x^{\mathcal{J}} \leq u^{\mathcal{J}}$ in (2.45), the Lagrangian to minimize becomes

$$\mathcal{L}(x) = \frac{1}{2}x^T Q x + c^T x + (\lambda^k)^T(b - Ax) + \frac{1}{2\delta_k}\|Ax - b\|_2^2 + \frac{\rho_k}{2}\|x - \zeta^k\|^2 \\ - \mu_k \sum_{j \in \mathcal{IJ}} \ln x_j - \mu_k \sum_{j \in \mathcal{J}} \ln(u_j - x_j).$$

Setting $\nabla_x \mathcal{L}(x) = 0$ and introducing the new variables

$$y = \lambda^k - \frac{1}{\mu_k}(Ax - b), \qquad z_j = \begin{cases} \mu_k x_j^{-1}, & \text{if } j \in \mathcal{IJ}; \\ 0, & \text{if } j \in \mathcal{F}, \end{cases}$$

$$w_j = \begin{cases} u_j - x_j, & \text{if } j \in \mathcal{J}; \\ 0, & \text{if } j \in \mathcal{IF}, \end{cases} \qquad s_j = \begin{cases} \mu_k w_j^{-1}, & \text{if } j \in \mathcal{J}; \\ 0, & \text{if } j \in \mathcal{IF}, \end{cases}$$

we obtain the following (non-linear) system to solve:

$$\begin{bmatrix} c + Qx - A^T y - z + s + \rho_k(x - \zeta^k) \\ Ax - b + \delta_k(y - \lambda^k) \\ x^{\mathcal{J}} + w^{\mathcal{J}} - u^{\mathcal{J}} \\ \operatorname{diag}(x^{\mathcal{IJ}}) z^{\mathcal{IJ}} - \mu_k \mathbb{1}_{|\mathcal{IJ}|} \\ \operatorname{diag}(w^{\mathcal{J}}) s^{\mathcal{J}} - \mu_k \mathbb{1}_{|\mathcal{J}|} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{A.2}$$

Applying Newton's method to (A.2) yields the Newton system taking the form:

$$\begin{bmatrix} -(Q + \rho_k I_n) & A^T & 0 & I_n & -I_n \\ A & \delta_k I_m & 0 & 0 & 0 \\ E_{\mathcal{J}} & 0 & E_{\mathcal{J}} & 0 & 0 \\ E_{\mathcal{IJ}} Z_k & 0 & 0 & E_{\mathcal{IJ}} X_k & 0 \\ 0 & 0 & E_{\mathcal{J}} S_k & 0 & E_{\mathcal{J}} W_k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta w^k \\ \Delta z^k \\ \Delta s^k \end{bmatrix} = \begin{bmatrix} r_d \\ r_p \\ (r_u)_{\mathcal{J}} \\ (r_{xz})_{\mathcal{IJ}} \\ (r_{ws})_{\mathcal{J}} \end{bmatrix} \tag{A.3}$$

where $E_{\mathcal{S}} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{S}|}$ is the projection matrix such that $E_{\mathcal{S}} x = x^{\mathcal{S}}$ for all $x \in \mathbb{R}^n$, the uppercase letters X^k, Z^k, W^k, S^k represent diagonal matrices with diagonal entries corresponding to the iterates in the associated lowercase letters, and $r_d, r_p, r_u, r_{xz}, r_{ws}$ are appropriate RHS vectors. By eliminating the variables $\Delta x^k, \Delta w^k, \Delta z^k$, and Δs^k, the Newton system (A.3) simplifies to the normal equations (2.46) of size m, and four closed-form formulae (2.47)–(2.50).

A.2.2 Construction for initial point

The construction is based on the development of initial point for original `IP-PMM` in [Pougkakiotis and Gondzio, 2021]. We first construct a candidate point by ignoring the non-negative constraints and solving the primal and dual equality constraints $Ax = b$ and $-Qx + A^T y + z - s = c$ in (P̃)–(D̃). We force primal candidate $\tilde{x}$ satisfy $Ax = b$ while centering around $u/2$, and solve the dual candidate $\tilde{y}$ from the least squares problem $\min_y \| - Q\tilde{x} + A^T y - c\|_2$, ignoring z and s in dual constraint meanwhile. Then dual candidates $\tilde{z}$ and $\tilde{s}$ are chosen such that dual constraint $-Q\tilde{x} + A^T\tilde{y} + \tilde{z} - \tilde{s} = c$ is satisfied. The primal candidate $\tilde{w}$ is set as $\tilde{w}_{\mathcal{J}} = u_{\mathcal{J}} - \tilde{x}_{\mathcal{J}}$ and $\tilde{w}_{\mathcal{IF}} = 0$. In summary, the candidate point is:

$$\begin{aligned}
&\tilde{x} = \frac{u}{2} + A^T(AA^T)^{-1}\left(b - \frac{1}{2}Au\right), \quad \tilde{y} = (AA^T)^{-1}A(c + Q\tilde{x}),\\
&\tilde{z}_{\mathcal{J}} = \frac{1}{2}\left(c - A^T\tilde{y} + Q\tilde{x}\right)_{\mathcal{J}}, \quad \tilde{z}_{\mathcal{IF}} = \left(c - A^T\tilde{y} + Q\tilde{x}\right)_{\mathcal{IF}},\\
&\tilde{s}_{\mathcal{J}} = -\tilde{z}_{\mathcal{J}}, \quad \tilde{s}_{\mathcal{IF}} = 0, \quad \tilde{w}_{\mathcal{J}} = u_{\mathcal{J}} - \tilde{x}_{\mathcal{J}}, \quad \tilde{w}_{\mathcal{IF}} = 0.
\end{aligned}$$

However, to ensure the stability and efficiency, we regularize the matrix AA^T as in [Pougkakiotis and Gondzio, 2021] and use Nyström PCG to solve for the systems in $\tilde{x}$ and $\tilde{y}$, for which the regularization parameter is taken as $\delta = 10$.

Next, to guarantee the positivity of $x_{\mathcal{IJ}}$, $z_{\mathcal{IJ}}$, $w_{\mathcal{J}}$, and $s_{\mathcal{J}}$, we compute the following quantities:

$$\begin{aligned}
&\delta_p = \max\{-1.5\min(\tilde{x}_{\mathcal{IJ}}), -1.5\min(\tilde{w}_{\mathcal{J}}), 0\},\\
&\delta_d = \max\{-1.5\min(\tilde{z}_{\mathcal{IJ}}), -1.5\min(\tilde{s}_{\mathcal{J}}), 0\},\\
&\gamma_{xz} = (\tilde{x}_{\mathcal{IJ}} + \delta_p \mathbb{1}_{|\mathcal{IJ}|})^T(\tilde{z}_{\mathcal{IJ}} + \delta_d \mathbb{1}_{|\mathcal{IJ}|}),\\
&\gamma_{ws} = (\tilde{w}_{\mathcal{J}} + \delta_p \mathbb{1}_{|\mathcal{J}|})^T(\tilde{s}_{\mathcal{J}} + \delta_d \mathbb{1}_{|\mathcal{J}|}),\\
&\tilde{\delta}_p = \delta_p + 0.5\frac{\gamma_{xz} + \gamma_{ws}}{\sum_{\ell\in\mathcal{IJ}}(\tilde{z}^\ell + \delta_d) + \sum_{\ell\in\mathcal{J}}(\tilde{s}^\ell + \delta_d)},\\
&\tilde{\delta}_d = \delta_d + 0.5\frac{\gamma_{xz} + \gamma_{ws}}{\sum_{\ell\in\mathcal{IJ}}(\tilde{x}^\ell + \delta_d) + \sum_{\ell\in\mathcal{J}}(\tilde{w}^\ell + \delta_d)}.
\end{aligned}$$

Finally, we set the final initial point as:

$$\begin{aligned}
&y^0 = \tilde{y}, \quad x^0_{\mathcal{IJ}} = \tilde{x}_{\mathcal{IJ}} + \tilde{\delta}_p \mathbb{1}_{|\mathcal{IJ}|}, \quad x^0_{\mathcal{F}} = \tilde{x}_{\mathcal{F}},\\
&w^0_{\mathcal{J}} = \tilde{w}_{\mathcal{J}} + \tilde{\delta}_p \mathbb{1}_{|\mathcal{J}|}, \quad w^0_{\mathcal{IF}} = 0,\\
&z^0_{\mathcal{IJ}} = \tilde{z}_{\mathcal{IJ}} + \tilde{\delta}_d \mathbb{1}_{|\mathcal{IJ}|}, \quad z^0_{\mathcal{F}} = 0, \quad s^0_{\mathcal{J}} = \tilde{s}_{\mathcal{J}} + \tilde{\delta}_d \mathbb{1}_{|\mathcal{J}|}, \quad s^0_{\mathcal{IF}} = 0.
\end{aligned} \tag{A.4}$$

A.2.3 Stepsizes

The primal and dual stepsizes are chosen as in standard IPMs, which ensure that the updated iterate remains in the non-negative orthant after transitioning from the current iterate x^k, z^k, w^k, and s^k. Given the search directions Δx^k, Δz^k, Δw^k, and Δs^k, denote the sets of indices for negative components by

$$I_x = \{i \in \mathcal{I} \cup \mathcal{J} \mid \Delta x_i^k < 0\}; \quad I_w = \{i \in \mathcal{J} \mid \Delta w_i^k < 0\};$$
$$I_z = \{i \in \mathcal{I} \cup \mathcal{J} \mid \Delta z_i^k < 0\}; \quad I_s = \{i \in \mathcal{J} \mid \Delta s_i^k < 0\}.$$

The primal and dual stepsizes are defined as follows:

$$\alpha_p = 0.995 \min\left\{1, \min_{i \in I_x}\left\{-\frac{x_i^k}{\Delta x_i^k}\right\}, \min_{i \in I_w}\left\{-\frac{w_i^k}{\Delta w_i^k}\right\}\right\}, \tag{A.5}$$

$$\alpha_d = 0.995 \min\left\{1, \min_{i \in I_z}\left\{-\frac{z_i^k}{\Delta z_i^k}\right\}, \min_{i \in I_s}\left\{-\frac{s_i^k}{\Delta s_i^k}\right\}\right\}. \tag{A.6}$$

A.3 Experiment Details for `Nys-IP-PMM`

A.3.1 QP formulation for portfolio optimization

Consider a portfolio optimization problem, which aims at determining the asset allocation to maximize risk-adjusted returns while constraining correlation with market indexes or competing portfolios:

$$\begin{array}{ll} \text{minimize} & -r^T x + \gamma x^T \Sigma x \\ \text{subject to} & Mx \leq u,\ \mathbb{1}_n^T x = 1,\ x \geq 0, \end{array} \tag{A.7}$$

where variable $x \in \mathbb{R}^n$ represents the portfolio, $r \in \mathbb{R}^n$ denotes the vector of expected returns, $\gamma > 0$ denotes the risk aversion parameter, $\Sigma \in \mathbb{S}_n^+(\mathbb{R})$ represents the risk model covariance matrix, each row of $M \in \mathbb{R}^{d \times n}$ represents another portfolio, and $u \in \mathbb{R}^d$ upper bounds the correlations.

We assume a factor model for the covariance matrix $\Sigma = FF^T + D$, where $F \in \mathbb{R}^{n \times s}$ is the factor loading matrix and $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix representing asset-specific risk. By replacing Σ with $FF^T + D$ in (A.7) and introducing a new variable $y = F^T x$, we write an equivalent problem in variables x and y:

$$\begin{array}{ll} \text{minimize} & -\gamma^{-1} r^T x + x^T D x + y^T y \\ \text{subject to} & y = F^T x,\ Mx \leq u,\ \mathbb{1}_n^T x = 1,\ x \geq 0,\ y : \text{free}. \end{array} \tag{A.8}$$

Problem (A.8) can be transformed into form ($\tilde{\text{P}}$) via standard techniques.

A.3.2 Support vector machine (SVM) formulations in QP

The linear support vector machine (SVM) problem solves a binary classification task on n samples with d features [Deisenroth et al., 2020, Chapter 12]. Let $X \in \mathbb{R}^{d\times n}$ be a feature matrix whose columns are the attribute vectors $x_i \in \mathbb{R}^d$ associated with the i-th sample, $i = 1, \ldots, n$, and let $y_i \in \{-1, 1\}$ be the corresponding binary classification label. The dual linear SVM with ℓ_1-regularization can be formulated as a convex quadratic program [Fine and Scheinberg, 2001, Woodsend and Gondzio, 2011, Gondzio and Grothey, 2009, Čiegis et al., 2009, Woodsend and Gondzio, 2009]:

$$\begin{array}{ll} \text{minimize} & \dfrac{1}{2}v^T v - \displaystyle\sum_{i=1}^{n} p_i \\ \text{subject to} & v - X\,\mathrm{diag}(y)p = 0,\ y^T p = 0, \\ & v\text{: free},\ 0 \le p_i \le \tau,\ i = 1, \ldots, n, \end{array} \tag{A.9}$$

where $v \in \mathbb{R}^d$ and $p \in \mathbb{R}^n$ are optimization variables, and $\tau > 0$ is the penalty parameter for misclassification. The dual SVM problem (A.9) can be formulated into form ($\tilde{\mathrm{P}}$) by setting

$$x = \begin{bmatrix} v \\ p \end{bmatrix} \in \mathbb{R}^{d+n},\ Q = \begin{bmatrix} I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(d+n)\times(d+n)},$$

$$A = \begin{bmatrix} I_d & -X\,\mathrm{diag}(y) \\ \mathbf{0} & y^T \end{bmatrix} \in \mathbb{R}^{(d+1)\times(d+n)},\ c = \begin{bmatrix} \mathbf{0}_d \\ \mathbb{1}_n \end{bmatrix} \in \mathbb{R}^{d+n},\ b = \mathbf{0}_{d+1},$$

$$\mathcal{F} = \{1, 2, \ldots, d\},\ \mathcal{I} = \emptyset,\ \mathcal{J} = \{d+1, \ldots, d+n\},\ u^{\mathcal{J}} = \tau\mathbb{1}_{|\mathcal{J}|}.$$

Note that the constraint matrix A has a dense $(1, 2)$-block if the feature matrix X is dense.

A.3.3 Regularization parameters in the experiments

Table A.1 and Table A.2 present the primal and dual regularization parameters ρ_k and δ_k for `Nys-IP-PMM` on problems in Section 2.4.

	Portfolio (section 5.1)		**CIFAR10_1000 (section 5.2.2)**		**RNASeq**		**SensIT**		**sector**	
k	ρ_k	δ_k	ρ_k	δ_k	ρ_k	δ_k	ρ_k	δ_k	ρ_k	δ_k
0	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
1	7.12×10^{-1}	7.12×10^{-1}	3.81	1.71	2.70	4.03×10^{-2}	3.56	1.33	2.96	2.96
2	1.02×10^{-1}	1.02×10^{-1}	1.82	3.67×10^{-1}	2.64×10^{-2}	3.95×10^{-4}	8.03×10^{-1}	3.00×10^{-1}	1.13	2.15×10^{-1}
3	4.77×10^{-2}	2.03×10^{-2}	1.24	2.50×10^{-1}	5.21×10^{-3}	7.77×10^{-5}	2.42×10^{-1}	9.05×10^{-2}	8.98×10^{-1}	1.48×10^{-1}
4	2.37×10^{-2}	4.95×10^{-3}	8.19×10^{-1}	1.65×10^{-1}	2.24×10^{-3}	3.34×10^{-5}	1.53×10^{-1}	5.73×10^{-2}	7.14×10^{-1}	1.02×10^{-1}
5	3.40×10^{-3}	7.10×10^{-4}	5.87×10^{-1}	1.19×10^{-1}	1.63×10^{-3}	1.98×10^{-5}	8.99×10^{-2}	3.36×10^{-2}	6.07×10^{-1}	8.71×10^{-2}
6	1.91×10^{-4}	3.99×10^{-5}	1.22×10^{-1}	2.45×10^{-2}	1.39×10^{-3}	1.68×10^{-5}	4.95×10^{-2}	1.85×10^{-2}	3.27×10^{-1}	4.70×10^{-2}
7	2.86×10^{-5}	5.97×10^{-6}	5.66×10^{-2}	1.14×10^{-2}	1.16×10^{-3}	1.40×10^{-5}	2.90×10^{-2}	1.08×10^{-2}	2.18×10^{-1}	3.13×10^{-2}
8	5.13×10^{-6}	1.07×10^{-6}	1.31×10^{-2}	2.64×10^{-3}	1.13×10^{-3}	1.35×10^{-5}	1.59×10^{-2}	5.94×10^{-3}	1.97×10^{-1}	2.82×10^{-2}
9	2.17×10^{-6}	1.42×10^{-7}	2.84×10^{-3}	5.74×10^{-4}	1.10×10^{-3}	1.29×10^{-5}	9.83×10^{-3}	3.67×10^{-3}	1.21×10^{-1}	1.74×10^{-2}
10	1.57×10^{-6}	1.03×10^{-7}	4.72×10^{-4}	9.53×10^{-5}	1.08×10^{-3}	1.28×10^{-5}	5.04×10^{-3}	1.88×10^{-3}	8.03×10^{-2}	1.15×10^{-2}
11	3.13×10^{-7}	2.05×10^{-8}	5.39×10^{-6}	1.09×10^{-6}	7.86×10^{-4}	9.28×10^{-6}	2.82×10^{-3}	1.05×10^{-3}	5.09×10^{-2}	7.31×10^{-3}
12	1.87×10^{-8}	1.22×10^{-9}	4.32×10^{-8}	8.73×10^{-9}	5.76×10^{-4}	6.80×10^{-6}	1.45×10^{-3}	5.43×10^{-4}	2.18×10^{-2}	3.13×10^{-3}
13	5.00×10^{-10}	5.00×10^{-10}	5.00×10^{-10}	2.95×10^{-9}	3.31×10^{-4}	3.91×10^{-6}	6.73×10^{-4}	2.52×10^{-4}	9.95×10^{-3}	1.43×10^{-3}
14	5.00×10^{-10}	5.00×10^{-10}	5.00×10^{-10}	5.00×10^{-10}	5.95×10^{-5}	7.02×10^{-7}	3.26×10^{-4}	1.22×10^{-4}	5.16×10^{-3}	7.40×10^{-4}
15	5.00×10^{-10}	5.00×10^{-10}	5.00×10^{-10}	5.00×10^{-10}	–	–	1.55×10^{-4}	5.78×10^{-5}	1.79×10^{-3}	2.56×10^{-4}
16	5.00×10^{-10}	5.00×10^{-10}	–	–	–	–	–	–	3.22×10^{-4}	4.62×10^{-5}
17	5.00×10^{-10}	5.00×10^{-10}	–	–	–	–	–	–	–	–

Table A.1: Regularization parameters for for experiments in Section 2.4.

	CIFAR10		**STL10**		**arcene**		**dexter**	
k	ρ_k	δ_k	ρ_k	δ_k	ρ_k	δ_k	ρ_k	δ_k
0	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
1	4.18	2.27	2.72	7.29×10^{-2}	2.85	2.66×10^{-1}	2.85	2.66×10^{-1}
2	7.35×10^{-1}	3.99×10^{-1}	2.18×10^{-2}	5.83×10^{-4}	1.42×10^{-2}	1.33×10^{-3}	1.43×10^{-2}	1.33×10^{-3}
3	4.01×10^{-1}	2.17×10^{-1}	4.29×10^{-3}	1.15×10^{-4}	8.07×10^{-5}	7.53×10^{-6}	1.28×10^{-4}	1.19×10^{-5}
4	2.41×10^{-1}	1.31×10^{-1}	2.37×10^{-3}	6.34×10^{-5}	3.54×10^{-6}	3.31×10^{-7}	1.51×10^{-5}	1.41×10^{-6}
5	1.02×10^{-1}	5.55×10^{-2}	1.40×10^{-3}	2.43×10^{-5}	7.72×10^{-7}	7.20×10^{-8}	–	–
6	4.51×10^{-2}	2.44×10^{-2}	6.85×10^{-4}	1.19×10^{-5}	–	–	–	–
7	2.24×10^{-2}	1.22×10^{-2}	2.88×10^{-4}	5.02×10^{-6}	–	–	–	–
8	9.58×10^{-3}	5.20×10^{-3}	1.10×10^{-4}	1.92×10^{-6}	–	–	–	–
9	3.55×10^{-3}	1.93×10^{-3}	3.69×10^{-5}	6.43×10^{-7}	–	–	–	–
10	1.28×10^{-3}	6.92×10^{-4}	3.55×10^{-6}	6.20×10^{-8}	–	–	–	–
11	2.53×10^{-4}	1.37×10^{-4}	2.54×10^{-7}	4.43×10^{-9}	–	–	–	–
12	3.48×10^{-5}	1.89×10^{-5}	7.18×10^{-9}	5.00×10^{-10}	–	–	–	–

Table A.2: Regularization parameters for for experiments in Section 2.4 (continued).

A.3.4 Spectrums of AA^T and N_k

Figure A.1 illustrates the eigenvalue distributions of AA^T and N_k for SVM problem in Section 2.4.2. Our `Nys-IP-PMM` converges at $k = 12$ for a relative tolerance $\epsilon = 10^{-6}$; and at $k = 15$ for $\epsilon = 10^{-8}$. As the algorithm progresses, the normal equation matrix N_k becomes increasingly ill-conditioned,

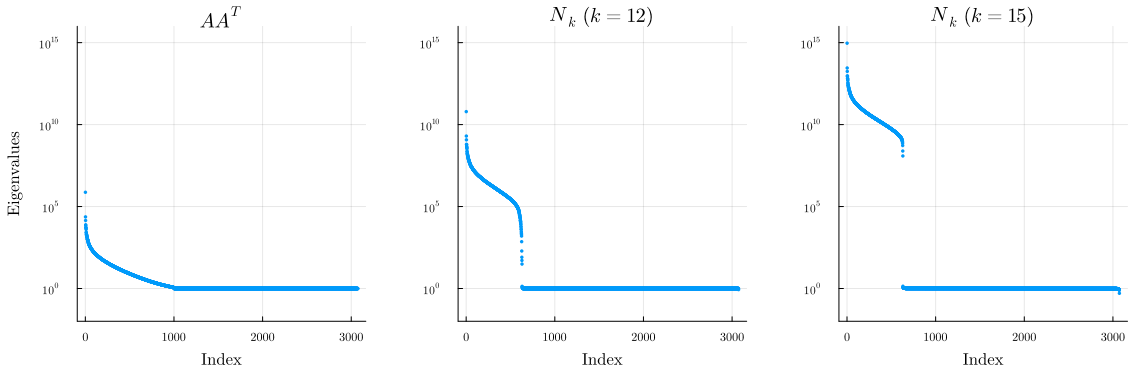


Figure A.1: Distribution of eigenvalues for AA^T and N_k at different `IP-PMM` iterations of the SVM problem formed from 1000 samples of CIFAR10.

which is typical for IPMs. For this problem, the rapid decay of the top eigenvalues of N_k suggests the suitability and success of the Nyström preconditioner.

Appendix B

Supplement for Chapter 3

B.1 Proof of Theorem 3.12

The proof of Theorem 3.12 relies on the following auxiliary results.

Lemma B.1. *Under* **A1** *to* **A4**, $h_x(P) - \inf_{Q\in\mathbb{R}^{n\times n}} h_x(Q) \leq \frac{1}{2\mu}(LD+1)^2$.

Proof. Note that $h_x(P) = \frac{f(x-P\nabla f(x))-f(x)}{\|\nabla f(x)\|^2} \geq \frac{f(x^\star)-f(x)}{\|\nabla f(x)\|^2}$ for all $P \in \mathcal{P}$, we deduce that

$$
\begin{aligned}
h_x(P) - \inf_{Q\in\mathbb{R}^{n\times n}} h_x(Q) &\leq \frac{f(x-P\nabla f(x))-f(x)}{\|\nabla f(x)\|^2} - \frac{f(x^\star)-f(x)}{\|\nabla f(x)\|^2} && \text{(B.1)}\\
&= \frac{f(x-P\nabla f(x))-f(x^\star)}{\|\nabla f(x)\|^2}\\
&\leq \frac{1}{2\mu}\frac{\|\nabla f(x-P\nabla f(x))\|^2}{\|\nabla f(x)\|^2} && \text{(B.2)}\\
&\leq \frac{1}{2\mu}\frac{[\|\nabla f(x)\| + \|P\|\cdot\|\nabla f(x)\|]^2}{\|\nabla f(x)\|^2} && \text{(B.3)}\\
&\leq \frac{1}{2\mu}(LD+1)^2, && \text{(B.4)}
\end{aligned}
$$

where (B.1) applies $h_x(P) \geq \frac{f(x^\star)-f(x)}{\|\nabla f(x)\|^2}$; (B.2) uses $f(x) - f(x^\star) \leq \frac{1}{2\mu}\|\nabla f(x)\|^2$; (B.3) uses L-smoothness and (B.4) uses $\|P\| \leq D$. □

Then we show that `HDM` converges even when η is a constant that does not depend on K.

Lemma B.2. *Under* **A1** *to* **A4**, *Algorithm 7 with* $\eta_k \equiv \eta \in (0, \frac{1}{2L(LD+1)^2\kappa}]$ *satisfies*

- $\lim_{k\to\infty} \|x^k - x^\star\| = 0$.
- $\lim_{K\to\infty} \sum_{k=1}^{K} \|x^k - x^\star\|^2 < \infty$.

Proof. Using the online gradient descent update, we have

$$\begin{aligned}
\|P_{k+1} - P\|_F^2 &\le \|P_k - \eta \nabla h_{x^k}(P_k) - P\|_F^2 \\
&= \|P_k - P\|_F^2 - 2\eta \langle \nabla h_{x^k}(P_k), P_k - P\rangle + \eta^2 \|\nabla h_{x^k}(P_k)\|_F^2 \\
&\le \|P_k - P\|_F^2 - 2\eta [h_{x^k}(P_k) - h_{x^k}(P)] + 2L\eta^2 \left[h_{x^k}(P_k) - \inf_{P \in \mathbb{R}^{n\times n}} h_{x^k}(P) \right] \qquad \text{(B.5)} \\
&= \|P_k - P\|_F^2 - 2\eta h_{x^k}(P_k) + 2\eta h_{x^k}(P) + 2L\eta^2 \left[h_{x^k}(P_k) - \inf_{P \in \mathbb{R}^{n\times n}} h_{x^k}(P) \right],
\end{aligned}$$

where (B.5) follows from convexity $h_{x^k}(P) \ge h_{x^k}(P_k) + \langle \nabla h_{x^k}(P_k), P - P_k\rangle$ and L-smoothness of $h_x(P)$. Next, we invoke the upperbound on $h_{x^k}(P_k) - \inf_{Q\in\mathbb{R}^{n\times n}} h_{x^k}(Q)$ from Lemma B.1:

$$2L\eta^2 \left[h_{x^k}(P_k) - \inf_{P \in \mathbb{R}^{n\times n}} h_{x^k}(P) \right] \le \frac{2L}{2\mu}(LD+1)^2\eta^2 = \kappa(LD+1)^2\eta^2.$$

and deduce that

$$\begin{aligned}
2\eta h_{x^k}(P_k) &\le 2\eta h_{x^k}(P) + \|P_k - P\|_F^2 - \|P_{k+1} - P\|_F^2 + 2L\eta^2 \left[h_{x^k}(P_k) - \inf_{P \in \mathbb{R}^{n\times n}} h_{x^k}(P) \right] \\
&\le 2\eta h_{x^k}(P) + \|P_k - P\|_F^2 - \|P_{k+1} - P\|_F^2 + \eta^2 \kappa (LD+1)^2.
\end{aligned}$$

Next, we divide both sides of the inequality by 2η and

$$h_{x^k}(P_k) \le h_{x^k}(P) + \frac{\|P_k - P\|_F^2 - \|P_{k+1} - P\|_F^2}{2\eta} + \frac{\eta\kappa(LD+1)^2}{2}.$$

Telescoping the relation and using $\mathrm{diam}(\mathcal{P}) \le D$, we get

$$\sum_{k=1}^K h_{x^k}(P_k) \le \sum_{k=1}^K h_{x^k}(P) + \frac{D^2}{2\eta} + \frac{\eta\kappa(LD+1)^2}{2} K$$

Taking $P = (1/L)I$ and taking average, $\sum_{k=1}^K h_{x^k}(P) \le -\frac{1}{2L}K$ and

$$\frac{1}{K}\sum_{k=1}^K h_{x^k}(P_k) \le -\frac{1}{2L} + \frac{D^2}{2\eta K} + \frac{\eta\kappa(LD+1)^2}{2} = -\frac{1}{4L} + \frac{D^2}{2\eta K} + \frac{\eta\kappa(LD+1)^2}{2} - \frac{1}{4L}$$

With $\eta \le \frac{1}{2L(LD+1)^2\kappa}$, we have $\frac{\eta\kappa(LD+1)^2}{2} - \frac{1}{4L} \le 0$ and

$$\frac{1}{K}\sum_{k=1}^K h_{x^k}(P_k) \le -\frac{1}{4L} + \frac{D^2 L(LD+1)^2\kappa}{K}.$$

Using the reduction Lemma 3.5, we get, for any $k \ge 1$ (since η does not depend on the iteration

number),

$$f(x^{k+1}) - f(x^\star) \leq [f(x^1) - f(x^\star)] \left(1 - 2\mu \max\left\{\frac{1}{4L} - \frac{D^2 L(LD+1)^2\kappa}{k}, 0\right\}\right)^k$$

and there exists some K_0 such that for all $k \geq K_0$, that $[f(x^k) - f(x^\star)](1 - \frac{1}{4\kappa})^k \leq [f(x^1) - f(x^\star)]$ since

$$\lim_{k\to\infty} 1 - 2\mu \max\left\{\frac{1}{4L} - \frac{2D^2 L(LD+1)^2\kappa}{k}, 0\right\} = 1 - \frac{1}{2\kappa} < 1 - \frac{1}{4\kappa}.$$

This proves the first relation $\lim_{k\to\infty} \|x^k - x^\star\| = 0$ since $\|x - x^\star\|^2 \leq \frac{2}{\mu}[f(x) - f(x^\star)]$ and the second relation follows directly from

$$\sum_{k=1}^{\infty} \|x^k - x^\star\|^2 = \sum_{k=1}^{K_0} \|x^k - x^\star\|^2 + \sum_{k=K_0+1}^{\infty} \|x^k - x^\star\|^2 \tag{B.6}$$

$$\leq \sum_{k=1}^{K_0} \|x^k - x^\star\|^2 + \sum_{k=K_0+1}^{\infty} \frac{2}{\mu}[f(x^1) - f(x^\star)] \left(1 - \frac{1}{4\kappa}\right)^{-k} < \infty. \tag{B.7}$$

□

Now we are ready to prove Theorem 3.12, and we start by stating the precise definition of a uniformly independent sequence.

Definition B.3 (Uniformly linearly indepdendent sequence Conn et al. [1991])**.** A sequence of unit-norm vectors $\{g^k\}, g^k \in \mathbb{R}^n, \|g^k\| = 1$ is uniformly linearly independent if there exists a constant $c > 0, K_0 \geq 0$ and $m \geq n$ such that for each $k \geq K_0$, one can choose n distinct indices

$$k \leq k_1 < \cdots < k_n \leq k + m$$

with $\sigma_{\min}([g^{k_1}, \ldots, g^{k_n}]) \geq c$.

Proof of Theorem 3.12. We prove by contradiction. For brevity we denote $g^k := \frac{\nabla f(x^k)}{\|\nabla f(x^k)\|}$ and $e_k := \|P_k - P^\star\|_F^2$. Recall that $P^\star = [\nabla^2 f(x^\star)]^{-1}$. First, using Lemma B.2, for any $\varepsilon > 0$, there exists some index K_1 such that for all $k \geq K_1$ we have $\|x^k - x^\star\|^2 \leq \varepsilon$ and that $\sum_{k=1}^{\infty} \|x^k - x^\star\|^2$ is bounded. Then we show that $\lim_{k\to\infty} \|\nabla h_{x^k}(P_k)\|_F = 0$ using (3.33): after re-arrangement, for any

$K \geq 1$,

$$\begin{aligned}
\sum_{k=1}^{K} \hat{h}_{x^k}(P_k) &\leq \frac{2\eta}{2\eta(1-\eta L)} \sum_{k=1}^{K} \hat{h}_{x^k}(P^\star) + \frac{1}{2\eta(1-\eta L)} \|P_1 - P^\star\|_F^2 \\
&\leq \frac{2\eta}{2\eta(1-\eta L)} \frac{H^2\kappa}{8\mu^3} \sum_{k=1}^{K} \|x^k - x^\star\|^2 + \frac{1}{2\eta(1-\eta L)} \|P_1 - P^\star\|_F^2. \qquad \text{(B.8)} \\
&\leq \frac{2\eta}{2\eta(1-\eta L)} \frac{H^2\kappa}{8\mu^3} \sum_{k=1}^{\infty} \|x^k - x^\star\|^2 + \frac{1}{2\eta(1-\eta L)} \|P_1 - P^\star\|_F^2,
\end{aligned}$$

where (B.8) applies (3.28). Since $\sum_{k=1}^{\infty} \|x^k - x^\star\|^2$ is bounded and $\hat{h}_x(P)$ is nonnegative, we must have $\lim_{k\to\infty} \hat{h}_{x^k}(P_k) = 0$. Further notice that $\|\nabla \hat{h}_{x^k}(P_k)\|_F^2 \leq 2L\hat{h}_{x^k}(P_k)$, it implies

$$\lim_{k\to\infty} \sum_{k=1}^{K} \|\nabla h_{x^k}(P_k)\|_F^2 < \infty,$$

giving $\lim_{k\to\infty} \|\nabla h_{x^k}(P_k)\|_F = 0$ and $\lim_{k\to\infty} P_k = \bar{P}$ also exists. Now suppose by contradiction that $\|\bar{P} - P^\star\|_F = \theta > 0$. Then there exists some $K_2 > 0$ such that for all $k \geq K_2$, $\|P_k - \bar{P}\|_F \leq \varepsilon$. For $k \geq \max\{K_0, K_1, K_2\} + 1$, we invoke Lemma 3.11 with $\eta \in (0, \frac{1}{2L}]$ to get

$$\begin{aligned}
\|P_{k+1} - P^\star\|_F^2 &\leq \|P_k - P^\star\|_F^2 - \alpha_1 \|(P_k - P^\star)g^k\|^2 + \alpha_2\varepsilon \\
&= \|P_k - P^\star\|_F^2 - \alpha_1 \|(P_k - \bar{P} + \bar{P} - P^\star)g^k\|^2 + \alpha_2\varepsilon \\
&\leq \|P_k - P^\star\|_F^2 - \frac{\alpha_1}{2} \|(\bar{P} - P^\star)g^k\|^2 + 3\alpha_1 \|(P_k - \bar{P})g^k\|^2 + \alpha_2\varepsilon \\
&\leq \|P_k - P^\star\|_F^2 - \frac{\alpha_1}{2} \|(\bar{P} - P^\star)g^k\|^2 + 3\alpha_1\varepsilon^2 + \alpha_2\varepsilon \qquad \text{(B.9)} \\
&= \|P_k - P^\star\|_F^2 - \frac{\alpha_1}{2} \left\langle g^k(g^k)^\top, (\bar{P} - P^\star)^\top(\bar{P} - P^\star) \right\rangle + 3\alpha_1\varepsilon^2 + \alpha_2\varepsilon, \qquad \text{(B.10)}
\end{aligned}$$

where $\alpha_1 = \frac{\mu(\eta - L\eta^2)}{2} > 0$, $\alpha_2 = \frac{1}{4}(2\eta - L\eta^2)H^2\kappa\mu^{-3}$, and (B.9) uses the fact that $\|P_k - P^\star\|_F \leq \varepsilon$. Telescoping (B.10) for the next $m+1$ iterations, we deduce that

$$\begin{aligned}
e_{k+m+1} &= \|P_{k+m+1} - P^\star\|_F^2 \\
&\leq \|P_k - P^\star\|_F^2 - \frac{\alpha_1}{2} \sum_{j=0}^{m} \langle g^{k+j}(g^{k+j})^\top, (\bar{P} - P^\star)^\top(\bar{P} - P^\star) \rangle + (3\alpha_1\varepsilon^2 + \alpha_2\varepsilon)(m+1) \\
&= e_k - \frac{\alpha_1}{2} \left\langle \sum_{j=0}^{m} g^{k+j}(g^{k+j})^\top, (\bar{P} - P^\star)^\top(\bar{P} - P^\star) \right\rangle + (3\alpha_1\varepsilon^2 + \alpha_2\varepsilon)(m+1)
\end{aligned}$$

and using the independent sequence assumption, we can pick $k_1, \ldots, k_n$ such that

$$\sigma_{\min}([g^{k_1}, \ldots, g^{k_n}]) \geq c$$

and $\sum_{j=0}^{m} g^{k+j}(g^{k+j})^\top \succeq \sum_{i=1}^{n} g^{k_i}(g^{k_i})^\top \succeq c^2 I$. Hence

$$\left\langle \sum_{j=0}^{m} g^{k+j}(g^{k+j})^\top, (\bar{P}-P^\star)^\top(\bar{P}-P^\star) \right\rangle \geq c^2 \operatorname{tr}[(\bar{P}-P^\star)^\top(\bar{P}-P^\star)] = c^2\|\bar{P}-P^\star\|_F^2 = c^2\theta^2$$

and $e_{k+m+1} \leq e_k - \frac{\alpha_1 c^2\theta^2}{2} + (3\alpha_1\varepsilon^2 + \alpha_2\varepsilon)(m+1)$. Since ε is arbitrary, we can repeat the argument till $e_{k+m+1} < 0$, which leads to contradiction unless $\theta = 0$. This completes the proof.

□

Appendix C

Supplement for Chapter 4

C.1 Neural Control Policies Experiment Details

The unicycle system follows the linear dynamics $\dot{x} = F(x) + Gu$, where

$$F(x) = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ w \\ 0 \\ 0 \end{bmatrix} \quad \text{and} \quad G(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

For an elliptical obstacle centered at (c_x, c_y) with axes (r_x, r_y), we use the higher order CBF $h(x) = \dot{h}_e(x) + \kappa h_e(x)$ with $\kappa > 0$, Min and Azizan [2024] in our experiments, where

$$h_e(x) = \big(\frac{c_x - p_x + \ell\cos\theta}{r_x}\big)^2 + \big(\frac{c_y - p_y + \ell\sin\theta}{r_y}\big)^2 - 1.$$

C.2 Tables for All Test Results in Section 4.5.1

This section includes all the evaluation results on test instances for experiments in Section 4.5.1. For ease of navigation, Table C.1 summarizes the correspondence between figures presented in the main text and their associated results tables organized by problem class. Columns "# of Ineq. Violations" and "# of Eq. Violations" are counted with threshold 10^{-4}.

Tables	Problem Class	Figures
Table C.2	NCP	Figure 4.4 & Figure 4.5
Table C.3	QCQP	Figure 4.4 & Figure 4.5
Table C.4	NCP	Figure 4.6
Table C.5	QCQP (10 inequality constraints)	Figure 4.7
Table C.6	QCQP (50 inequality constraints)	Figure 4.7
Table C.7	QCQP (100 inequality constraints)	Figure 4.7

Table C.1: Correspondence between tables, problem classes, and figures in main paper.

Table C.2: Evaluation metrics on the NCP test set. Values shown as mean ± std across 5 random seeds.

Method	Max Opt. Gap	GMean Opt. Gap	GMean Ineq. Error	# Ineq Violations	Max Eq. Error	GMean Eq. Error	# Eq Violations	Test Time (s)
DC3	5.31 ± 4.05	1.86 ± 1.21	$(1.00 \pm 0.00) \times 10^{-16}$	$(7.20 \pm 10.74) \times 10^{-4}$	$(6.76 \pm 4.32) \times 10^{-14}$	$(5.29 \pm 1.54) \times 10^{-15}$	0.00	0.14 ± 0.10
HardNet	$(1.02 \pm 0.47) \times 10^{1}$	$(9.17 \pm 0.99) \times 10^{-1}$	$(1.10 \pm 0.01) \times 10^{-16}$	0.00	$(1.13 \pm 0.14) \times 10^{-13}$	$(1.04 \pm 0.03) \times 10^{-14}$	0.00	0.05 ± 0.01
HProj	$(5.97 \pm 12.04) \times 10^{2}$	3.72 ± 7.12	0.00	0.00	$(2.24 \pm 4.49) \times 10^{-12}$	$(3.12 \pm 5.58) \times 10^{-14}$	0.00	0.05 ± 0.03
SnareNet (tol = 10^{-12})	$(4.26 \pm 0.44) \times 10^{-1}$	$(7.91 \pm 1.00) \times 10^{-2}$	$(1.35 \pm 0.11) \times 10^{-16}$	0.00	$(7.34 \pm 0.83) \times 10^{-13}$	$(5.27 \pm 2.07) \times 10^{-15}$	0.00	0.80 ± 0.14
SnareNet (tol = 10^{-4})	$(4.26 \pm 0.42) \times 10^{-1}$	$(7.90 \pm 1.00) \times 10^{-2}$	$(5.40 \pm 1.53) \times 10^{-16}$	0.00	$(8.07 \pm 1.00) \times 10^{-5}$	$(1.81 \pm 0.64) \times 10^{-6}$	0.00	0.33 ± 0.05
SnareNet (tol = 10^{-6})	$(4.28 \pm 0.45) \times 10^{-1}$	$(7.90 \pm 1.01) \times 10^{-2}$	$(3.82 \pm 0.88) \times 10^{-16}$	0.00	$(8.24 \pm 0.64) \times 10^{-7}$	$(1.20 \pm 0.44) \times 10^{-8}$	0.00	0.46 ± 0.17
SnareNet (tol = 10^{-8})	$(4.28 \pm 0.46) \times 10^{-1}$	$(7.90 \pm 1.00) \times 10^{-2}$	$(2.71 \pm 0.51) \times 10^{-16}$	0.00	$(7.88 \pm 1.14) \times 10^{-9}$	$(7.74 \pm 3.91) \times 10^{-11}$	0.00	0.64 ± 0.26
SnareNet (tol = 10^{-10})	$(4.26 \pm 0.45) \times 10^{-1}$	$(7.91 \pm 1.00) \times 10^{-2}$	$(1.93 \pm 0.26) \times 10^{-16}$	0.00	$(8.42 \pm 1.36) \times 10^{-11}$	$(5.23 \pm 2.86) \times 10^{-13}$	0.00	0.88 ± 0.18

Table C.3: Evaluation metrics on the QCQP test set. Values shown as mean ± std across 5 random seeds.

Method	Max Opt. Gap	GMean Opt. Gap	GMean Ineq. Error	# Ineq Violations	Max Eq. Error	GMean Eq. Error	# Eq Violations	Test Time (s)
DC3	$(1.52 \pm 1.00) \times 10^{1}$	9.22 ± 7.30	$(1.17 \pm 0.32) \times 10^{-16}$	0.19 ± 0.33	$(3.00 \pm 1.45) \times 10^{-14}$	$(3.28 \pm 0.98) \times 10^{-15}$	0.00	0.15 ± 0.06
HProj	$(7.78 \pm 17.18) \times 10^{1}$	$(3.41 \pm 7.57) \times 10^{1}$	$(1.10 \pm 2.47) \times 10^{-13}$	10.00 ± 22.36	$(2.01 \pm 4.41) \times 10^{-12}$	$(2.77 \pm 6.08) \times 10^{-13}$	0.00	0.26 ± 0.37
SnareNet (tol = 10^{-12})	$(1.24 \pm 0.06) \times 10^{-1}$	$(2.60 \pm 0.26) \times 10^{-2}$	$(1.36 \pm 0.17) \times 10^{-16}$	0.00	$(7.06 \pm 0.95) \times 10^{-13}$	$(9.93 \pm 3.87) \times 10^{-16}$	0.00	1.37 ± 0.48
SnareNet (tol = 10^{-4})	$(1.24 \pm 0.06) \times 10^{-1}$	$(2.60 \pm 0.26) \times 10^{-2}$	$(7.74 \pm 1.97) \times 10^{-15}$	0.00	$(8.33 \pm 0.87) \times 10^{-5}$	$(1.93 \pm 0.45) \times 10^{-6}$	0.00	0.69 ± 0.74
SnareNet (tol = 10^{-6})	$(1.25 \pm 0.06) \times 10^{-1}$	$(2.60 \pm 0.27) \times 10^{-2}$	$(2.68 \pm 0.64) \times 10^{-15}$	0.00	$(7.64 \pm 0.97) \times 10^{-7}$	$(5.92 \pm 2.64) \times 10^{-9}$	0.00	0.79 ± 0.28
SnareNet (tol = 10^{-8})	$(1.24 \pm 0.07) \times 10^{-1}$	$(2.60 \pm 0.26) \times 10^{-2}$	$(9.52 \pm 1.98) \times 10^{-16}$	0.00	$(7.88 \pm 0.82) \times 10^{-9}$	$(2.13 \pm 1.23) \times 10^{-11}$	0.00	1.18 ± 0.17
SnareNet (tol = 10^{-10})	$(1.25 \pm 0.07) \times 10^{-1}$	$(2.60 \pm 0.26) \times 10^{-2}$	$(3.35 \pm 0.70) \times 10^{-16}$	0.00	$(7.31 \pm 0.72) \times 10^{-11}$	$(7.35 \pm 5.48) \times 10^{-14}$	0.00	1.57 ± 0.46

Table C.4: Evaluation metrics on the NCP test set. Values shown as mean ± std across 5 random seeds.

Method	Obj. Value	Max Opt. Gap	GMean Opt. Gap	Max Ineq. Error	GMean Ineq. Error	# Ineq Violations	Max Eq. Error	GMean Eq. Error	# Eq Violations	Test Time (s)
SnareNet ($\lambda = 0.001$)	-11.19 ± 0.29	2.92 ± 1.73	$(4.28 \pm 2.66) \times 10^{-1}$	$(4.07 \pm 1.46) \times 10^{-9}$	$(1.72 \pm 0.14) \times 10^{-16}$	0.00	$(6.57 \pm 2.36) \times 10^{-9}$	$(1.87 \pm 0.97) \times 10^{-10}$	0.00	0.28 ± 0.12
SnareNet ($\lambda = 0.01$)	-11.59 ± 0.01	$(4.28 \pm 0.46) \times 10^{-1}$	$(7.90 \pm 1.00) \times 10^{-2}$	$(5.35 \pm 0.88) \times 10^{-9}$	$(2.71 \pm 0.51) \times 10^{-16}$	0.00	$(7.88 \pm 1.14) \times 10^{-9}$	$(7.74 \pm 3.91) \times 10^{-11}$	0.00	0.76 ± 0.74
SnareNet ($\lambda = 0.1$)	-11.64 ± 0.00	$(2.18 \pm 0.12) \times 10^{-1}$	$(3.86 \pm 0.22) \times 10^{-2}$	$(1.40 \pm 0.22) \times 10^{-8}$	$(3.58 \pm 0.04) \times 10^{-16}$	0.00	$(2.08 \pm 0.33) \times 10^{-8}$	$(4.29 \pm 0.82) \times 10^{-10}$	0.00	0.61 ± 0.26
SnareNet ($\lambda = 10$)	-11.67 ± 0.00	$(1.87 \pm 0.16) \times 10^{-2}$	$(4.29 \pm 0.06) \times 10^{-3}$	$(8.65 \pm 0.46) \times 10^{-8}$	$(6.38 \pm 0.16) \times 10^{-16}$	0.00	$(1.20 \pm 0.06) \times 10^{-7}$	$(4.98 \pm 0.61) \times 10^{-9}$	0.00	2.19 ± 1.24
SnareNet ($\lambda = 1$)	-11.66 ± 0.00	$(6.99 \pm 0.74) \times 10^{-2}$	$(1.57 \pm 0.05) \times 10^{-2}$	$(1.96 \pm 0.08) \times 10^{-8}$	$(5.01 \pm 0.09) \times 10^{-16}$	0.00	$(3.21 \pm 0.13) \times 10^{-8}$	$(6.27 \pm 0.93) \times 10^{-10}$	0.00	1.31 ± 0.32
SnareNet ($\lambda = 5$)	-11.67 ± 0.00	$(2.74 \pm 0.08) \times 10^{-2}$	$(6.34 \pm 0.06) \times 10^{-3}$	$(5.07 \pm 0.25) \times 10^{-8}$	$(6.78 \pm 0.12) \times 10^{-16}$	0.00	$(7.52 \pm 0.39) \times 10^{-8}$	$(3.19 \pm 0.24) \times 10^{-9}$	0.00	2.22 ± 1.17

Table C.5: Evaluation metrics on the QCQP with 10 inequality constraints test set. Values shown as mean ± std across 5 random seeds.

Method	Max Opt. Gap	GMean Opt. Gap	GMean Ineq. Error	# Ineq Violations	Max Eq. Error	GMean Eq. Error	# Eq Violations	Test Time (s)
Optimizer (Gurobi)	1.42×10^{-14}	4.48×10^{-16}	0.00	0.00	5.43×10^{-10}	1.06×10^{-14}	0.00	428.44
DC3	$(7.96 \pm 11.33) \times 10^{1}$	$(2.21 \pm 1.94) \times 10^{1}$	$(3.52 \pm 7.87) \times 10^{-10}$	0.97 ± 1.88	$(1.48 \pm 1.11) \times 10^{-13}$	$(1.40 \pm 0.94) \times 10^{-14}$	0.00	0.08 ± 0.07
HProj	$(2.40 \pm 1.99) \times 10^{-1}$	$(1.60 \pm 1.42) \times 10^{-1}$	0.00	0.00	$(1.08 \pm 0.92) \times 10^{-13}$	$(1.41 \pm 0.98) \times 10^{-14}$	0.00	0.02 ± 0.02
SnareNet (Ours)	$(1.74 \pm 0.08) \times 10^{-1}$	$(3.16 \pm 0.29) \times 10^{-2}$	$(4.78 \pm 0.92) \times 10^{-15}$	0.00	$(5.13 \pm 1.28) \times 10^{-5}$	$(6.35 \pm 1.68) \times 10^{-7}$	0.00	0.25 ± 0.01

Table C.6: Evaluation metrics on the QCQP with 50 inequality constraints test set. Values shown as mean ± std across 5 random seeds.

Method	Max Opt. Gap	GMean Opt. Gap	GMean Ineq. Error	# Ineq Violations	Max Eq. Error	GMean Eq. Error	# Eq Violations	Test Time (s)
Optimizer (Gurobi)	7.11×10^{-15}	2.74×10^{-16}	1.01×10^{-16}	0.00	1.58×10^{-8}	3.89×10^{-13}	0.00	1742.79
DC3	$(3.37 \pm 1.78) \times 10^{1}$	$(1.59 \pm 0.76) \times 10^{1}$	$(2.39 \pm 4.72) \times 10^{-13}$	5.33 ± 5.07	$(7.63 \pm 5.12) \times 10^{-14}$	$(5.42 \pm 3.01) \times 10^{-15}$	0.00	0.18 ± 0.08
HProj	$(7.78 \pm 17.18) \times 10^{1}$	$(3.41 \pm 7.57) \times 10^{1}$	$(1.10 \pm 2.47) \times 10^{-13}$	10.00 ± 22.36	$(2.01 \pm 4.41) \times 10^{-12}$	$(2.77 \pm 6.08) \times 10^{-13}$	0.00	0.26 ± 0.37
SnareNet (Ours)	$(1.25 \pm 0.06) \times 10^{-1}$	$(2.60 \pm 0.27) \times 10^{-2}$	$(2.68 \pm 0.64) \times 10^{-15}$	0.00	$(7.64 \pm 0.97) \times 10^{-7}$	$(5.92 \pm 2.64) \times 10^{-9}$	0.00	0.39 ± 0.01

Table C.7: Evaluation metrics on the QCQP with 100 inequality constraints test set. Values shown as mean ± std across 5 random seeds.

Method	Max Opt. Gap	GMean Opt. Gap	GMean Ineq. Error	# Ineq Violations	Max Eq. Error	GMean Eq. Error	# Eq Violations	Test Time (s)
Optimizer (Gurobi)	7.11×10^{-15}	2.57×10^{-16}	1.04×10^{-16}	0.00	1.57×10^{-7}	7.90×10^{-13}	0.00	3563.30
DC3	$(1.14 \pm 0.66) \times 10^{1}$	6.86 ± 4.63	$(1.42 \pm 0.91) \times 10^{-16}$	0.65 ± 1.37	$(1.76 \pm 1.27) \times 10^{-14}$	$(1.70 \pm 0.89) \times 10^{-15}$	0.00	0.48 ± 0.08
HProj	$(2.35 \pm 5.22) \times 10^{2}$	$(1.53 \pm 3.40) \times 10^{2}$	$(5.40 \pm 12.08) \times 10^{-11}$	20.07 ± 44.86	$(1.83 \pm 4.03) \times 10^{-12}$	$(3.04 \pm 6.73) \times 10^{-13}$	0.00	0.44 ± 0.69
SnareNet (Ours)	$(1.74 \pm 0.08) \times 10^{-1}$	$(3.16 \pm 0.29) \times 10^{-2}$	$(1.88 \pm 0.22) \times 10^{-15}$	0.00	$(7.77 \pm 0.54) \times 10^{-7}$	$(2.80 \pm 0.97) \times 10^{-9}$	0.00	0.68 ± 0.12

C.3 Scaling of Computational Resources

Table C.8 shows the training time, memory usage, and number of repair iterations taken for SnareNet on non-convex programs (NCPs) of varying sizes. The experiments were run on an Intel 6426Y CPU and an NVIDIA L40S GPU, constrained to 16, 16, and 32 GB of GPU memory per run for 100, 500, and 1000 constraints, respectively. Note that SnareNet may use less computational resources for larger problems under a fixed number of constraints since the problems can be less constrained and require fewer repair iterations to achieve feasibility. For example, the 500 and 1000 variable problems with 100 constraints are cheaper (both in time and memory) than the 100 variable problem with 100 constraints. Table C.8 indicates that SnareNet is more suitable for problems with $m \ll n$.

Table C.8: Total training time (sec) / memory usage / maximum repair iterations on NCPs of varying sizes.

	100 constraints (50 eq. + 50 ineq.)	500 constraints (50 eq. + 450 ineq.)	1000 constraints (50 eq. + 950 ineq.)
100 var	3674.4s / 2.3GB / 32	3496.9s / 4.4GB / 100	5832.5s / 4.9GB / 100
500 var	869.1s / 1.3GB / 3	6493.8s / 11.6GB / 11	29301.7s / 23.8GB / 30
1000 var	1107.3s / 1.3GB / 3	2724.5s / 3.3GB / 3	>86400.0s / 31.1GB / 8

C.4 Soft Constraint Training Can Be Counterproductive

Soft constraint training is commonly employed as a warm-up strategy for hard-constrained neural networks. However, as shown in Figure C.1, this approach can be counterproductive. While `HardNet` achieves a lower optimality gap after 1000 soft epochs than after 500, this reduction does not persist: once hard constraint training begins, the optimality gap increases sharply, negating any apparent progress. Additional soft constraint epochs constitute an inefficient use of computational resources, as the gains from soft constraint training do not transfer to improved final model performance.

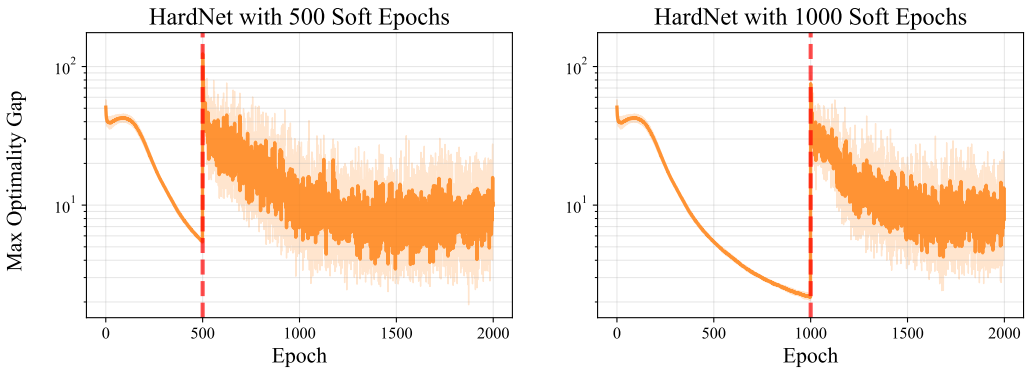


Figure C.1: Maximum optimality gap over all NCP validation instances for `HardNet` trained with soft constraints for 500 and 1000 epochs and hard constraints for the rest.

Bibliography

Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Disentangling adaptive gradient methods from learning rates. *arXiv preprint arXiv:2002.11803*, 2020.

Levent Aksoy, Paulo Flores, and José Monteiro. Exact and approximate algorithms for the filter design optimization problem. *IEEE Transactions on Signal Processing*, 63(1):142–154, 2014.

Luís B Almeida, Thibault Langlois, José D Amaral, and Alexander Plakhov. Parameter adaptation in stochastic optimization. In *On-line learning in neural networks*, pages 111–134. 1999.

Anna Altman and Jacek Gondzio. Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization. *Optimization Methods and Software*, 11(1-4):275–302, 1999. doi: 10.1080/10556789908805754.

Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8): 3861–3876, 2016.

Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications, 2019. URL `https://arxiv.org/abs/1903.11199`.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pages 136–145. PMLR, 2017.

Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular systems biology*, 12(7):878, 2016.

Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BkrsAzWAb`.

Stefania Bellavia, Jacek Gondzio, and Benedetta Morini. A Matrix-Free Preconditioner for Sparse Symmetric Positive Definite Systems and Least-Squares Problems. *SIAM Journal on Scientific Computing*, 35(1):A192–A211, January 2013. ISSN 1064-8275. doi: 10.1137/110840819.

Luca Bergamaschi, Jacek Gondzio, Ángeles Martínez, John W. Pearson, and Spyridon Pougkakiotis. A new preconditioning approach for an interior point-proximal method of multipliers for linear and convex quadratic programming. *Numerical Linear Algebra with Applications*, 28(4):e2361, 2021. ISSN 1099-1506. doi: 10.1002/nla.2361.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1):65–98, February 2017. doi: 10.1137/141000671.

Silvana Bocanegra, Frederico F Campos, and Aurelio Ribeiro Leite Oliveira. Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods. *Computational Optimization and Applications*, 36(2-3):149–164, February 2007. doi: 10.1007/s10589-006-9009-5.

Boris Bonev, Thorsten Kurth, Ankur Mahesh, Mauro Bisson, Jean Kossaifi, Karthik Kashinath, Anima Anandkumar, William D Collins, Michael S Pritchard, and Alexander Keller. Fourcastnet 3: A geometric approach to probabilistic machine-learning weather forecasting at scale. *arXiv preprint arXiv:2507.12144*, 2025.

Luciana Casacio, Christiano Lyra, Aurelio Ribeiro Leite Oliveira, and Cecilia Orellana Castro. Improving the preconditioning of linear systems from interior point methods. *Computers & Operations Research*, 85:129–138, September 2017. ISSN 03050548. doi: 10.1016/j.cor.2017.04.005.

Tony F. Chan. An Optimal Circulant Preconditioner for Toeplitz Systems. *SIAM Journal on Scientific and Statistical Computing*, 9(4):766–771, July 1988. ISSN 0196-5204. doi: 10.1137/0909051.

Kartik Chandra, Audrey Xie, Jonathan Ragan-Kelley, and Erik Meijer. Gradient descent: The ultimate optimizer. *Advances in Neural Information Processing Systems*, 35:8214–8225, 2022.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, April 2011. ISSN 2157-6904, 2157-6912. doi: 10.1145/1961189.1961199.

Siyuan Chen, Minghao Guo, Caoliwen Wang, Anka He Chen, Yikun Zhang, Jingjing Chai, Yin Yang, Wojciech Matusik, and Peter Yichen Chen. Physically valid biomolecular interaction modeling with gauss-seidel projection. *arXiv preprint arXiv:2510.08946*, 2025.

Agniva Chowdhury, Gregory Dexter, Palma London, Haim Avron, and Petros Drineas. Faster randomized interior point methods for Tall/Wide linear programs. *Journal of Machine Learning Research*, 23(336):1–48, 2022. URL `http://jmlr.org/papers/v23/21-0709.html`.

Raimondas Čiegis, David Henty, Bo Kågström, Julius Žilinskas, Kristian Woodsend, and Jacek Gondzio. High-performance parallel support vector machine training. *Parallel Scientific Computing and Optimization: Advances and Applications*, pages 83–92, 2009.

Andrew R Conn, Nicholas IM Gould, and Ph L Toint. Convergence of quasi-newton matrices generated by the symmetric rank one update. *Mathematical programming*, 50(1):177–195, 1991.

Marina Danilova, Anastasiia Kulakova, and Boris Polyak. Non-monotone behavior of the heavy ball method. In *Difference Equations and Discrete Dynamical Systems with Applications: 24th ICDEA, Dresden, Germany, May 21–25, 2018 24*, pages 213–230. Springer, 2020.

Rudrajit Das, Naman Agarwal, Sujay Sanghavi, and Inderjit S Dhillon. Towards quantifying the preconditioning effect of adam. *arXiv preprint arXiv:2402.07114*, 2024.

F. de Prenter, C. V. Verhoosel, G. J. van Zwieten, and E. H. van Brummelen. Condition number analysis and preconditioning of the finite cell method. *Computer Methods in Applied Mechanics and Engineering*, 316:297–327, April 2017. ISSN 0045-7825. doi: 10.1016/j.cma.2016.07.006.

Aaron Defazio, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, Ashok Cutkosky, et al. The road less scheduled. *arXiv preprint arXiv:2405.15682*, 2024.

Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.

Alp Dener, Marco Andres Miller, Randy Michael Churchill, Todd Munson, and Choong-Seock Chang. Training neural networks under physical constraints using a stochastic augmented lagrangian approach, 2020. URL `https://arxiv.org/abs/2009.07330`.

Gregory Dexter, Agniva Chowdhury, Haim Avron, and Petros Drineas. On the convergence of inexact predictor-corrector methods for linear programming. In *International Conference on Machine Learning*, pages 5007–5038. PMLR, 2022. URL `https://proceedings.mlr.press/v162/dexter22a.html`.

Priya L Donti, David Rolnick, and J Zico Kolter. Dc3: A learning method for optimization with hard constraints. *arXiv preprint arXiv:2104.12225*, 2021.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

C. Durazzi and V. Ruggiero. Indefinitely preconditioned conjugate gradient method for large sparse equality and inequality constrained quadratic problems. *Numerical Linear Algebra with Applications*, 10(8):673–688, 2003. ISSN 1099-1506. doi: 10.1002/nla.308.

Shai Fine and Katya Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec):243–264, 2001. URL `https://www.jmlr.org/papers/v2/fine01a.html`.

Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

Anders Forsgren, Philip E. Gill, and Margaret H. Wright. Interior Methods for Nonlinear Optimization. *SIAM Review*, 44(4):525–597, January 2002. ISSN 0036-1445. doi: 10.1137/S0036144502414942.

Zachary Frangella, Pratik Rathore, Shipu Zhao, and Madeleine Udell. Sketchysgd: Reliable stochastic optimization via robust curvature estimates. *arXiv preprint arXiv:2211.08597*, 2022.

Zachary Frangella, Pratik Rathore, Shipu Zhao, and Madeleine Udell. Promise: Preconditioned stochastic optimization methods by incorporating scalable curvature estimates. *arXiv preprint arXiv:2309.02014*, 2023a.

Zachary Frangella, Joel A. Tropp, and Madeleine Udell. Randomized Nyström Preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 44(2):718–752, June 2023b. ISSN 0895-4798, 1095-7162. doi: 10.1137/21M1466244.

M. P. Friedlander and D. Orban. A primal–dual regularized interior-point method for convex quadratic programs. *Mathematical Programming Computation*, 4(1):71–107, March 2012. ISSN 1867-2957. doi: 10.1007/s12532-012-0035-2.

Wenzhi Gao, Ya-Chi Chu, Yinyu Ye, and Madeleine Udell. Gradient methods with online scaling. *arXiv preprint arXiv:2411.01803*, 2024.

Philip E. Gill, Walter Murray, Michael A. Saunders, J. A. Tomlin, and Margaret H. Wright. On projected newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. *Mathematical Programming*, 36(2):183–209, June 1986. ISSN 1436-4646. doi: 10.1007/BF02592025.

D. Goldfarb and S. Liu. An $\mathcal{O}(n^3\mathrm{L})$ primal interior point algorithm for convex quadratic programming. *Mathematical Programming*, 49(1):325–340, November 1990. ISSN 1436-4646. doi: 10.1007/BF01588795.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, 4 edition, 2013. ISBN 978-1-4214-0794-4.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel,

Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Jacek Gondzio. Matrix-free interior point method. *Computational Optimization and Applications*, 51(2):457–480, 2012a. doi: 10.1007/s10589-010-9361-3.

Jacek Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, May 2012b. ISSN 0377-2217. doi: 10.1016/j.ejor.2011.09.017.

Jacek Gondzio and Andreas Grothey. Exploiting structure in parallel implementation of interior point methods for optimization. *Computational Management Science*, 6(2):135–160, May 2009. ISSN 1619-6988. doi: 10.1007/s10287-008-0090-3.

Jacek Gondzio, Spyridon Pougkakiotis, and John W. Pearson. General-purpose preconditioning for regularized interior point methods. *Computational Optimization and Applications*, 83(3):727–757, December 2022. ISSN 0926-6003, 1573-2894. doi: 10.1007/s10589-022-00424-5.

Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of field robotics*, 37(3):362–386, 2020.

Panagiotis D Grontas, Antonio Terpin, Efe C Balta, Raffaello D'Andrea, and John Lygeros. Pinet: Optimizing hard-constrained neural networks with orthogonal projection layers. *arXiv preprint arXiv:2508.10480*, 2025.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL `https://www.gurobi.com`.

Oscar Gustafsson and Lars Wanhammar. Design of linear-phase fir filters combining subexpression sharing with milp. In *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.*, volume 3, pages III–III. IEEE, 2002.

Trevor Hastie. The elements of statistical learning: data mining, inference, and prediction, 2009.

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Magnus R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, November 1969. ISSN 1573-2878. doi: 10.1007/BF00927673.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

Ashfaq Iftakher, Rahul Golder, Bimol Nath Roy, and MM Faruque Hasan. Physics-informed neural networks with hard nonlinear equality and inequality constraints. *Computers & Chemical Engineering*, page 109418, 2025.

Ruichen Jiang, Qiujiang Jin, and Aryan Mokhtari. Online learning guided curvature approximation: A quasi-newton method with global non-asymptotic superlinear convergence. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 1962–1992. PMLR, 2023.

Qiujiang Jin, Ruichen Jiang, and Aryan Mokhtari. Non-asymptotic global convergence rates of bfgs with exact line search. *arXiv preprint arXiv:2404.01267*, 2024a.

Qiujiang Jin, Ruichen Jiang, and Aryan Mokhtari. Non-asymptotic global convergence analysis of bfgs with the armijo-wolfe line search. *arXiv preprint arXiv:2404.16731*, 2024b.

Olin G. Johnson, Charles A. Micchelli, and George Paul. Polynomial Preconditioners for Conjugate Gradient Calculations. *SIAM Journal on Numerical Analysis*, 20(2):362–376, April 1983. ISSN 0036-1429. doi: 10.1137/0720025.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

S Kapoor and P M Vaidya. Fast algorithms for convex quadratic programming and multicommodity flows. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing - STOC '86*, pages 147–159, Berkeley, 1986. ACM Press. ISBN 978-0-89791-193-1. doi: 10.1145/12130.12145.

Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4 (4):373–395, 1984. doi: 10.1007/BF02579150.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI Machine Learning Repository. URL `http://archive.ics.uci.edu`.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

Masakazu Kojima, Shinji Mizuno, and Akiko Yoshise. A Primal-Dual Interior Point Algorithm for Linear Programming. In Nimrod Megiddo, editor, *Progress in Mathematical Programming: Interior-Point and Related Methods*, pages 29–47. Springer, New York, NY, 1989. ISBN 978-1-4613-9617-8. doi: 10.1007/978-1-4613-9617-8_2.

Masakazu Kojima, Nimrod Megiddo, and Shinji Mizuno. A primal-dual infeasible-interior-point algorithm for linear programming. *Mathematical Programming*, 61(3):263–280, 1993. doi: 10.1007/BF01582151.

Frederik Kunstner, Victor Sanches Portella, Mark Schmidt, and Nicholas Harvey. Searching for optimal per-coordinate step-sizes with multidimensional backtracking. *Advances in Neural Information Processing Systems*, 36, 2024.

Jonathan Lacotte and Mert Pilanci. Effective dimension adaptive sketching methods for faster regularized least-squares optimization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 19377–19387, Red Hook, NY, USA, December 2020. Curran Associates Inc. ISBN 978-1-71382-954-6.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

Yuh-Jye Lee and Olvi L Mangasarian. Ssvm: A smooth support vector machine for classification. *Computational optimization and Applications*, 20:5–22, 2001.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Meiyi Li, Soheil Kolouri, and Javad Mohammadi. Learning to solve optimization problems with hard linear constraints. *IEEE Access*, 11:59995–60004, 2023.

Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE transactions on neural networks and learning systems*, 29(5):1454–1466, 2017.

Xiaoyu Li, Zhenxun Zhuang, and Francesco Orabona. A second look at exponential and cosine step sizes: Simplicity, adaptivity, and performance. In *International Conference on Machine Learning*, pages 6553–6564. PMLR, 2021.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024.

Enming Liang, Minghua Chen, and Steven H Low. Homeomorphic projection to ensure neural-network solution feasibility for constrained optimization. *Journal of Machine Learning Research*, 25(329):1–55, 2024.

Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1):193–228, November 1998. ISSN 0024-3795. doi: 10.1016/S0024-3795(98)10032-0.

Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.

Irvin J. Lustig, Roy E. Marsten, and David F. Shanno. On Implementing Mehrotra's Predictor–Corrector Interior-Point Method for Linear Programming. *SIAM Journal on Optimization*, 2(3): 435–449, August 1992. ISSN 1052-6234. doi: 10.1137/0802022.

Irvin J. Lustig, Roy E. Marsten, and David F. Shanno. Interior Point Methods for Linear Programming: Computational State of the Art. *ORSA Journal on Computing*, 6(1):1–14, February 1994. ISSN 0899-1499, 2326-3245. doi: 10.1287/ijoc.6.1.1.

Chris J Maddison, Daniel Paulin, Yee Whye Teh, and Arnaud Doucet. Dual space preconditioning for gradient descent. *SIAM Journal on Optimization*, 31(1):991–1016, 2021.

Yura Malitsky and Konstantin Mishchenko. Adaptive gradient descent without descent. In *International Conference on Machine Learning*, pages 6702–6712. PMLR, 2020.

Yura Malitsky and Konstantin Mishchenko. Adaptive proximal gradient method for convex optimization. *Advances in Neural Information Processing Systems*, 37:100670–100697, 2024.

Roy Marsten, Radhika Subramanian, Matthew Saltzman, Irvin Lustig, and David Shanno. Interior Point Methods for Linear Programming: Just Call Newton, Lagrange, and Fiacco and McCormick! *Interfaces*, 20(4):105–116, August 1990. ISSN 0092-2102. doi: 10.1287/inte.20.4.105.

Per-Gunnar Martinsson and Joel A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, May 2020. ISSN 0962-4929, 1474-0508. doi: 10.1017/S0962492920000021.

Nimrod Megiddo. Pathways to the Optimal Set in Linear Programming. In *Progress in Mathematical Programming*, pages 131–158. Springer, New York, NY, 1989. ISBN 978-1-4613-9619-2. doi: 10.1007/978-1-4613-9617-8_8.

Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4):575–601, 1992. doi: 10.1002/nla.2361.

Sanjay Mehrotra and Jie Sun. An Algorithm for Convex Quadratic Programming That Requires O(n3.5L) Arithmetic Operations. *Mathematics of Operations Research*, 15(2):342–363, May 1990. ISSN 0364-765X. doi: 10.1287/moor.15.2.342.

Youngjae Min and Navid Azizan. Hardnet: Hard-constrained neural networks with universal approximation guarantees. *arXiv preprint arXiv:2410.10807*, 2024.

Shinji Mizuno and Florian Jarre. Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation. *Mathematical Programming*, 84(1):105–122, January 1999. ISSN 1436-4646. doi: 10.1007/s10107980020a.

Renato D C Monteiro and Ilan Adler. Interior path following primal-dual algorithms. Part I: Linear programming. *Mathematical Programming*, 44(1-3):27–41, 1989. doi: 10.1007/BF01587075.

Alexis Montoison and Dominique Orban. Krylov.jl: A Julia basket of hand-picked Krylov methods. *Journal of Open Source Software*, 8(89):5187, 2023. doi: 10.21105/joss.05187.

Benedetta Morini. On Partial Cholesky Factorization and a Variant of Quasi-Newton Preconditioners for Symmetric Positive Definite Matrices. *Axioms*, 7(3):44, July 2018. ISSN 2075-1680. doi: 10.3390/axioms7030044.

Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Imposing hard constraints on deep networks: Promises and limitations, 2017. URL `https://arxiv.org/abs/1706.02025`.

Ion Necoara, Yu Nesterov, and Francois Glineur. Linear convergence of first order methods for non-strongly convex optimization. *Mathematical Programming*, 175:69–107, 2019.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Studies in Applied and Numerical Mathematics. SIAM, January 1994. ISBN 978-0-89871-319-0. doi: 10.1137/1.9781611970791.

Hoang T Nguyen and Priya L Donti. Fsnet: Feasibility-seeking neural network for constrained optimization with guarantees. *arXiv preprint arXiv:2506.00362*, 2025.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

Aurelio Ribeiro Leite Oliveira and Danny C Sorensen. A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. *Linear Algebra and its Applications*, 394:1–24, 2005. doi: doi:10.1016/j.laa.2004.08.019.

Francesco Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

Francesco Orabona and Dávid Pál. Coin betting and parameter-free online learning. *Advances in Neural Information Processing Systems*, 29, 2016.

Dominique Orban and Abel Soares Siqueira. LinearOperators.jl, February 2024. URL `https://github.com/JuliaSmoothOptimizers/LinearOperators.jl`.

Kaan Ozkara, Can Karakus, Parameswaran Raman, Mingyi Hong, Shoham Sabach, Branislav Kveton, and Volkan Cevher. MADA: Meta-adaptive optimizers through hyper-gradient descent. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=tASXcrMekp`.

Xiang Pan. Deepopf: deep neural networks for optimal power flow. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 250–251, 2021.

John Platt and Alan Barr. Constrained differential optimization. In *Neural information processing systems*, 1987.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Boris T Polyak. Introduction to optimization. 1987.

Spyridon Pougkakiotis and Jacek Gondzio. An interior point-proximal method of multipliers for convex quadratic programming. *Computational Optimization and Applications*, 78:307–351, 2021. doi: 10.1002/nla.2361.

Spyridon Pougkakiotis and Jacek Gondzio. An Interior Point-Proximal Method of Multipliers for Linear Positive Semi-Definite Programming. *Journal of Optimization Theory and Applications*, 192(1):97–129, January 2022. ISSN 1573-2878. doi: 10.1007/s10957-021-01954-4.

Michael J D Powell. A method for nonlinear constraints in minimization problems. In R Fletcher, editor, *Optimization*, pages 283–298. Academic Press, London, 1969.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training pinns: A loss landscape perspective. *arXiv preprint arXiv:2402.01868*, 2024.

R. T. Rockafellar. Augmented Lagrangians and Applications of the Proximal Point Algorithm in Convex Programming. *Mathematics of Operations Research*, 1(2):97–116, May 1976. ISSN 0364-765X. doi: 10.1287/moor.1.2.97.

Anton Rodomanov and Yurii Nesterov. Greedy quasi-newton methods with explicit superlinear convergence. *SIAM Journal on Optimization*, 31(1):785–811, 2021.

Anton Rodomanov and Yurii Nesterov. Rates of superlinear convergence for classical quasi-newton methods. *Mathematical Programming*, pages 1–32, 2022.

David Martınez Rubio. Convergence analysis of an adaptive method of gradient descent. *University of Oxford, Oxford, M. Sc. thesis*, 2017.

Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

Luiz-Rafael Santos, Fernando Rocha Villas-Bôas, Aurelio Ribeiro Leite Oliveira, and Clóvis Perin. Optimized choice of parameters in interior-point methods for linear programming. *Computational Optimization and Applications*, 73(2):535–574, June 2019. doi: 10.1007/s10589-019-00079-9.

Michael A Saunders. Cholesky-based Methods for Sparse Least Squares: The Benefits of Regularization. In Loyce Adams and John Lawrence Nazareth, editors, *Linear and Nonlinear Conjugate Gradient-Related Methods*, pages 92–100. SIAM, Philadelphia, 1996. ISBN 0-89871-376-5.

Michael A Saunders and John A Tomlin. Solving regularized linear programs using barrier methods and KKT systems. Technical Report IBM Research Report RJ 10064 and Stanford SOL Report 96-4, IBM Thomas J. Watson Research Center, 1996.

Lukas Schork and Jacek Gondzio. Implementation of an interior point method with basis preconditioning. *Mathematical Programming Computation*, 12(4):603–635, December 2020. ISSN 1867-2957. doi: 10.1007/s12532-020-00181-8.

Manan Tayal, Bhavya Giri Goswami, Karthik Rajgopal, Rajpal Singh, Tejas Rao, Jishnu Keshavan, Pushpak Jagtap, and Shishir Kolathaya. A collision cone approach for control barrier functions, 2024. URL `https://arxiv.org/abs/2403.07043`.

Madeleine Udell and Alex Townsend. Why Are Big Data Matrices Approximately Low Rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, January 2019. doi: 10.1137/18M1183480.

Pascal Van Hentenryck. Optimization learning. *arXiv preprint arXiv:2501.03443*, 2025.

Lieven Vandenberghe and Stephen Boyd. Semidefinite Programming. *SIAM Review*, 38(1):49–95, March 1996. ISSN 0036-1445. doi: 10.1137/1038003.

Xiaoyu Wang and Ya-xiang Yuan. On the convergence of stochastic gradient descent with bandwidth-based step size. *Journal of Machine Learning Research*, 24(48):1–49, 2023.

Ziyao Wang, Jianyu Wang, and Ang Li. Fedhyper: A universal and robust learning rate scheduler for federated learning with hypergradient descent. *arXiv preprint arXiv:2310.03156*, 2023.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In

2017 IEEE international conference on robotics and automation (ICRA), pages 1714–1721. IEEE, 2017.

Kristian Woodsend and Jacek Gondzio. Hybrid MPI/OpenMP Parallel Linear Support Vector Machine Training. *Journal of Machine Learning Research*, 10(67):1937–1953, 2009. URL `http://jmlr.org/papers/v10/woodsend09a.html`.

Kristian Woodsend and Jacek Gondzio. Exploiting separability in large-scale linear support vector machine training. *Computational Optimization and Applications*, 49(2):241–269, June 2011. ISSN 1573-2894. doi: 10.1007/s10589-009-9296-8.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Yinyu Ye and Edison Tse. An extension of Karmarkar's projective algorithm for convex quadratic programming. *Mathematical Programming*, 44(1):157–179, May 1989. ISSN 1436-4646. doi: 10.1007/BF01587086.

Juha Yli-Kaakinen and T Saramaki. A systematic algorithm for the design of multiplierless fir filters. In *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No. 01CH37196)*, volume 2, pages 185–188. IEEE, 2001.

Shengwei Zhang and AG Constantinides. Lagrange programming neural networks. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(7):441–452, 1992.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*, 2024.

Shipu Zhao, Zachary Frangella, and Madeleine Udell. NysADMM: Faster composite convex optimization via low-rank approximation. In *Proceedings of the 39th International Conference on Machine Learning*, pages 26824–26840. PMLR, June 2022. URL `https://proceedings.mlr.press/v162/zhao22a.html`.