Efficient numerical algorithms for large-scale conic optimization

Yuwen Chen

Lincoln College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

August 2024

Abstract

Conic optimization has been widely used in many areas, such as machine learning, image processing and automatic control. It usually relies on iterative algorithms to find the optimum numerically. Compared to linear programming and quadratic programming containing only linear constraints, conic optimization usually takes more time and is more prone to numerical instability. This thesis aims to accelerate computation for conic optimization from different perspectives. The more structural information we have, the more acceleration we can achieve in algorithmic design.

By exploiting both sparsity and low-rank information of a special class of semidefinite programmings, we reformulate the problems and propose an ADMM algorithm that is computational-friendly on GPU. Numerical results show that it is very efficient and scales up quite well to large dimensionality.

Next, we introduce Clarabel solver, which is a new general-purpose solver for conic optimization with the quadratic objective. It outperforms cutting-edge solvers on quadratic programming and second-order cone programming in terms of speed and robustness while also performs competitively on other conic optimization problems.

In the meantime, a novel augmentation method for a class of nonsymmetric cones sparsifies the linear system to factorize underlying interior point methods. This sparse augmentation can also be utilized for other conic solvers supporting the same nonsymmetric cones.

Finally, we propose an early termination saving computational time for mixed integer conic programming within the branch-and-bound framework. It demonstrates how to generate a dual feasible point from the infeasible iterates of primal-dual algorithms, which enables early termination for solvers based on the primal-dual algorithms, including operator splitting methods and interior point methods.

Efficient numerical algorithms for large-scale conic optimization

Yuwen Chen

Lincoln College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

August 2024

Acknowledgements

The last four years at Oxford have been a uniquely enriching experience. I am deeply grateful for the support and encouragement I received from many during my DPhil journey.

First, I must show my deep gratitude to my DPhil supervisor Prof Paul Goulart. He is an expert in optimization algorithms and an enthusiastic coder who sparked my interests in solver development. Paul is also an easygoing person who encourages me to explore my own interests in research while consistently offering his help in the meantime. Discussions with him have always been enlightening and enjoyable.

I would also like to thank the other professors in our group for their invaluable support during my DPhil. Prof. Mark Cannon consistently provided assistance during his lab tutorials. Prof. Antonis Papachristodoulou, Prof. Harrison Steel, and Prof. Jack Umenberger generously shared valuable visions and information throughout my time in the group.

My academic life was also enjoyable thanks to my peers in the control group. Thanks to Yana Lishkova, who is ardently helping group members and organizing the group events. Han Wang, Zhoudan Pan, Idris Kempf and Sebastian Steffen, with whom I always enjoyed office chats. Thanks to Licio Romao and Michael Garstka for helpful discussions and support at the start of my DPhil.

Further gratitude goes to Suli Zou, who introduced me to the field of optimization, and to my master's thesis supervisors, Antonio Orvieto and Aurelien Lucchi, who broadened my perspectives on various topics in optimization. Their support made my way to DPhil at Oxford.

Lastly, I would like to express my warm gratitude to Yao, who has been my steadfast companion through both challenges and joys over these years. I am grateful for my parents, Haiyan and Ling, and my grandpa Anzao, who just passed away, for their unwavering love and support. They have always stood by my side, enabling me to reach this milestone.

Yuwen Chen

Oxford, August 2024

Abstract

Conic optimization has been widely used in many areas, such as machine learning, image processing and automatic control. It usually relies on iterative algorithms to find the optimum numerically. Compared to linear programming and quadratic programming containing only linear constraints, conic optimization usually takes more time and is more prone to numerical instability. This thesis aims to accelerate computation for conic optimization from different perspectives. The more structural information we have, the more acceleration we can achieve in algorithmic design.

By exploiting both sparsity and low-rank information of a special class of semidefinite programmings, we reformulate the problems and propose an ADMM algorithm that is computational-friendly on GPU. Numerical results show that it is very efficient and scales up quite well to large dimensionality.

Next, we introduce Clarabel solver, which is a new general-purpose solver for conic optimization with the quadratic objective. It outperforms cutting-edge solvers on quadratic programming and second-order cone programming in terms of speed and robustness while also performs competitively on other conic optimization problems.

In the meantime, a novel augmentation method for a class of nonsymmetric cones sparsifies the linear system to factorize underlying interior point methods. This sparse augmentation can also be utilized for other conic solvers supporting the same nonsymmetric cones.

Finally, we propose an early termination saving computational time for mixed integer conic programming within the branch-and-bound framework. It demonstrates how to generate a dual feasible point from the infeasible iterates of primal-dual algorithms, which enables early termination for solvers based on the primal-dual algorithms, including operator splitting methods and interior point methods.

Notation

Sets

$\mathbb{R}$ . . . . . the set of real numbers

$\mathbb{R}_+$ ($\mathbb{R}_{++}$) . the nonnegative (positive) real numbers

$\mathbb{R}^n$ . . . . . the set of real n-dimensional vectors

$\mathbb{Z}$ . . . . . the set of integers $\{\dots, -2, -1, 0, 1, \dots\}$

$\mathbb{S}^n$ . . . . . the set of real $n \times n$ symmetric matrices

$\mathbb{S}^n_{++}$ ($\mathbb{S}^n_+$) . . the set of real symmetric positive (semi)definite matrices

$\text{int}(\mathcal{C})$ . . . the set of interior points of set $\mathcal{C}$

$\text{rint}(\mathcal{C})$ . . . the set of relative interior points of set $\mathcal{C}$

$\text{cl}(\mathcal{C})$ . . . . the closure of set $\mathcal{C}$

$\mathcal{K}^*$ . . . . . the dual cone of $\mathcal{K}$

$\mathcal{K}^\circ$ . . . . . the polar cone of $\mathcal{K}$

Operators

$\mathbf{1}^n$ . . . . . a n-dimensional vector of value 1

$|\mathbb{I}|$ . . . . . the number of elements in the discrete set $\mathbb{I}$

$[\![n]\!]$ . . . . . the set of $\{1, \dots, n\}$

$\lceil x \rceil$ . . . . . the smallest integer value that is bigger than or equal to x

$f'(x)$ . . . . 1st-order derivative of function $f(x)$

$f''(x)$ . . . . 2nd-order derivative of function $f(x)$

$f'''(x)$ . . . 3rd-order derivative of function $f(x)$

$\nabla^k(x)$ . . . k-th-order derivative of function $f(x)$

$\partial f(x)$ . . . . the subdifferential of function f at x

$\mathcal{I}_{\mathcal{X}}(x)$ . . . . the indicator function of set $\mathcal{X}$

$\Pi_{\mathcal{C}}(x)$ . . . . the projection of $x \in \mathbb{R}^n$ onto the set $\mathcal{C}$

$\text{prox}_f(x)$ . . the proximal operator w.r.t. function f at point x

$\sigma_{\mathcal{C}}(x)$ . . . . the support function of $\mathcal{C}$ at x

$f^*(x)$ . . . . the conjugate function of function $f(x)$

$\langle x, y \rangle$ . . . . the inner product of vector x, y.

x^{-k} . . . . . element-wise inverse of the vector $x \in \mathbb{R}^n$ of power k

$\text{diag}(x)$ . . . transforming the vector $x \in \mathbb{R}^n$ into a diagonal matrix $\text{diag}(x) \in \mathbb{S}^n$

$\text{mat}(x)$ . . . transforming the vector $x \in \mathbb{R}^{n(n+1)/2}$ to the matrix form $X \in \mathbb{S}^n$

$\text{Diag}(X)$ . . vectorizing the diagonal terms of matrix $X \in \mathbb{R}^{n \times n}$

$X_{i,\cdot}$ . . . . . i-th row of matrix X

$X_{\cdot,j}$ . . . . . j-th column of matrix X

$X \succeq 0$ . . . X is in the set $\mathbb{S}^n_+$

$X \succ 0$ . . . X is in the set $\mathbb{S}^n_{++}$

Norms

$\|x\|_2$ . . . . Euclidean norm of vector x: $\|x\|_2 := \sqrt{\langle x, x \rangle}$

$\|A\|_F$ . . . . Frobenius norm of matrix A: $\|A\|_F = \sqrt{\langle A, A \rangle} = \sqrt{\text{Tr}(A^\top A)}$

$\|A\|_p$ . . . . induced operator p-norm of matrix A: $\|A\|_p = \sup_{\|x\| \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$

$\|A\|$ . . . . . 2-operator norm of matrix A

$\|A\|_\infty$ . . . . infinite norm of matrix A: $\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$

Acronyms

ADMM . . . Alternating direction method of multipliers

BM . . . . . Burer-Monteiro method

B&B . . . . Branch-and-bound

CPU . . . . Central processing unit

GPU . . . . Graphics processing unit

HSDE . . . Homogeneous self-dual embedding

IPM . . . . Interior point method

KKT . . . . Karush-Kuhn-Tucker condition

LHSCB . . Logarithmically homogeneous self-concordant barrier

LP . . . . . Linear programming

MICP . . . Mixed integer conic programming

MIP . . . . Mixed integer programming

MPC . . . . Model predictive control

OSM . . . . Operator splitting method

PSD . . . . Positive semidefinite

QP . . . . . Quadratic programming

SDP . . . . Semidefinite programming

SOCP . . . Second-order cone programming

Contents

List of Figures

1

Introduction

Contents

The dissertation aims to design efficient and numerically stable algorithms for convex conic optimization problems of the following form:

$$\begin{aligned} \min_{x,s} \quad & \tfrac{1}{2}x^\top Px + q^\top x \\ \text{subject to:} \quad & Ax + s = b, \\ & s \in \mathcal{K}, \end{aligned} \tag{1.1}$$

The formulation above is very general and can model a wide range of common convex optimization problems depending on the particular choice of problem data and the choice of cone $\mathcal{K}$. It encompasses common problem classes such as linear programming (LP), quadratic programming (QP), second-order cone programming (SOCP) and semidefinite programming (SDP) problems, as well as problems with other exotic cones. Such problems appear in a wide variety of applications including constrained optimal control [1] and moving horizon estimation [2], limit analysis of engineering structures [3, 4] and fluid flows [5], image processing [6], support vector machines [7], lasso problems [8, 9], circuit design [10], portfolio optimization [11, 12], and many others.

1.1 Overview of conic optimization

Previously, most research on conic optimization has focussed on the class of symmetric cones that are homogeneous and self-dual, also called self-scaled [13, 14], such as second-order cones and positive semidefinite cones. Research on nonsymmetric cones has become increasingly popular in recent years since it can model more complex constraints than existing symmetric cones [15]. Moreover, a symmetric conic constraint can be reformulated as a nonsymmetric cone if additional properties are known to this cone, which can reduce the dimension of an optimization problem [16, 17].

Common modern approaches for solving conic optimization include first-order operator-splitting methods (OSM) [18, 19], and second-order interior-point methods (IPM) [14]. The former can be easily paralleled and is suitable for large-scale problems while the latter is preferred when high accuracy solutions are required.

OSMs like ADMM [20] split an optimization problem into several subproblems and then solve them sequentially for each iteration. These subproblems can be solved more efficiently compared to the original one. IPMs commonly employ ν-logarithmically-homogeneous self-concordant barrier (LHSCB) functions, which apply penalties on convex inequality constraints defined over cones. The parameter ν is related to the convergence speed of an IPM, where the total iteration number to ϵ-optimality is known to be $\mathcal{O}(\sqrt{\nu}\ln(1/\epsilon))$ in theory.

If we set an additional integer constraint over variable x, the problem (1.1) becomes a mixed integer conic programming (MICP) [21]. It is able to formulate discrete constraints in an optimization problem, such as hybrid model predictive control [22], portfolio optimization [23], power electronics [24] and robust truss topology [25]. The most common approach is to use the branch-and-bound (B&B) method, which solves a convex relaxation for each node, and requires an underlying solver for the convex relaxation. Since the number of nodes increases exponentially w.r.t. the count of discrete variables, various acceleration techniques have been developed to reduce the number of nodes we need to compute, like presolving [26] and feasibility pump [27], or shorten the computational time within a convex relaxation, like warm-start and early termination [28, 29].

1.2 Problems of interest

Semidefinite programming

Semidefinite programming is a powerful modeling tool used in various areas, like Maxcut in combinatorial optimization [30], signal processing [31] and sum of square in control [32]. However, the size of an SDP problem can grow rapidly, which makes it hard to solve. Although current general SDP solvers have exploited underlying properties such as low-rank information and chordal sparsity to accelerate the computation, they are not able to deal with huge SDPs when the dimension of an SDP exceeds tens of thousands unless the problem possesses additional structure. Instead of developing an algorithm for general SDPs, it is possible to transform an

SDP of a special class of constraints into an equivalent optimization problem that is able to be solved by an efficient algorithm [33].

Nonsymmetric cones

DDS [34] and Hypatia [35] interior-point solvers have already supported a bunch of nonsymmetric cones in use. However, they need to reduce a linear systems into an equivalent normal equation, which may become dense and fail to exploit the potential sparsity information in the factorization step.

Mixed integer conic programming

Dual feasible algorithms like the dual simplex and the dual active set methods can generate a dual feasible point that can terminate a node solve early if the current dual cost is larger than the best feasible solution at present [36], but they are not efficient on conic optimization. On the contrary, primal-dual algorithms are widely used for convex conic optimization, but they can not generate a dual feasible solution until solving the convex relaxation to optimality. Preliminary work on generating an intermediate dual feasible point has been done for primal-dual interior point methods, but it only works for mixed integer quadratic programming and is a heuristic technique without theoretical guarantee [29].

Development of interior-point solvers

The interior point method is one of the most widely used algorithms in convex continuous optimization solvers [37]. It usually incorporates the homogeneous self-dual embedding [38] which is able to detect infeasibility of a problem efficiently. However, the homogeneous self-dual embedding only holds for linear cost functions and we have to transform a quadratic cost into a second-order cone constraint, which will introduce additional fill-in and slow the convergence of IPMs, and the transformed problem is more prone to numerical instability due to the additional conic constraint.

1.3 Outline

In this thesis we focus on the development of efficient algorithms for conic optimization and aim to tackle the problems mentioned in the last section. Main contributions are organised into the following chapters:

Chapter 2

We first introduce some important definitions and properties related to convex sets, convex cones and convex functions. Then we detail duality theory with various form of optimality conditions for convex optimization problems. We end the chapter with fundamentals of optimization algorithms appearing in the thesis such as proximal operators, ADMM algorithms and interior point methods.

Chapter 3

In this chapter we propose a low-rank ADMM algorithm for solving diagonally constrained SDPs. It introduces a bilinear decomposition to the original problem such that all updates can be run in parallel. We prove that our algorithm converges to a first-order stationary point globally and show that the result can be strengthened to the convergence of a solution within $O(1/r)$ global optimality if the negative curvature is exploited. We also develop a proximal variant of the algorithm that can solve SDPs with block-diagonal constraints and prove its global convergence to a first-order stationary point. Our experiments show that the proposed algorithm and its proximal variant outperform the state-of-the-art coordinate descent method and Riemannian manifold algorithms at moderate accuracy, and both are suitable for large-scale SDPs on a GPU. The chapter is based on

- Yuwen Chen and Paul Goulart. "Burer-Monteiro ADMM for large-scale diagonally constrained SDPs". In *20th European Control Conference (ECC)*, 2022, pp. 66-71.
- Yuwen Chen and Paul Goulart. "Burer-Monteiro ADMM for large-scale SDPs". In *arXiv:2302.04016*, 2023.

Chapter 4

In this chapter we present a general-purpose solver, called *Clarabel*, for conic optimization with quadratic objectives. It uses the homogeneous embedding within the primal-dual interior point method, which avoids transforming a quadratic cost into a second-order cone due to the use of homogeneous self-dual embedding. Our open-source Julia/Rust implementation of Clarabel has been tested on multiple standard datasets including a wide variety of applications.[1] Numerical results show that our solver is competitive with the cutting-edge conic solvers and performs faster and more numerically stable over them on problems like QPs and SOCPs. The chapter is based on

- Paul Goulart and Yuwen Chen. "Clarabel: A conic interior-point solver with quadratic objectives". In *arXiv:2405.12762*, 2024.

Chapter 5

In this chapter we develop an efficient interior-point method for some nonsymmetric cones, which exploits the diagonal plus low-rank property inside Hessians of their barrier functions. We show that the proposed augmented decomposition ensures the quasidefiniteness of the augmented linear system and makes it factorizable under the sparse LDL factorization with static pivoting. The experimental results demonstrate that our sparse implementation for these cones performs much better than the Hypatia solver for nonsymmetric cones and opens the way to exploiting the sparsity of nonsymmetric cones in interior point methods.

- Yuwen Chen and Paul Goulart. "An efficient IPM implementation for a class of nonsymmetric cones". In *arXiv:2305.12275*, 2023.

[1] Paul contributes to most of code implementation. Yuwen primarily contributed to the methodology and the initial Julia development of nonsymmetric cones, as well as to enhancing the speed and numerical stability of Clarabel.

Chapter 6

In this chapter we propose an early termination technique to accelerate branch-and-bound for mixed integer conic programming with an underlying convex optimization solver based on primal-dual algorithms, including operator splitting methods and interior point methods. We show how to utilize existing dual iterates inside either an operator splitting method or an interior point method to generate a dual feasible point for early termination with little additional effort, and we provide sufficient conditions for the two proposed early termination techniques respectively. We also prove that our early termination technique can be applied directly to a hybrid MPC problem if input is bounded. Numerical results show that the proposed early termination method can reduce the total computational time for different MICPs consistently.

- Yuwen Chen and Paul Goulart. "An early termination technique for ADMM in mixed integer conic programming". In *20th European Control Conference (ECC)*, 2022, pp. 60-65.
- Yuwen Chen, Catherine Ning and Paul Goulart. "A unified early termination technique for primal-dual algorithms in mixed integer conic programming". In *IEEE Control Systems Letters*, 2023.

Chapter 7

We summarize the main contributions of the thesis in this chapter and suggest some directions for future research.

2

Background

Contents

In this chapter we introduce some definitions and results covering convex analysis, convex optimization, optimization algorithms and operator theory which will be used in subsequent chapters. These definitions are standard; further details can be found in [20, 37, 39–43].

2.1 Convexity

Convexity is an important notion of sets and functions in optimization theory. Many real-world applications can be formulated as optimization problems with convex sets and convex functions. The main property of convex optimization is that the first-order stationary condition is indeed a global optimality condition for convex optimization problems.

2.1.1 Definitions of convex set and cones

Definition 2.1.1 (Convex set)**.** A set $\mathcal{C} \subseteq \mathbb{R}^n$ is *convex* if, for every pair of points $x, y \in \mathcal{C}$, it includes the line segment joining them:

$$(1-\alpha)x + \alpha y \in \mathcal{C}, \forall \alpha \in [0,1].$$

A subclass of convex sets is closed under nonnegative scalar multiplication:

Definition 2.1.2 (Convex cone)**.** A set $\mathcal{K} \subseteq \mathbb{R}^n$ is a *convex cone* if it is convex and closed under nonnegative scaling with scalars $\alpha_1, \alpha_2 \geq 0$, i.e.

$$\alpha_1 x + \alpha_2 y \in \mathcal{K} \quad \forall x, y \in \mathcal{K}.$$

There are several key definitions w.r.t. to a cone $\mathcal{K}$ [44]:

Definition 2.1.3 (Proper cone)**.** A convex cone $\mathcal{K}$ in $\mathbb{R}^m$ is *proper* if it

- is closed,
- is pointed: $\mathcal{K} \cap (-\mathcal{K}) = \{0\}$,
- has nonempty interior.

There are several important cones related to a given convex set $\mathcal{C}$:

Definition 2.1.4 (Dual cone)**.** The *dual cone* $\mathcal{C}^*$ of a convex set $\mathcal{C} \subseteq \mathbb{R}^n$ is defined as

$$\mathcal{C}^* := \left\{ y \in \mathbb{R}^n \mid \inf_{x \in \mathcal{C}} \langle x, y \rangle \geq 0 \right\}.$$

The dual cone is always closed and convex, even if $\mathcal{C}$ is not. If $\mathcal{C}$ is a cone and satisfies $\mathcal{C} = \mathcal{C}^*$, the cone is called *self-dual*.

Definition 2.1.5 (Polar cone)**.** The *polar cone* $\mathcal{C}^\circ$ of a convex set $\mathcal{C} \subseteq \mathbb{R}^n$ is defined as

$$\mathcal{C}^\circ := \left\{ y \in \mathbb{R}^n \mid \sup_{x \in \mathcal{C}} \langle x, y \rangle \leq 0 \right\}. \tag{2.1}$$

Note that the polar cone is equal to the negative dual cone, i.e. $\mathcal{C}^\circ = -\mathcal{C}^*$.

Definition 2.1.6 (Recession cone)**.** The *recession cone* $\mathcal{C}^\infty$ of a convex set $\mathcal{C} \subseteq \mathbb{R}^n$ is defined as

$$\mathcal{C}^\infty := \{ y \in \mathbb{R}^n \mid x + \alpha y \in \mathcal{C}, \forall x \in \mathcal{C}, \alpha \geq 0 \}.$$

The recession cone contains all directions y of half-lines that start in $x \in \mathcal{C}$ and remain inside $\mathcal{C}$. Note that the recession cone of a bounded set is the set that only contains the origin (Corollary 6.52 in [42]).

Definition 2.1.7 (Normal cone)**.** The *normal cone* $N_\mathcal{C}(x)$ of a convex set $\mathcal{C} \subseteq \mathbb{R}^n$ at a point $x \in \mathcal{C}$ is defined as

$$N_\mathcal{C}(x) := \left\{ y \in \mathbb{R}^n \mid \sup_{\bar{x} \in \mathcal{C}} \langle \bar{x} - x, y \rangle \leq 0 \right\}. \tag{2.2}$$

The normal cone at any $x \in \mathcal{C}$ is always convex and we have $N_\mathcal{C}(x) = 0, \ \forall x \in \text{int}(\mathcal{C})$ and (Proposition 6.45 in [42]). If the set $\mathcal{C}$ is itself a cone, then it is related to the polar cone by $N_\mathcal{C}(x) = \mathcal{C}^\circ \cap \{x\}^\perp$ where $\{x\}^\perp = \{y \in \mathbb{R}^n : y \perp x\}$ (Eq.(6.38) in [42]).

2.1.2 Convex optimization

Definition 2.1.8 (Convex function)**.** A function $f : \mathcal{X} \to \mathbb{R}$ is convex if the domain $\mathcal{X}$ of f is a convex set and it holds that

$$f\left(\alpha x_1 + (1-\alpha)x_2\right) \leq \alpha f\left(x_1\right) + (1-\alpha) f\left(x_2\right),$$

for all pairs of points $x_1, x_2 \in \mathcal{X}$ and $\alpha \in [0,1]$.

A convex optimization problem is a minimization problem of the form

$$\begin{aligned} \min \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m, \\ & g_j(x) = 0, \quad j = 1, \ldots, p, \end{aligned} \tag{2.3}$$

where each of the functions $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ is convex, and each of the functions $g_j : \mathbb{R}^n \mapsto \mathbb{R}$ is affine. The function f_0 is called the cost (objective function) while the functions $f_i, \forall i \in [\![m]\!]$ and $g_j, \forall j \in [\![p]\!]$ are referred to the inequality and equality constraints respectively. The set of points satisfying all constraints in (2.3) is referred to as the *feasible set*. We say that the problem is *infeasible* if the feasible set is empty.

2.2 Convex conic optimization

The inequality constraints in (2.3) can be rewritten as $f_i(x) + s_i = 0, s_i \geq 0$. We can generalize the inequality constraints $s_i \geq 0$ to conic set constraints $s_i \in \mathcal{K}$. Setting f_0 to the quadratic cost and $f_i, \forall i \in [\![m]\!]$ to affine mappings yields a standard form of conic optimization problem:

$$\begin{aligned} p^* := \min_{x,s} \quad & \frac{1}{2} x^\top P x + q^\top x \\ \text{s.t.} \quad & Ax + s = b, \ s \in \mathcal{K} \end{aligned} \tag{2.4}$$

with $P \in \mathbb{S}^n_+, A \in \mathbb{R}^{m \times n}, q \in \mathbb{R}^n, b \in \mathbb{R}^m$. Note that the affine conic form in (2.4) can also represents linear equality constraints when we set $s = 0$.

2.2.1 Dual problem

We define the *Lagrangian* w.r.t. (2.4) as

$$L(x, s, y) := \frac{1}{2}x^\top Px + q^\top x + y^\top(Ax + s - b),$$

where y is the multiplier of the constraint in (2.4). The dual function is defined as the minimization of the Lagrangian over primal variables x, s:

$$g(x, y) := \inf_{x,s\in\mathcal{K}} L(x, s, y).$$

Taking the maximization over dual variables x, y yields the dual problem of (2.4)

$$\begin{aligned} d^* := \max_{x,y} \quad & -\frac{1}{2}x^\top Px - b^\top y \\ \text{s.t.} \quad & Px + A^\top y + q = 0,\ y \in \mathcal{K}^*. \end{aligned} \tag{2.5}$$

2.2.2 Conic constraints

In (2.4), a small family of convex cones is sufficient to express almost all convex optimization problems appearing in practice [40]:

Zero cone

The *zero cone* of dimension n is denoted as

$$\{0\}^n := \{x \in \mathbb{R}^n \mid x_i = 0, \forall i = 1, \ldots, n\}.$$

The dual cone of the zero cone is $(\{0\}^n)^* = \mathbb{R}^n$.

Nonnegative cone

The *nonnegative cone* of dimension n is defined as

$$\mathbb{R}^n_+ := \{x \in \mathbb{R}^n \mid x_i \geq 0,\ \forall i = 1, \ldots, n\}.$$

The nonnegative cone is a self-dual convex cone, i.e. $\left(\mathbb{R}^n_+\right)^* = \mathbb{R}^n_+$.

Second-order cone

The *second-order cone*, also sometimes called the *quadratic* or *Lorentz cone*, is defined as

$$\mathcal{K}_{\text{soc}}^n := \left\{(t, x) \;\middle|\; x \in \mathbb{R}^{n-1}, t \in \mathbb{R}_+, \|x\|_2 \leq t\right\},$$

or abbreviated as $\mathcal{K}_{\text{soc}}$ if dimension is obvious from the context. The second-order cone is self-dual, i.e. $\mathcal{K}_{\text{soc}} = \mathcal{K}_{\text{soc}}^*$.

Meanwhile, the *rotated second-order cone* of dimension n is defined as

$$\mathcal{K}_{\text{rsoc}}^n := \left\{(u, v, x) \;\middle|\; x \in \mathbb{R}^{n-2}, u, v \geq 0, \|x\|_2^2 \leq 2uv\right\}.$$

It can be obtained by applying a transformation

$$T := \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & I_{n-2} \end{bmatrix}$$

that rotates the vector $(u, v, x) \in \mathcal{K}_{\text{soc}}^n$ by 45° in the (u, v)-plane, and we can easily verify that

$$y \in \mathcal{K}_{\text{soc}}^n \Leftrightarrow Ty \in \mathcal{K}_{\text{rsoc}}^n .$$

Semidefinite cone

The *positive semidefinite cone* is defined as

$$\mathbb{S}_+^n := \left\{X \in \mathbb{S}^n \;\middle|\; v^\top X v \geq 0 \quad \forall v \in \mathbb{R}^n\right\}.$$

It is the set of symmetric matrices with nonnegative eigenvalues and we also denote $X \in \mathbb{S}_+^n$ as $X \succeq 0$. The positive semidefinite cone is self-dual, i.e. $\left(\mathbb{S}_+^n\right)^* = \mathbb{S}_+^n$.

Exponential cone

The *exponential cone* is a 3-dimensional cone defined as

$$\mathcal{K}_{\text{exp}} := \left\{(x, y, z) \;\middle|\; y > 0, y \exp\left(\frac{x}{y}\right) \leq z\right\} \cup \{(x, 0, z) \mid x \leq 0, z \geq 0\}.$$

with its dual cone given by

$$\mathcal{K}_{\text{exp}}^* = \left\{(u, v, w) \;\middle|\; u < 0, -u \exp\left(\frac{v}{u}\right) \leq \exp(1) w\right\} \cup \{(0, v, w) \mid v \geq 0, w \geq 0\}.$$

Here the second term in both unions makes the respective cone proper.

Power cone

The 3-dimensional *power cone* is defined as

$$\mathcal{K}_{\text{pow},\alpha} = \left\{(x, y, z) \;\middle|\; x^\alpha y^{1-\alpha} \geq |z|, x \geq 0, y \geq 0\right\}.$$

with the exponent $\alpha \in (0, 1)$. Its dual cone is given by

$$\mathcal{K}^*_{\text{pow},\alpha} = \left\{(u, v, w) \;\middle|\; \left(\frac{u}{\alpha}\right)^\alpha \left(\frac{v}{1-\alpha}\right)^{1-\alpha} \geq |w|, u \geq 0, v \geq 0\right\}.$$

2.2.3 Optimality conditions

Weak duality always holds regardless of convexity of the primal problem (2.4), i.e. $p^* \geq d^*$. Strong duality ($p^* = d^*$) holds when the primal problem (2.4) is convex and satisfies an appropriate constraint qualification, like the *Slater's condition* that there exists x, s satisfy $Ax + s = b$ and $s \in \text{rint}(\mathcal{K})$. Based on the strong convexity, we list four equivalent optimality conditions for problem (2.4) below.

KKT condition

Strong convexity implies that the *complementary slackness* condition holds at an optimal solution (x^*, s^*, y^*), i.e. $\langle s^*, y^* \rangle = 0$. Considering the constraints in both primal (2.4) and dual (2.5) problems along with the complementary slackness constraint yields the *Karush-Kuhn-Tucker* (KKT) condition for (x, s, y),

$$\begin{aligned} Ax + s &= b, \\ Px + A^\top y + q &= 0, \\ s \in \mathcal{K}, y &\in \mathcal{K}^*, \\ \langle s, y \rangle &= 0, \end{aligned} \tag{2.6}$$

which is a necessary and sufficient optimality condition for the convex problem (2.4).

Linear complementarity problems

Note that the complementary slackness condition $\langle s, y \rangle = 0$ can be rewritten as $s \perp y$ denoting s is orthogonal to y. Therefore, finding (x, s, y) satisfying the KKT

condition is equivalent to solving the following linear complementarity problem (LCP):

$$\mathbb{R}^n \times \mathcal{K}^* \ni \begin{bmatrix} x \\ y \end{bmatrix} \perp \underbrace{\begin{bmatrix} P & A^\top \\ -A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} q \\ b \end{bmatrix}}_{M(z):=} \in \{0\}^n \times \mathcal{K}, \tag{2.7}$$

where $z := [x; y] \in \mathbb{R}^{m+n}$.

Variational inequality problems

A function $F(x) : \mathbb{R}^n \to \mathbb{R}^n$ is called *monotone* on set $\mathcal{C} \subset \operatorname{dom} F$ if $\mathcal{C}$ is convex and

$$\langle F(x) - F(y), x - y \rangle \geq 0, \ \forall x, y \in \mathcal{C},$$

and it is said to be a *maximal monotone operator* if there is no monotone operator that properly contains it. Note that the operator $M(z)$ defined in (2.7) is a maximal monotone operator [42] on $\mathbb{R}^{m+n}$ and we denote $\mathcal{C} := \mathbb{R}^n \times \mathcal{K}^*$. The LCP problem (2.7) is then equivalent to the classical variational inequality problem (Example 26.21-26.22 in [42]), i.e.

$$\exists z \in \mathcal{C}, \text{ such that } \langle u - z, M(z) \rangle \geq 0, \ \forall u \in \mathcal{C}. \tag{2.8}$$

Sum of maximal monotone operators

For the solution z of (2.8), we have

$$\langle u - z, -M(z) \rangle \leq 0, \ \forall u \in \mathcal{C},$$

which implies $-M(z) \in N_{\mathcal{C}}(z)$ given the definition of normal cone (2.2). Hence, solving (2.8) is equivalent to finding z that satisfies

$$0 \in M(z) + N_{\mathcal{C}}(z), \tag{2.9}$$

where both $M(z)$ and $N_{\mathcal{C}}(z)$ are maximal monotone operators in [44, Example 12.7 and Theorem 12.17]. In summary, all of the optimality conditions above are equivalent and we would check different optimality conditions in convergence analysis depending on which algorithm we want to use.

2.3 Algorithms

After formulating a real-world application into a standard convex problem we usually need an iterative algorithm to find an optimum of it, which is equivalent to finding a solution satisfying one of the optimality conditions mentioned in Section 2.2.3. There are many algorithms to solve convex optimization problems, including operator splitting methods and interior point methods. The former is related to monotone operator theory and fixed point evaluation while the latter relies on the Newton method to solve nonlinear equations.

2.3.1 Proximal operator

Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$, the *proximal operator* is defind as $\text{prox}_f : \mathbb{R}^n \to \mathbb{R}^n$,

$$\text{prox}_f(v) = \underset{x}{\text{argmin}} \left(f(x) + \frac{1}{2} \|x - v\|_2^2 \right), \tag{2.10}$$

and the scaled version with regularization $\lambda > 0$ as

$$\text{prox}_{\lambda f}(v) = \underset{x}{\text{argmin}} \left(f(x) + \frac{1}{2\lambda} \|x - v\|_2^2 \right), \tag{2.11}$$

The solution of (2.11) is unique since the function minimized is strongly convex. When f is the indicator function

$$I_{\mathcal{C}}(x) = \begin{cases} 0 & x \in \mathcal{C} \\ +\infty & x \notin \mathcal{C} \end{cases}$$

where $\mathcal{C}$ is a closed nonempty convex set, the proximal operator of f becomes the Euclidean projection onto $\mathcal{C}$, i.e.

$$\Pi_{\mathcal{C}}(v) = \underset{x \in \mathcal{C}}{\text{argmin}} \|x - v\|_2.$$

Proximal operators can also be viewed as generalized projections when we choose different regularization function other than $\|\cdot\|_2^2$ in (2.10). Note that the prox_f operator is a *nonexpansive* operator, i.e.

$$\left\| \text{prox}_f(x) - \text{prox}_f(y) \right\| \leq \|x - y\|, \forall x, y \in \mathbb{R}^n.$$

This property is quite useful in the analysis of operator splitting methods later on.

Moreau decomposition

The convex *conjugate* function of f is defined as

$$f^*(y) = \sup_x \left(\langle y, x \rangle - f(x) \right).$$

The *Moreau decomposition* connects the proximal operator of f to its conjugate by

$$v = \text{prox}_f(v) + \text{prox}_{f^*}(v). \tag{2.12}$$

When we choose f as the indicator function of a closed convex cone $\mathcal{K}$, we have

$$v = \Pi_{\mathcal{K}}(v) + \Pi_{\mathcal{K}^\circ}(v), \tag{2.13}$$

where $\mathcal{K}^\circ$ is the polar cone as defined in (2.1). It implies that the projection of any point v onto the cone $\mathcal{K}$ is orthogonal to its projection onto the polar cone $\mathcal{K}^\circ$.

2.3.2 Operator splitting methods

The general idea of operator splitting methods is to regard the convex optimization problem as the problem of finding a zero of the sum of monotone operators, e.g.

$$\text{Find } z, \quad 0 \in F(z) := A(z) + B(z), \tag{2.14}$$

where $A(z), B(z)$ are maximal monotone operators [44]. We then can split them into problems that can be handled by efficient proximal operators separately.

Convex optimization can be connected to monotone operator theory via the following theorem:

Theorem 2.3.1 (Theorem 12.17 in [44])**.** *If $f(x) < \infty$ for at least one $x \in \mathbb{R}^n$, and $f(x) > -\infty$ for all $x \in \mathbb{R}^n$, we call f a proper function. The function f is lower semicontinuous at $\bar{x}$ if*

$$\liminf_{x \to \bar{x}} f(x) \geq f(\bar{x}), \text{ or equivalently } \liminf_{x \to \bar{x}} f(x) = f(\bar{x}),$$

and lower semicontinuous on $\mathbb{R}^n$ if this holds for every $\bar{x} \in \mathbb{R}^n$.

For any proper, convex function $f : \mathbb{R}^n \to \mathbb{R}$, the subgradient mapping $\partial f : \mathbb{R}^n \to \mathbb{R}^n$ is monotone. Indeed, a proper, lower semicontinuous function f is convex if and only if ∂f is monotone, in which case ∂f is maximal monotone. Such a function f is almost strictly convex if and only if ∂f is strictly monotone.

For example, suppose we have an optimization problem of the form below,

$$\min_x \quad f(x) + h(x), \tag{2.15}$$

where $f(x)$ and $h(x)$ are both convex lower semicontinuous functions. The optimality condition is

$$0 \in \partial f(x) + \partial h(x),$$

which satisfies (2.14), since $\partial f(x)$ and $\partial h(x)$ are both maximal monotone operators [44, Theorem 12.17]. Looking back to the optimality condition (2.9), it can also be interpreted as finding a zero z of the sum of two operators since both $M(z)$ and $N_{\mathcal{C}}(z)$ are maximal-monotone operators, since the normal cone is the subdifferential, the set of subgradients, of the indicator function of convex set $\mathcal{C}$.

Solving (2.14) is equivalent to finding a fixed point of the *resolvent* of $F(z)$,

$$R_{\lambda F}(z) := (I + \lambda F)^{-1}(z). \tag{2.16}$$

When $\lambda \geq 0$ and $F(z)$ is monotone, the resolvent $R_{\lambda F}(z)$ is a nonexpansive operator [42]. Note that (2.16) is a proximal operator when $F(z)$ is set to be the subdifferential of a convex function. It deals with $A(z), B(z)$ separately since the individual resolvents can be computed more efficiently than the resolvent on the sum $F(z)$ directly.

Operator splitting methods, like the popular alternating direction method of multipliers (ADMM), rely on the resolvent operators (proximal operators accordingly) of $A(z), B(z)$. We can reformulate (2.15) as

$$\begin{aligned} \min \quad & f(x) + h(z) \\ s.t. \quad & x = z, \end{aligned} \tag{2.17}$$

with the augmented Lagrangian defined as

$$L_\rho(x, z, y) := f(x) + h(z) + \langle y, x - z\rangle + \frac{\rho}{2}\|x - z\|_2^2 .$$

The ADMM algorithm optimizes subproblems w.r.t. x and z alternatively and then updates the dual variable y in each iteration,

$$\begin{aligned} x^{k+1} &= \mathrm{prox}_{\frac{1}{\rho}f}\left(z^k - \frac{1}{\rho}y^k\right), \\ z^{k+1} &= \mathrm{prox}_{\frac{1}{\rho}h}\left(x^{k+1} + \frac{1}{\rho}y^k\right), \\ y^{k+1} &= y^k + \rho\left(x^{k+1} - z^{k+1}\right). \end{aligned} \tag{2.18}$$

ADMM is suitable for the case when prox_f and prox_h are computationally efficient to evaluate, e.g. either $f(x)$ or $h(x)$ is decomposable that the computation of subproblems can be performed in parallel. It is well suited for large-scale optimization and various applications can be found in [20].

2.3.3 Interior point methods

The interior point method is another popular method that is used in modern solvers [40, 41, 45, 46]. It penalizes the conic constraint $\mathcal{K}$ via a logarithmically-homogeneous self-concordant barrier function, which helps to transform the complementary slackness constraint in the KKT condition (2.6) into a nonlinear equation. The Newton method is able to solve the nonlinear equation with conic constraints after the transformation. Note that the iterate (x, s, y) should always stay in the interior of conic constraints, i.e. $s \in \mathrm{int}(\mathcal{K}), y \in \mathrm{int}(\mathcal{K}^*)$, which is guaranteed by the line-search during the update of an interior point method.

Logarithmically-homogeneous self-concordant barrier function

A function is called *ν-logarithmically-homogeneous self-concordant barrier* (LHSCB) function for cone $\mathcal{K}$, if it satisfies

$$\begin{aligned} &|\nabla^3 f(s)[r, r, r]| \le 2\left(\nabla^2 f(s)[r, r]\right)^{3/2} && \forall s \in \mathrm{int}(\mathcal{K}), r \in \mathbb{R}^d, \\ &f(\lambda s) = f(s) - \nu\log(\lambda) && \forall s \in \mathrm{int}(\mathcal{K}), \lambda > 0, \end{aligned} \tag{2.19}$$

where $\nabla^2 f(s)[r,r]$ is the second order directional derivative and $\nabla^3 f(s)[r,r,r]$is the third order directional derivative along the direction r. We call ν is the *degree* of f. The convex conjugate f^* is defined as

$$f^*(y) := \sup_{s\in\text{int}(\mathcal{K})} \{-\langle y,s\rangle - f(s)\}, \tag{2.20}$$

which is also a ν-LHSCB for $\mathcal{K}^*$ [13] and we call it *conjugate barrier*. The gradient ∇f^* of f^* is the solution of

$$\nabla f^*(y) := -\arg\sup_{s\in\text{int}(\mathcal{K})} \{-\langle y,s\rangle - f(s)\}. \tag{2.21}$$

There are some key properties of LHSCB $f(s), \forall s \in \text{int}(\mathcal{K})$ [13],

$$\begin{aligned} \nabla f(\tau s) = \frac{1}{\tau}\nabla f(s), \quad \nabla^2 f(\tau s) = \frac{1}{\tau^2}\nabla^2 f(s), \\ \nabla^2 f(s)s = -\nabla f(s), \quad \nabla^3 f(s)[s] = -2\nabla^2 f(s), \\ \langle \nabla f(s), s\rangle = -\nu, \quad \left\langle \nabla^2 f(s)s, s\right\rangle = \nu, \\ \left\langle \nabla f(s), \left[\nabla^2 f(s)\right]^{-1}\nabla f(s)\right\rangle = \nu, \end{aligned} \tag{2.22}$$

and its relation with the conjugate barrier $f^*(y), \forall y \in \text{int}(\mathcal{K}^*)$:

$$\begin{aligned} -\nabla f(s) \in \text{int}(\mathcal{K}^*), -\nabla f^*(y) \in \text{int}(\mathcal{K}), \\ f^*\left(-\nabla f(s)\right) = -\nu - f(s), \quad f\left(-\nabla f^*(y)\right) = -\nu - f^*(y), \\ \nabla f^*\left(-\nabla f(s)\right) = -s, \quad \nabla f\left(-\nabla f^*(y)\right) = -y, \\ \nabla^2 f\left(-\nabla f^*(y)\right) = \left[\nabla^2 f^*(y)\right]^{-1}, \quad \nabla^2 f^*\left(-\nabla f(s)\right) = \left[\nabla^2 f(s)\right]^{-1}, \\ f(s) + f^*(y) \geq -\nu + \nu\ln\nu - \nu\ln\langle s,y\rangle. \end{aligned} \tag{2.23}$$

Central path

Primal-dual interior point methods aim to solve the KKT optimality condition (2.6). Suppose both the primal problem (2.4) and the dual problem (2.5) are feasible and f is the barrier function for $\mathcal{K}$. Given an initial point (s^0, y^0) that are interior to conic constraints, i.e. $s^0 \in \text{int}(\mathcal{K}), y^0 \in \text{int}(\mathcal{K}^*)$, the residuals are denoted as $r_p^0 := Ax^0 + s^0 - b, r_d^0 := Px^0 + A^\top y^0 + q$. We define the *primal central path* as

$$(x_\mu, s_\mu) = \text{argmin}\left\{\begin{matrix}\min_{x,s} & \frac{1}{\mu}(\frac{1}{2}x^\top Px + q^\top x) - {r_d^0}^\top x + f(s) \\ & Ax + s - b = \mu r_p^0,\end{matrix}\right\}. \tag{2.24}$$

which is uniquely defined by μ since the objective function in (2.24) is strictly convex when $P \succ 0$. The KKT optimality condition of (2.24) implies the dual multiplier λ_μ satisfies

$$\frac{1}{\mu}(Px_\mu + q) + A^\top \lambda_\mu = r_d^0, \nabla f(s_\mu) = -\lambda_\mu, \tag{2.25}$$

where λ_μ is the dual multiplier of problem (2.24). Likewise, we define the *dual central path* as

$$(x_\mu, y_\mu) = \text{argmin} \begin{Bmatrix} \min_{x,s} \ \frac{1}{\mu}(\frac{1}{2}x^\top Px + b^\top y) - {r_p^0}^\top y + f^*(y) \\ Px + A^\top y + q = \mu r_d^0, \end{Bmatrix}. \tag{2.26}$$

and the optimality condition of (2.26) yields (x_μ, y_μ) satisfying

$$\frac{1}{\mu}Px_\mu + P\chi_\mu = 0, \frac{1}{\mu}b + \nabla f^*(y_\mu) + A^\top \chi_\mu = r_p^0, \tag{2.27}$$

where χ_μ is the dual multiplier of (2.26). By replacing $\lambda_\mu = y_\mu/\mu$ in (2.25), we obtain

$$y_\mu = -\mu \nabla f(s_\mu), \ f^*(y_\mu) = f^*(-\mu \nabla f(s_\mu)) = f^*\left(-\nabla f\left(\frac{s_\mu}{\mu}\right)\right) = -\frac{s_\mu}{\mu}$$

via (2.23). Hence, we can unify (2.25) and (2.27) as the *primal-dual central path*

$$\begin{aligned} Ax_\mu + s_\mu - b &= \mu r_p^0, \\ Px_\mu + A^\top y_\mu + q &= \mu r_d^0, \\ y_\mu = -\mu \nabla f(s_\mu), \ s_\mu &= -\mu \nabla f^*(y_\mu). \end{aligned} \tag{2.28}$$

The point $v(\mu) := (x_\mu, s_\mu, y_\mu)$ on the central path is the solution of the KKT condition (2.6) and s_μ, y_μ satisfies the complementary slackness in the limit of $\mu \to 0$, due to

$$\langle s_\mu, y_\mu \rangle = -\mu \langle s_\mu, \nabla f(s_\mu) \rangle = \mu\nu,$$

where ν is the degree of cone $\mathcal{K}$ (and $\mathcal{K}^*$).

Path-following interior point methods start with an iterate (x^0, s^0, y^0) on the central path and every iterate $(x^k, s^k, y^k), k \geq 0$ should stay in a neighbourhood of the central path. The iterates (x^k, s^k, y^k) will finally converge to the point on the central path at $\mu = 0$ in the limit, which is also the optimal solution of the original problem. A primal-dual interior point method converges to ϵ optimality within $O(\sqrt{\nu}\log(1/\epsilon))$ iterations [47, 48].

3

A parallel algorithm for block-diagonally constrained semidefinite programs

Contents

3.1 Introduction

SDPs [49] are widely used in various fields such as control engineering [32], signal processing [31], combinatorial optimization [30, 50] and finance [51]. While interior-point methods [52] can be used to solve small to medium size SDPs in polynomial time, they do not scale well with dimension of the problem. At present, several methods have been proposed to tackle the scalability of SDPs [53]. One is to exploit the chordal decomposition of a SDP [54, 55], which typically reduce a single cone constraint to a collection of constraints on lower dimensional cones. This technique has been successfully implemented for interior-point methods [16] and ADMM-based methods [19, 56]. Approaches based on diagonal dominance and scaled diagonal dominance have been studied for solving SDP approximately and shown to be effective in the SDP relaxation of the sum-of-square problems [57]. Another direction is to exploit low-rank information of a SDP [58–61]. Burer-Monteiro method [59] is a popular one, which substitutes $X \succeq 0$ with a low-rank factorization $X = VV^\top$ where $V \in \mathbb{R}^{n\times r}$ with $r \ll n$, and it saves both computational time and data storage. Frank-Wolfe algorithms [62], which are projection-free and circumvent a full eigenvalue decomposition, have also been proposed to solve large-scale SDPs and exploit low-rank information via in-face exploitation [60]. Meanwhile, some work has been done on the storage issue of SDPs. The sketching method, which has been well-studied in numerical linear algebra [63], is applied to SDPs [61, 64]. Based on approximate complementary slackness, [65] proposes a method to iteratively estimate a low-rank eigenspace and then optimize the SDP with reduced dimension. In addition, SDPs can be reformulated as a equivalent problem that can be solved efficiently, e.g, formulating diagonally constrained SDPs as a nonconvex QP (3.4) and solving it by block-coordinate descent methods [33, 66].

The Burer-Monteiro (BM) method is a popular low-rank method for SDPs and works well in practice [59]. For the standard primal SDP problem

$$\min\{\langle C, X\rangle : \langle A_i, X\rangle = b_i, i = 1, \ldots, m, X \succeq 0\}, \tag{3.1}$$

the classical BM approach replaces the positive semidefinite constraint $X \succeq 0$ with the factorization $X = VV^\top$ where $V \in \mathbb{R}^{n\times r}$, so that the problem becomes

$$\min\{\langle C, VV^\top\rangle : \langle A_i, VV^\top\rangle = b_i, i = 1, \ldots, m\}. \tag{3.2}$$

for very large scale problems with low rank solutions. Compared with (3.1), (3.2) requires storage of $O(nr)$ of V which is much less than $O(n^2)$ of X when $n \gg r$. Replacing X with V also avoids the expensive computation related to the positive semidefinite constraint. The problem (3.2) is nonconvex due to the existence of quadratic equality constraints and global optimality is hard to guarantee, although a global optimum is usually obtained using the BFGS algorithm [59]. Generally speaking, the rank r is selected according to the rank-inequality constraint in SDPs [58, 67, 68]. Such a choice ensures that any local minimum is a global optimum for almost all cost matrices C in (3.3) [69] and is tight as shown in [70].

The ADMM algorithm, which is closely related to the alternating direction method and the augmented Lagrangian method, has become very popular in recent years for use in the large-scale optimization [20]. Convergence proofs of ADMM in convex optimization typically rely on the design of a Lyapunov function [20],[71] which is monotonically decreasing, or the interpretation of it as a variant of the Douglas Rachford (DR) algorithm [19, 72, 73] whose convergence is proved based on monotone operator theory [42]. ADMM often works extremely well even for nonconvex problems [74, 75], such as nonnegative matrix factorization [76], optimal power flow [77] and image reconstruction [78] but convergence analysis is much less mature. In recent years, some results have been developed for the convergence of nonconvex ADMM to first-order stationary points in limited situations [71, 79–81].

In this Chapter:

1. We propose an ADMM algorithm with a novel bilinear decomposition to the Burer-Monteiro approach for diagonally constrained SDPs in Section 3.2. We prove convergence of this algorithm in *value* in Section 3.3 and in *iterates* in Section 3.4.

2. In Section 3.5, we show that exploiting negative curvature guarantees convergence to a solution of $O(1/r)$ global optimality, where r is the rank selected for the Burer-Monteiro decomposition.

3. In Section 3.6, we introduce a proximal variant of the algorithm that can also solve block-diagonally constrained SDPs with global convergence to a first-order stationary point.

4. Finally, numerical results show both algorithm and its proximal variant perform better than the state-of-the-art Riemannian manifold algorithms and scale well w.r.t. the dimension of SDPs in Section 3.7.

3.2 Bilinear decomposition and ADMM algorithm

We first consider semidefinite programming problems with diagonal constraints in the following form,

$$\begin{aligned} \min \quad & \langle C, X \rangle \\ \text{s.t} \quad & X_{ii} = 1, \text{for } i \in [\![n]\!], \\ & X \in \mathbb{S}^n_+, \end{aligned} \tag{3.3}$$

where $C \in \mathbb{S}^n$. This kind of SDP applies to many applications, including the max-cut problem [82], graphical model inference [83] and the community detection problem [84].

It is well known that (3.3) is equivalent to [33, 66]:

$$\begin{aligned} \min \quad & f(\sigma) := \langle C, \sigma\sigma^\top \rangle \\ \text{s.t.} \quad & \|\sigma_i\| = 1, \text{for } i \in [\![n]\!], \end{aligned} \tag{3.4}$$

if $r \geq \lceil \sqrt{2n} \rceil$, where $\sigma := [\sigma_1, \sigma_2, ..., \sigma_n]^\top \in \mathbb{R}^{n \times r}$ and $\sigma_i \in \mathbb{R}^r$ is the i-th row of σ. We denote the unit norm constraint set as $\mathcal{M} = \{[\sigma_1, \sigma_2, \ldots, \sigma_n]^\top \in \mathbb{R}^{n \times r} \mid \|\sigma_i\| = 1,\ i \in [\![n]\!]\}$. The problem (3.4) is inspired by the popular Burer Monteiro (BM) method, which substitutes the SDP constraint $X \succeq 0$ with a low-rank factorization $X = \sigma\sigma^\top$. The diagonal constraints $X_{ii} = 1, \text{for } i \in [\![n]\!]$ are replaced by norm constraints $\|\sigma_i\| = 1, \text{for } i \in [\![n]\!]$. It is easy to see that the constraint set is decoupled

while the objective function is coupled w.r.t $\sigma_i, \forall i \in [\![n]\!]$. In this paper, we introduce a bilinear decomposition for (3.4) as in [76]. Instead of setting $X = \sigma\sigma^\top$, we decompose the variable X as a bilinear term, $X = \sigma\tilde{\sigma}^\top$ subject to $\sigma = \tilde{\sigma}$, resulting in the following equivalent problem formulation:

$$\begin{aligned} &\min \ \langle C, \tilde{\sigma}\sigma^\top \rangle \\ &\text{s.t. } \tilde{\sigma} \in \mathcal{M}, \ \tilde{\sigma} = \sigma. \end{aligned} \tag{3.5}$$

We define the corresponding augmented Lagrangian as:

$$\begin{aligned} L_\rho(\tilde{\sigma}, \sigma, y) :=& \langle C, \tilde{\sigma}\sigma^\top \rangle + \langle y, \tilde{\sigma} - \sigma \rangle + \frac{\rho}{2}\|\tilde{\sigma} - \sigma\|_F^2 + \sum_{i=1}^n \mathcal{I}_{\|\tilde{\sigma}_i\|=1}(\tilde{\sigma}_i) \\ =& \langle C, \tilde{\sigma}\sigma^\top \rangle + \sum_{i=1}^n \left[\frac{\rho}{2}\|\tilde{\sigma}_i - \sigma_i\|^2 + y_i^\top(\tilde{\sigma}_i - \sigma_i) + \mathcal{I}_{\|\tilde{\sigma}_i\|=1}(\tilde{\sigma}_i) \right], \end{aligned} \tag{3.6}$$

where $\rho > 0$ and $y := [y_1, y_2, \ldots, y_n]^\top \in \mathbb{R}^{n\times r}$ is the dual variable for the equality constraint $\tilde{\sigma} = \sigma$ and $\tilde{\sigma} := [\tilde{\sigma}_1, \tilde{\sigma}_2, ..., \tilde{\sigma}_n]^\top \in \mathbb{R}^{n\times r}$. A standard ADMM algorithm given this splitting is then

$$\tilde{\sigma}^{k+1} = \operatorname*{argmin}_{\tilde{\sigma}\in\mathcal{M}} \ L_\rho(\tilde{\sigma}, \sigma^k, y^k), \tag{3.7a}$$

$$\sigma^{k+1} = \operatorname*{argmin}_{\sigma} \ L_\rho(\tilde{\sigma}^{k+1}, \sigma, y^k), \tag{3.7b}$$

$$y^{k+1} = y^k + \rho(\tilde{\sigma}^{k+1} - \sigma^{k+1}). \tag{3.7c}$$

For the ADMM algorithm above, step (3.7a) corresponds to the solution of n decomposable nonconvex QPs w.r.t. $\tilde{\sigma}_i$ for $\forall i \in [\![n]\!]$,

$$\tilde{\sigma}_i^{k+1} = \operatorname*{argmin}_{\|\tilde{\sigma}_i\|=1} \ \frac{\rho}{2}\|\tilde{\sigma}_i - \sigma_i^k\|^2 + {y_i^k}^\top\tilde{\sigma}_i + \sum_{j=1}^n C_{i,j}\langle\sigma_j^k, \tilde{\sigma}_i\rangle, \tag{3.8}$$

and has a closed-form solution due to $\|\tilde{\sigma}_i\| = 1, \forall i \in [\![n]\!]$,

$$\tilde{\sigma}^{k+1} \leftarrow \text{Normalize-Row}\left(\sigma^k - \frac{1}{\rho}(y^k + C\sigma^k)\right), \tag{3.9}$$

which is to normalize each row vector into unit length. We will require the following assumption to ensure that (3.9) is valid everywhere.

Assumption 3.2.1. *For $\forall i \in [\![n]\!]$, $\|\gamma_i^k\|$ is nonzero for any iteration k where γ_i^k is the i-th row of*

$$\gamma^k := \sigma^k - \frac{1}{\rho}(y^k + C\sigma^k), \tag{3.10}$$

i.e. $\gamma^k = [\gamma_1^k, \cdots, \gamma_n^k]^\top$.

In Lemma 3.3.2, we will show that Assumption 3.2.1 is always satisfied for every iteration k given an appropriate choice of ρ. In addition, step (3.7b) is an unconstrained QP and amounts to

$$\sigma^{k+1} \leftarrow \tilde{\sigma}^{k+1} + \frac{1}{\rho}(y^k - C\tilde{\sigma}^{k+1}). \tag{3.11}$$

Our approach, outlined in (3.7), can therefore be implemented as in Algorithm 1.

Algorithm 1 ADMM Burer-Monteiro algorithm (ADMM-BM)

1: Initialization: set $\sigma^0 = \tilde{\sigma}^0 \in \mathcal{M}, y^0 = C\tilde{\sigma}^0$.
2: **while** *termination criteria not satisfied* **do**
3: $\quad \gamma^k \leftarrow \sigma^k - \frac{1}{\rho}(y^k + C\sigma^k)$
4: $\quad \tilde{\sigma}^{k+1} \leftarrow$ Normalize-Row(γ^k)
5: $\quad \sigma^{k+1} \leftarrow \tilde{\sigma}^{k+1} + \frac{1}{\rho}(y^k - C\tilde{\sigma}^{k+1})$
6: $\quad y^{k+1} \leftarrow y^k + \rho(\tilde{\sigma}^{k+1} - \sigma^{k+1})$
7: **end while**

Algorithm 1 has a nice property that connects the primal variable $\tilde{\sigma}$ with the dual variable y for any $k \geq 1$, i.e.

$$y^k = y^{k-1} + \rho(\tilde{\sigma}^k - \sigma^k) = C\tilde{\sigma}^k, \tag{3.12}$$

where the last equality comes from step 3 in Algorithm 1. Note that our algorithm avoids inverting the cost matrix C, in contrast to the SOC algorithm proposed in [74], and operations can be easily paralleled on a GPU. In addition, we can prove the Algorithm 1 converges theoretically to a first-order stationary point.

3.3 Convergence of the objective value

A first order stationary point of (3.4) can be defined [71, Def. 3.6] as

$$-(C\tilde{\sigma}^*)^\top_{i,\cdot} \in \frac{\partial I_{\|\tilde{\sigma}_i^*\|=1}(\tilde{\sigma}_i^*)}{\partial \tilde{\sigma}_i^*}, \ \forall i \in [\![n]\!], \tag{3.13}$$

where $\tilde{\sigma}^*$ belongs to the set of first-order stationary points and $(C\tilde{\sigma}^*)_{i,\cdot}$ denotes the i-th row of the matrix $C\tilde{\sigma}^*$. The main result of this section is then the following theorem:

Theorem 3.3.1. *If we choose* $\rho \geq \max\{\alpha\|C\|_\infty, \beta\|C\|\}$ *such that* $\alpha > 0, \beta > 0$ *satisfy* $\frac{(\alpha^2-4\alpha-2)\beta}{2\alpha^2} - \frac{1}{\beta} > 0$*, then the augmented Lagrangian sequence* $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)_{k=1}^{+\infty}$ *converges to the objective function* $\langle C\tilde{\sigma}^k, \tilde{\sigma}^k\rangle$ *in Algorithm 1, under Assumption 3.2.1. Moreover, every convergent subsequence of* $\{\tilde{\sigma}^k\}$ *converges globally to* Ω*, the set of first-order stationary points of the problem* (3.4)*, and such a convergent subsequence exists.*

Theorem 3.3.1 guarantees the convergence of Algorithm 1. The proof of Theorem 3.3.1 is composed of three parts. We first prove monotonic decrease of the augmented Lagrangian $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$ in Lemma 3.3.3; we then derive a lower bound for $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k), \forall k$ in Lemma 3.3.4. Finally, we use Lemmas 3.3.3 and 3.3.4 to prove convergence of $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$ and $\lim_{k\to\infty} \|\tilde{\sigma}^k - \sigma^k\| \to 0$, which implies convergence to the set of first-order stationary points.

3.3.1 Part I: Monotonic decrease of $L_\rho(\tilde{\sigma}, \sigma, y)$

The key to proving Theorem 3.3.1 is showing that $L_\rho(\tilde{\sigma}, \sigma, y)$ is monotonically decreasing, i.e.

$$L_\rho(\tilde{\sigma}^k, \sigma^k, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^{k+1}) > 0, \quad \forall k. \tag{3.14}$$

In the remainder of this section, we first characterize the iteration-wise change of $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$ in Lemma 3.3.1 and then prove that the change is non-negative if ρ is properly lower bounded in Lemma 3.3.3, which implies (3.14) is valid.

Change of $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$

From (3.9), we have $\tilde{\sigma}_i^{k+1} = \frac{\gamma_i^k}{\|\gamma_i^k\|}, \forall i \in [\![n]\!]$ which says γ_i^k is aligned with $\tilde{\sigma}_i^{k+1}$ without necessarily being a unit vector itself. The change of $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$ per iteration is then summarized in Lemma 3.3.1 below:

Lemma 3.3.1. *For any iteration* k*, we have*

$$\begin{aligned} &L_\rho(\tilde{\sigma}^k, \sigma^k, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^{k+1}) \\ &\geq \left(\frac{\rho \min_{i\in[\![n]\!]}\{\|\gamma_i^k\|\}}{2} - \frac{\|C\|^2}{\rho}\right)\|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\|_F^2 + \frac{\rho}{2}\|\sigma^{k+1} - \sigma^k\|_F^2. \end{aligned} \tag{3.15}$$

Proof. We split the right hand side of our inequality into three parts, i.e.

$$\begin{aligned}
&L_\rho(\tilde{\sigma}^k, \sigma^k, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^{k+1}) \\
=&\underbrace{L_\rho(\tilde{\sigma}^k, \sigma^k, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^k, y^k)}_{(A)} + \underbrace{L_\rho(\tilde{\sigma}^{k+1}, \sigma^k, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^k)}_{(B)} \\
&+ \underbrace{L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^{k+1})}_{(C)},
\end{aligned}$$

and will consider them in turn. Part (A) arises from the minimization over $\tilde{\sigma}$ in Algorithm 1. This minimization is a nonconvex optimization problem due to $\|\tilde{\sigma}_i\| = 1, \forall i \in [\![n]\!]$, but we can obtain the global optimum by the row-wise normalization over γ^k. Suppose we define $L_{\rho,i}(\tilde{\sigma}_i, \sigma, y)$ as

$$L_{\rho,i}(\tilde{\sigma}_i, \sigma, y) := \frac{\rho}{2}\|\tilde{\sigma}_i - \sigma_i\|^2 + {y_i}^\top(\tilde{\sigma}_i - \sigma_i) + \sum_{j=1}^{n} C_{i,j}\langle\sigma_j, \tilde{\sigma}_i\rangle. \tag{3.16}$$

Note that $L_{\rho,i}(\tilde{\sigma}_i, \sigma^k, y^k)$ in (3.16) is differentiable and ρ-strongly convex w.r.t. $\tilde{\sigma}_i$. Hence, we have

$$\begin{aligned}
&L_{\rho,i}(\tilde{\sigma}_i^k, \sigma^k, y^k) \\
\geq&L_{\rho,i}(\tilde{\sigma}_i^{k+1}, \sigma^k, y^k) + \langle\nabla L_{\rho,i}(\tilde{\sigma}_i^{k+1}, \sigma^k, y^k), \tilde{\sigma}_i^k - \tilde{\sigma}_i^{k+1}\rangle + \frac{\rho}{2}\|\tilde{\sigma}_i^k - \tilde{\sigma}_i^{k+1}\|^2, \\
=&L_{\rho,i}(\tilde{\sigma}_i^{k+1}, \sigma^k, y^k) + \rho\langle\tilde{\sigma}_i^{k+1} - \gamma_i^k, \tilde{\sigma}_i^k - \tilde{\sigma}_i^{k+1}\rangle + \frac{\rho}{2}\|\tilde{\sigma}_i^k - \tilde{\sigma}_i^{k+1}\|^2, \\
=&L_{\rho,i}(\tilde{\sigma}_i^{k+1}, \sigma^k, y^k) + \rho(1 - \|\gamma_i^k\|)\langle\tilde{\sigma}_i^{k+1}, \tilde{\sigma}_i^k - \tilde{\sigma}_i^{k+1}\rangle + \frac{\rho}{2}\|\tilde{\sigma}_i^k - \tilde{\sigma}_i^{k+1}\|^2,
\end{aligned} \tag{3.17}$$

where both equalities above come from the definition of γ^k in (3.10). Noting that

$$\begin{aligned}
\langle\tilde{\sigma}_i^{k+1}, \tilde{\sigma}_i^k - \tilde{\sigma}_i^{k+1}\rangle = \langle\tilde{\sigma}_i^{k+1}, \tilde{\sigma}_i^k\rangle - 1 &= \langle\tilde{\sigma}_i^{k+1}, \tilde{\sigma}_i^k\rangle - \frac{1}{2}(\|\tilde{\sigma}_i^{k+1}\|^2 + \|\tilde{\sigma}_i^k\|^2) \\
&= -\frac{1}{2}\|\tilde{\sigma}_i^{k+1} - \tilde{\sigma}_i^k\|^2,
\end{aligned}$$

which uses $\|\tilde{\sigma}_i^{k+1}\| = \|\tilde{\sigma}_i^k\| = 1$. Then, (3.17) becomes

$$L_{\rho,i}(\tilde{\sigma}_i^k, \sigma^k, y^k) \geq L_{\rho,i}(\tilde{\sigma}_i^{k+1}, \sigma^k, y^k) + \frac{\rho\|\gamma_i^k\|}{2} \cdot \|\tilde{\sigma}_i^k - \tilde{\sigma}_i^{k+1}\|^2. \tag{3.18}$$

For part (B), we can establish monotonic decrease when updating σ,

$$\begin{aligned}
&L_\rho(\tilde{\sigma}^{k+1}, \sigma^k, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^k) \\
=&\langle\tilde{\sigma}^{k+1}, C(\sigma^k - \sigma^{k+1})\rangle - \langle y^k, \sigma^k - \sigma^{k+1}\rangle + \frac{\rho}{2}\|\tilde{\sigma}^{k+1} - \sigma^k\|_F^2
\end{aligned}$$

$$
\begin{aligned}
&- \frac{\rho}{2}\|\tilde{\sigma}^{k+1} - \sigma^{k+1}\|_F^2 \\
=&\langle C\tilde{\sigma}^{k+1} - y^k, \sigma^k - \sigma^{k+1}\rangle + \frac{\rho}{2}\|\tilde{\sigma}^{k+1} - \sigma^k\|_F^2 - \frac{\rho}{2}\|\tilde{\sigma}^{k+1} - \sigma^{k+1}\|_F^2 \\
\overset{(3.11)}{=}&\rho\langle \tilde{\sigma}^{k+1} - \sigma^{k+1}, \sigma^k - \sigma^{k+1}\rangle + \frac{\rho}{2}\|\tilde{\sigma}^{k+1} - \sigma^k\|_F^2 - \frac{\rho}{2}\|\tilde{\sigma}^{k+1} - \sigma^{k+1}\|_F^2 \\
=&\frac{\rho}{2}\|\tilde{\sigma}^{k+1} - \sigma^k\|_F^2 - \frac{\rho}{2}\langle \tilde{\sigma}^{k+1} - \sigma^{k+1}, \tilde{\sigma}^{k+1} - \sigma^k + \sigma^{k+1} - \sigma^k\rangle \\
=&\frac{\rho}{2}\|\tilde{\sigma}^{k+1} - \sigma^k\|_F^2 - \frac{\rho}{2}\left(\|\tilde{\sigma}^{k+1} - \sigma^k\|_F^2 - \|\sigma^{k+1} - \sigma^k\|_F^2\right) \\
=&\frac{\rho}{2}\|\sigma^{k+1} - \sigma^k\|_F^2. \qquad (3.19)
\end{aligned}
$$

Finally, for part (C) we have

$$
\begin{aligned}
L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^{k+1}) &= \langle y^k - y^{k+1}, \tilde{\sigma}^{k+1} - \sigma^{k+1}\rangle \\
&= -\frac{1}{\rho}\|y^{k+1} - y^k\|_F^2, \qquad (3.20)
\end{aligned}
$$

where the last equality comes from the update rule for y^k, i.e. step 4 in Algorithm 1. Furthermore, we can obtain

$$
L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^{k+1}) \geq -\frac{1}{\rho}\|C\|^2 \cdot \|(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k)\|_F^2 \qquad (3.21)
$$

from (3.12). Combining (3.18), (3.19) and (3.21), we finally obtain (3.15). □

Choice of ρ

From Lemma 3.3.1, it is sufficient to show that $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$ is monotonically decreasing in (3.14) if the coefficient $\frac{\rho \min_{i\in[\![n]\!]}\{\|\gamma_i^k\|\}}{2} - \frac{\|C\|^2}{\rho} > 0$ while $\|\gamma_i^k\|$ is dependent on the choice of row i and iteration k. We argue that by choosing ρ properly, $\|\gamma_i^k\|$ can be uniformly lower-bounded in Lemma 3.3.2 and then $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$ always satisfies the monotonically decreasing condition (3.14) in Lemma 3.3.3.

From (3.10), we expect that the magnitude of $\|\gamma_i^k\|$ will depend strongly on the choice of ρ, and $\|\gamma_i^k\| \to 1,\ \forall i \in [\![n]\!]$ if ρ is set to be sufficiently large as $\sigma \to \tilde{\sigma}$ with $\|\tilde{\sigma}_i\| = 1,\ \forall i \in [\![n]\!]$. In Lemma 3.3.2, we provide a lower bound for ρ such that $\|\gamma_i^k\|, \forall i \in [\![n]\!]$ is not too small for any k.

Lemma 3.3.2. *Suppose $\rho \geq \alpha\|C\|_\infty, \alpha > 0$, then*

$$
\|\gamma_i^k\| \geq 1 - \frac{4}{\alpha} - \frac{2}{\alpha^2},\ \forall i,\ \forall k \geq 2.
$$

Proof. We note that, $\forall i \in [\![n]\!]$, $\forall k \geq 2$,

$$\sigma_i^{k+1} = \tilde{\sigma}_i^{k+1} + \frac{1}{\rho}(y_i^k - \sum_{j=1}^n C_{i,j}\tilde{\sigma}_j^{k+1}) \overset{(3.12)}{=} \tilde{\sigma}_i^{k+1} + \frac{1}{\rho}\sum_{j=1}^n C_{i,j}(\tilde{\sigma}_j^k - \tilde{\sigma}_j^{k+1}).$$

Substituting into (3.10), we can write γ_i^k in terms of $\tilde{\sigma}$ as

$$\begin{aligned}\gamma_i^k &= \sigma_i^k - \frac{1}{\rho}(y_i^k + \sum_{j=1}^n C_{i,j}\sigma_j^k) \overset{(3.12)}{=} \sigma_i^k - \frac{1}{\rho}\sum_{j=1}^n C_{i,j}(\tilde{\sigma}_j^k + \sigma_j^k) \\ &= \tilde{\sigma}_i^k + \frac{1}{\rho}\sum_{j=1}^n C_{i,j}(\tilde{\sigma}_j^{k-1} - \tilde{\sigma}_j^k) - \frac{1}{\rho}\sum_{j=1}^n C_{i,j}\tilde{\sigma}_j^k \\ &\quad - \frac{1}{\rho}\sum_{j=1}^n C_{i,j}\left[\tilde{\sigma}_j^k + \frac{1}{\rho}\sum_{l=1}^n C_{j,l}(\tilde{\sigma}_l^{k-1} - \tilde{\sigma}_l^k)\right],\end{aligned}$$

where the last equality is based on (3.11) and (3.12), the update of σ^k. Since $\|\tilde{\sigma}_i^k\| = 1,\ \forall i \in [\![n]\!], \forall k \geq 1$ from step 2 of Algorithm 1, we can bound the norm of $\|\gamma_i^k\|$ by

$$\begin{aligned}\|\gamma_i^k\| &\geq 1 - \frac{4}{\rho}\sum_{j=1}^n |C_{i,j}| - \frac{1}{\rho}\sum_{j=1}^n |C_{i,j}| \cdot \left(\frac{2}{\rho}\sum_{l=1}^n |C_{j,l}|\right) \\ &\geq 1 - \frac{4}{\rho}\sum_{j=1}^n |C_{i,j}| - \frac{1}{\rho}\sum_{j=1}^n |C_{i,j}| \cdot \frac{2}{\alpha} \\ &\geq 1 - \frac{4}{\alpha} - \frac{2}{\alpha^2},\end{aligned}$$

where the second inequality relies on $\rho \geq \alpha\|C\|_\infty$. □

Note that C is in the linear objective function and can be scaled such that we do not need a large parameter ρ to satisfy $\rho \geq \alpha\|C\|_\infty$ in Lemma 3.3.2. Meanwhile, Assumption 3.2.1 will be satisfied automatically when $k \geq 2$, as long as $1 - \frac{4}{\alpha} - \frac{2}{\alpha^2} > 0$.

Based on (3.15) and Lemma 3.3.2, we can establish that $L_\rho(\tilde{\sigma}, \sigma, y)$ is monotonically decreasing given a proper ρ in Lemma 3.3.3.

Lemma 3.3.3. *If we set $\rho \geq \max\{\alpha\|C\|_\infty, \beta\|C\|\}$ and $\alpha > 0, \beta > 0$ satisfy $\kappa := \frac{(\alpha^2 - 4\alpha - 2)\beta}{2\alpha^2} - \frac{1}{\beta} > 0$, then the augmented Lagrangian sequence $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$ is monotonically decreasing $\forall k \geq 2$, and satisfies*

$$L_\rho(\tilde{\sigma}^k, \sigma^k, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^{k+1}) \geq \kappa\|C\| \cdot \|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\|_F^2 + \frac{\rho}{2}\|\sigma^{k+1} - \sigma^k\|_F^2.$$

Proof. First, Lemma 3.3.2 is satisfied under the condition $\rho \geq \alpha\|C\|_\infty$. Then, combining Lemma 3.3.2 with (3.15) we obtain

$$\begin{aligned}
&L_\rho(\tilde{\sigma}^k, \sigma^k, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^{k+1}) \\
\geq &\left[\frac{\alpha^2 - 4\alpha - 2}{2\alpha^2} \cdot \rho - \frac{\|C\|^2}{\rho}\right] \|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\|_F^2 + \frac{\rho}{2}\|\sigma^{k+1} - \sigma^k\|_F^2 \\
\geq &\kappa\|C\| \cdot \|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\|_F^2 + \frac{\rho}{2}\|\sigma^{k+1} - \sigma^k\|_F^2.
\end{aligned}$$

□

3.3.2 Part II: Lower bound of $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$

We next establish a lower bound on $\lim_{k\to\infty} L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$ in the following lemma:

Lemma 3.3.4. *For $k \geq 2$, we have*

$$L_\rho(\tilde{\sigma}^k, \sigma^k, y^k) \geq -n\|C\|_\infty. \tag{3.22}$$

Proof of Lemma 3.3.4.

$$\begin{aligned}
L(\tilde{\sigma}^k, \sigma^k, y^k) &= \langle C\tilde{\sigma}^k, \sigma^k\rangle + \langle y^k, \tilde{\sigma}^k - \sigma^k\rangle + \frac{\rho}{2}\|\tilde{\sigma}^k - \sigma^k\|_F^2 \\
&= \langle C\tilde{\sigma}^k, \tilde{\sigma}^k\rangle + \langle y^k - C\tilde{\sigma}^k, \tilde{\sigma}^k - \sigma^k\rangle + \frac{\rho}{2}\|\tilde{\sigma}^k - \sigma^k\|_F^2 \\
&\overset{(3.12)}{=} \langle C\tilde{\sigma}^k, \tilde{\sigma}^k\rangle + \frac{\rho}{2}\|\tilde{\sigma}^k - \sigma^k\|_F^2 \\
&\geq \langle C\tilde{\sigma}^k, \tilde{\sigma}^k\rangle = \sum_{i=1}^n \sum_{j=1}^n C_{i,j}\langle \tilde{\sigma}_i^k, \tilde{\sigma}_j^k\rangle \overset{\|\tilde{\sigma}_i^k\|=1}{\geq} -\sum_{i=1}^n \sum_{j=1}^n |C_{i,j}| \geq -n\|C\|_\infty.
\end{aligned}$$

□

3.3.3 Proof of Theorem 3.3.1

We can now prove Theorem 3.3.1 given the results of Lemma 3.3.3 and Lemma 3.3.4.

Proof of Theorem 3.3.1. Since $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$ is monotonically decreasing (Lemma 3.3.3) and lower bounded (Lemma 3.3.4), $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$ converges to a constant value

due to the monotone convergence theorem. As a consequence, both $\|\sigma^{k+1} - \sigma^k\|_F$ and $\|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\|_F$ converge to 0 from Lemma 3.3.3. We also have

$$\|y^{k+1} - y^k\|_F \leq \|C\|_F \|(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k)\|_F,$$

due to (3.12). Hence, $\|y^{k+1} - y^k\|_F$ also converges to 0 as $k \to \infty$. In addition, $\|y^{k+1} - y^k\|_F$ can be written as

$$\|y^{k+1} - y^k\|_F = \rho \|\tilde{\sigma}^{k+1} - \sigma^{k+1}\|_F$$

due to the dual update (3.7c). Since $\|y^{k+1} - y^k\|_F \to 0$, we obtain $\|\tilde{\sigma}^{k+1} - \sigma^{k+1}\|_F \to 0$ for constant ρ. According to (3.22), we have

$$L_\rho(\tilde{\sigma}^k, \sigma^k, y^k) = \langle C\tilde{\sigma}^k, \tilde{\sigma}^k \rangle + \frac{\rho}{2} \|\tilde{\sigma}^k - \sigma^k\|_F^2.$$

Due to the convergence of $\|\tilde{\sigma}^k - \sigma^k\|_F \to 0$, we obtain

$$L_\rho(\tilde{\sigma}^k, \sigma^k, y^k) \to \langle C\tilde{\sigma}^k, \tilde{\sigma}^k \rangle.$$

Moreover, the optimality condition of (3.7a) can be shown to imply that

$$0 \in (C\sigma^k)_{i,\cdot}^\top + y_i^k + \rho(\tilde{\sigma}_i^{k+1} - \sigma_i^k) + \frac{\partial I_{\|\tilde{\sigma}_i\|=1}(\tilde{\sigma}_i^{k+1})}{\partial \tilde{\sigma}_i^{k+1}}, \tag{3.23}$$

following the same argument as in [85]. Since $\|\tilde{\sigma}_i^{k+1} - \tilde{\sigma}_i^k\| \to 0$ and $\|\tilde{\sigma}_i^k - \sigma_i^k\| \to 0$, we obtain

$$\lim_{k \to +\infty} \operatorname{dist}\left(-(C\sigma^k)_{i,\cdot}^\top - y_i^k, \frac{\partial I_{\|\tilde{\sigma}_i\|=1}(\tilde{\sigma}_i^{k+1})}{\partial \tilde{\sigma}_i^{k+1}} \right) = 0 \tag{3.24}$$

and hence

$$\lim_{k \to +\infty} \operatorname{dist}\left(-2(C\tilde{\sigma}^k)_{i,\cdot}^\top, \frac{\partial I_{\|\tilde{\sigma}_i\|=1}(\tilde{\sigma}_i^k)}{\partial \tilde{\sigma}_i^k} \right) = 0 \tag{3.25}$$

due to $\|\sigma^{k+1} - \sigma^k\|_F \to 0$, $\|\tilde{\sigma}^k - \sigma^k\|_F \to 0$ and (3.12), which means $\tilde{\sigma}^k$ lies in the set of first-order stationary points as defined in (3.13) when $k \to \infty$. In addition, the compactness of $\mathcal{M}$ implies that there is a subsequence of $\tilde{\sigma}^k$ that converges to a first-order stationary point. □

3.4 Convergence of the sequence $\tilde{\sigma}^k$

Theorem 3.3.1 established convergence of the sequence of Lagrangian values $L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$. We next consider the behavior of the sequence of iterates $\tilde{\sigma}^k$. We show that this sequence converges to a first-order stationary point in the following theorem, which is the main result of this section:

Theorem 3.4.1. *If we set* $\rho \geq \max\{\alpha\|C\|_\infty, \beta\|C\|\}$ *and* $\alpha > 0, \beta > 0$ *satisfy* $\frac{(\alpha^2-4\alpha-2)\beta}{2\alpha^2} - \frac{1}{\beta} > 0$*, then the sequence* $(\tilde{\sigma}^k)_{k\in\mathbb{N}}$ *generated by Algorithm 1 converges to a critical point* $\bar{\sigma}$ *of* f*. Moreover, the sequence* $(\tilde{\sigma}^k)_{k\in\mathbb{N}}$ *has a finite length, i.e.* $\sum_{k=0}^{+\infty} \|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\| < \infty$.

Our result relies on the Kurdyka-Łojasiewicz property, defined as follows:

Definition 3.4.1 (Kurdyka-Łojasiewicz (KL) property [85])**.** The function f is said to have the *Kurdyka-Łojasiewicz property* at $\bar{x} \in \operatorname{dom} \partial f$ if there exist $\eta \in (0, +\infty]$, a neighborhood U of $\bar{x}$ and a continuous concave function $\varphi : [0, \eta) \to \mathbb{R}_+$ such that:

- $\varphi(0) = 0$,

- φ is C^1 on $(0, \eta)$,

- for all $s \in (0, \eta)$, $\varphi'(s) > 0$,

- and for all x in $U \cap [f(\bar{x}) < f < f(\bar{x}) + \eta]$, the Kurdyka-Łojasiewicz inequality holds

$$\varphi'(f(x) - f(\bar{x}))\operatorname{dist}(0, \partial f(x)) \geq 1.$$

Moreover, f is called a *KL function* if it satisfies KL property at each point of $\operatorname{dom} \partial f$.

A class of nonsmooth functions called *functions definable in an o-minimal structure* satisfies Definition 3.4.1. We can easily verify that problems considered in this work are composed of polynomial components and then satisfies KL property [86, Section 4.3]. When a function f is known to be a KL function, we can prove the

convergence of an algorithm to critical points by checking the conditions of the following lemma:

Lemma 3.4.1 (Theorem 2.9 in [85]). *Suppose $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a proper lower semicontinuous function and $(x^k)_{k\in\mathbb{N}}$ is a sequence satisfying all of the following:*

- *(C1) Sufficient decrease condition: $\forall k \in \mathbb{N}$,*

$$f(x^{k+1}) + a\|x^{k+1} - x^k\|^2 \leq f(x^k);$$

- *(C2) Relative error condition: $\forall k \in \mathbb{N}, \exists\ g^{k+1} \in \partial f(x^{k+1})$ such that*

$$\|g^{k+1}\| \leq b\|x^{k+1} - x^k\|;$$

- *(C3) Continuity condition: There exists a subsequence $(x^{k_j})_{j\in\mathbb{N}}$ and $\bar{x}$ such that, when $j \to \infty$,*

$$x^{k_j} \to \bar{x} \text{ and } f(x^{k_j}) \to f(\bar{x}).$$

Here, a, b are positive constant. If f has the Kurdyka-Łojasiewicz property at the cluster point $\bar{x}$ specified in (C3), then $\bar{x}$ is a critical point of f and the sequence $(x^k)_{k\in\mathbb{N}}$ converges to it as $k \to \infty$. Moreover, the sequence $(x^k)_{k\in\mathbb{N}}$ has a finite length, i.e.

$$\sum_{k=0}^{+\infty} \|x^{k+1} - x^k\| < \infty.$$

The lemma above is the key component to prove Theorem 3.4.1.

Our proof will follow the approach [81], [85] and proceeds in three parts. We first define a *twin problem* in Section 3.4.1 and prove the equivalence of first-order stationary points between the original problem and the twin one. Next, we show that Algorithm 1 converges to a critical point of the twin problem via Lemma 3.4.1 and thus also a critical point of the original problem (3.4) by Lemma 3.4.2. The key is to show that the twin problem satisfies (C1), (C2), (C3) and use the fact that $G_\rho(\tilde{\sigma}, \sigma)$ is a KL function.

3.4.1 Twin problem

Since we have obtained $y^k = C\tilde{\sigma}^k, k \geq 1$ in (3.12), Algorithm 1 can also be interpreted as an alternating method for minimization of the function

$$G_\rho(\tilde{\sigma}, \sigma) := \langle C, \tilde{\sigma}\tilde{\sigma}^\top \rangle + \frac{\rho}{2}\|\tilde{\sigma} - \sigma\|_F^2 + \sum_{i=1}^{n} \mathcal{I}_{\|\tilde{\sigma}_i\|=1}(\tilde{\sigma}_i). \tag{3.26}$$

Note that $G_\rho(\tilde{\sigma}^k, \sigma^k) = L_\rho(\tilde{\sigma}^k, \sigma^k, y^k)$ for all $k \geq 1$ when we take $y = C\tilde{\sigma}$ in the augmented Lagrangian $L_\rho(\sigma, \tilde{\sigma}, y)$. It then remains to prove that Algorithm 1 converges to a critical point of (3.26), which is also a critical point of (3.4) due to the following lemma:

Lemma 3.4.2. *The critical point of the problem* (3.26) *is also a critical point of the problem* (3.4).

Proof. A critical point of (3.26) satisfies

$$\begin{aligned}
\partial_{\tilde{\sigma}} G_\rho(\tilde{\sigma}, \sigma) &= 2C\tilde{\sigma} + \rho(\tilde{\sigma} - \sigma) + \partial\left(\sum_{i=1}^{n} \mathcal{I}_{\|\tilde{\sigma}_i\|=1}(\tilde{\sigma}_i)\right), \\
\partial_{\sigma} G_\rho(\tilde{\sigma}, \sigma) &= \rho(\sigma - \tilde{\sigma}).
\end{aligned}$$

When $0 \in \partial_{\tilde{\sigma}} G_\rho(\tilde{\sigma}, \sigma)$ and $\partial_\sigma G_\rho(\tilde{\sigma}, \sigma) = 0$ (a critical point to $G_\rho(\tilde{\sigma}, \sigma)$), it also implies that $\tilde{\sigma}$ is a critical point to the problem (3.4). □

3.4.2 Proof of Theorem 3.4.1

In order to prove Theorem 3.4.1 we will first need the following result showing that the property (C2) is satisfied for $G_\rho(\tilde{\sigma}, \sigma)$:

Lemma 3.4.3. *The sequence* $(\tilde{\sigma}^k, \sigma^k)$ *generated by Algorithm 1 satisfies the property (C2) w.r.t.* $G_\rho(\tilde{\sigma}, \sigma)$*:*

$$\left\| \begin{matrix} \partial_{\tilde{\sigma}} G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) \\ \partial_{\sigma} G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) \end{matrix} \right\|_F \leq \left(2\|C\| + \rho + \frac{\|C\|^2}{\rho}\right) \left\| \begin{matrix} \tilde{\sigma}^{k+1} - \tilde{\sigma}^k \\ \sigma^{k+1} - \sigma^k \end{matrix} \right\|_F, \ \forall k \geq 1. \tag{3.27}$$

Proof. According to Algorithm 1, we have

$$C\sigma^k + y^k + \rho(\tilde{\sigma}^{k+1} - \sigma^k) + v^{k+1} = 0, \tag{3.28}$$

$$C\tilde{\sigma}^{k+1} - y^k + \rho(\sigma^{k+1} - \tilde{\sigma}^{k+1}) = 0, \tag{3.29}$$

due to the first-order optimality condition of (3.7a), (3.7b) where

$$v^{k+1} = \sum_{i=1}^{n} \partial I_{\|\tilde{\sigma}_i\|=1}(\tilde{\sigma}_i^{k+1}).$$

Then, a subgradient of $G_\rho(\tilde{\sigma}, \sigma)$ at $(\tilde{\sigma}^{k+1}, \sigma^{k+1})$ is

$$\begin{aligned}
\partial_{\tilde{\sigma}} G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) =& 2C\tilde{\sigma}^{k+1} + \rho(\tilde{\sigma}^{k+1} - \sigma^{k+1}) + v^{k+1} \\
\overset{(3.28)}{=}& 2C\tilde{\sigma}^{k+1} + \rho(\tilde{\sigma}^{k+1} - \sigma^{k+1}) - [C\sigma^k + y^k + \rho(\tilde{\sigma}^{k+1} - \sigma^k)], \\
\overset{(3.12)}{=}& 2C\tilde{\sigma}^{k+1} - C\sigma^k - C\tilde{\sigma}^k - \rho(\sigma^{k+1} - \sigma^k) \\
=& C(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k) + C(\sigma^{k+1} - \sigma^k) + C(\tilde{\sigma}^{k+1} - \sigma^{k+1}) - \rho(\sigma^{k+1} - \sigma^k) \\
\overset{(3.7c)}{=}& C(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k) + C(\sigma^{k+1} - \sigma^k) + \frac{C}{\rho}(y^{k+1} - y^k) - \rho(\sigma^{k+1} - \sigma^k) \\
\overset{(3.12)}{=}& C(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k) + C(\sigma^{k+1} - \sigma^k) + \frac{C^2}{\rho}(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k) - \rho(\sigma^{k+1} - \sigma^k)
\end{aligned} \tag{3.30}$$

$$\partial_\sigma G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) = \rho(\sigma^{k+1} - \tilde{\sigma}^{k+1}) \overset{(3.7c)}{=} y^k - y^{k+1} \overset{(3.12)}{=} C(\tilde{\sigma}^k - \tilde{\sigma}^{k+1}). \tag{3.31}$$

Therefore,

$$\left\| \begin{matrix} \partial_{\tilde{\sigma}} G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) \\ \partial_\sigma G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) \end{matrix} \right\|_F \leq \left(2\|C\| + \rho + \frac{\|C\|^2}{\rho} \right) \left\| \begin{matrix} \tilde{\sigma}^{k+1} - \tilde{\sigma}^k \\ \sigma^{k+1} - \sigma^k \end{matrix} \right\|_F, \ \forall k \geq 1. \tag{3.32}$$

□

Finally, we give the proof for Theorem 3.4.1.

Proof of Theorem 3.4.1. (C1) is valid for $G_\rho(\tilde{\sigma}, \sigma)$ with $a = \min\{\kappa\|C\|, \rho\}$ since we can rewrite Lemma 3.3.3 as

$$G_\rho(\tilde{\sigma}^k, \sigma^k) - G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) \geq \kappa\|C\| \cdot \|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\|_F^2 + \frac{\rho}{2}\|\sigma^{k+1} - \sigma^k\|_F^2, \tag{3.33}$$

based on the primal-dual connection (3.12). (C2) is validated in Lemma 3.4.3. Also, $\tilde{\sigma}^k \in \mathcal{M}$ and the update of σ^k in Algorithm 1 imply that the sequence $(\tilde{\sigma}^k, \sigma^k)$ is bounded. Therefore, there exists a convergent subsequence $(\tilde{\sigma}^{k_j}, \sigma^{k_j})$ and the continuity of $G_\rho(\tilde{\sigma}, \sigma)$ on $\mathcal{M} \times \mathbb{R}^{n \times r}$ mean that (C3) is satisfied. In addition, $G_\rho(\tilde{\sigma}, \sigma)$ is semi-algebraic and thus a KL function that satisfies the KL property at any point

of $\operatorname{dom} G_\rho(\tilde{\sigma}, \sigma)$[1]. Therefore, Lemma 3.4.1 applies to $G_\rho(\tilde{\sigma}, \sigma)$ and the sequence $(\tilde{\sigma}, \sigma)$ generated by Algorithm 1 converges to a critical point of the problem (3.26) and thus a critical point of (3.4) due to Lemma 3.4.2, with

$$\sum_{k=0}^{+\infty} \left(\|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\| + \|\sigma^{k+1} - \sigma^k\| \right) < \infty,$$

which implies $\sum_{k=0}^{+\infty} \|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\| < \infty$. □

3.5 Exploitation of the negative curvature

So far we have analysed Algorithm 1 in the Euclidean metric, but it is known that SDPs can also be regarded as optimization over manifolds [87, 88]. For a class of SDP problems including diagonally constrained SDPs, any local optimum is within a neighbourhood of the global optimality in manifold optimization [89]. Hence, we can exploit second-order information in Algorithm 1 to guarantee the convergence to a neighbourhood of global optimality. In this section, we analyse the convergence of Algorithm 1 in view of manifold optimization.

3.5.1 Convergence to a first-order stationary point on manifolds

First, we show that the first-order stationary point defined in (3.13) is consistent with the counterpart defined on manifolds, and the previous convergence analysis also implies the convergence to a critical point on manifolds.

Lemma 3.5.1. *Any first-order stationary point (critical point) defined as in* (3.13) *is a first-order stationary point (critical point) on the Cartesian products of n spherical manifolds.*

Proof. Note that the manifold gradient of Problem (3.4) is

$$\operatorname{grad} f(\sigma) = 2(C - \operatorname{Diag}(\operatorname{diag}(C\sigma\sigma^\top)))\sigma, \ \forall \sigma \in \mathcal{M}. \tag{3.34}$$

[1]The semi-algebraic functions and related KL property are detailed in [81, §5].

Any first-order stationary point σ^* defined by (3.13) is equivalent to

$$C\sigma^* = \Lambda\sigma^*, \sigma^* \in \mathcal{M}, \tag{3.35}$$

where $\Lambda = \mathrm{Diag}(\lambda_1, \cdots, \lambda_n), \lambda_i \in \mathbb{R}, \forall i \in [\![n]\!]$, and we obtain

$$\begin{aligned} \mathrm{grad} f(\sigma^*) &= 2(C - \mathrm{Diag}(\mathrm{diag}(C\sigma^*{\sigma^*}^\top))\sigma^* \\ &= 2(\Lambda - \mathrm{Diag}(\mathrm{diag}(\Lambda\sigma^*{\sigma^*}^\top))\sigma^* \\ &= \mathbf{0}, \end{aligned} \tag{3.36}$$

where the last equality is due to the property of $\mathcal{M}$ that $\|\sigma_i\| = 1, \forall i \in [\![n]\!]$. □

3.5.2 Achieving $O(1/r)$ optimality with negative curvature

For nonconvex optimization, a first-order algorithm may stall at a saddle point. We can improve the convergence of our algorithm to second order stationary points by exploiting negative curvature when it is close to a saddle point. First, we define approximate convex points of a function f on manifold $\mathcal{M}$.

Definition 3.5.1 (Approximate convex point)**.** Let f be a twice differentiable function on a Riemannian manifold $\mathcal{M}$. The point $\sigma \in \mathcal{M}$ is an ϵ-approximate convex point of f on $\mathcal{M}$ if

$$\langle u, \mathrm{Hess} f(\sigma)[u] \rangle \geq -\epsilon \langle u, u \rangle, \quad \forall u \in T_\sigma \mathcal{M},$$

where $\mathrm{Hess} f(\sigma)$ denotes the Riemannian Hessian of f at point σ and $\langle \cdot, \cdot \rangle$ is the scalar product on $T_\sigma \mathcal{M}$, which is the tangent space of $\mathcal{M}$ at σ.

We can now extend Algorithm 1 to exploit negative curvature, resulting in Algorithm 2.

Algorithm 2 ADMM-BM2

1: Initialization: set $\sigma^0 = \tilde{\sigma}^0 \in \mathcal{M}, y^0 = C\tilde{\sigma}^0, \epsilon > 0$.

2: **for** $k = 0, 1, 2, \ldots$ **do**

3: $\quad \gamma^k = \sigma^k - \frac{1}{\rho}(y^k + C\sigma^k)$

4: $\quad \tilde{\sigma}^{k+1} \leftarrow$ Normalize-Row(γ^k)

5: $\sigma^{k+1} \leftarrow \tilde{\sigma}^{k+1} + \frac{1}{\rho}(y^k - C\tilde{\sigma}^{k+1})$

6: $y^{k+1} \leftarrow y^k + \rho(\tilde{\sigma}^{k+1} - \sigma^{k+1})$

7: **if** $G_\rho(\tilde{\sigma}^k, \sigma^k) - G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) \geq \Delta$ **then**

8: Continue with $\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^{k+1}$.

9: **else**

10: Find $u^k \in T_{\tilde{\sigma}^k}\mathcal{M}$ such that

$$\lambda_H(\tilde{\sigma}^k) := \langle u^k, \mathrm{Hess} f(\tilde{\sigma}^k)[u^k]\rangle \leq \lambda_{\min}(\mathrm{Hess} f(\tilde{\sigma}^k))/2$$

with $\langle u^k, \mathrm{grad} f(\tilde{\sigma}^k)\rangle \leq 0$ and $\|u^k\|_F = 1$.

11: **if** $\lambda_H(\tilde{\sigma}^k) < -\frac{\epsilon}{2}$ **then**

12: $\tilde{\sigma}_i^{k+1} \leftarrow \tilde{\sigma}_i^k \cos(\|u_i^k\|t) + \frac{u_i^k}{\|u_i^k\|}\sin(\|u_i^k\|t), \forall i \in [\![n]\!]$ with $t = \frac{\epsilon}{15\|C\|_1}$.

13: $\sigma^{k+1} \leftarrow \tilde{\sigma}^{k+1}$

14: $y^{k+1} \leftarrow C\tilde{\sigma}^{k+1}$

15: **else**

16: Return an ϵ-approximate convex point with high probability.

17: **end if**

18: **end if**

19: **end for**

Compared with Algorithm 1, we additionally check the consecutive difference of $G_\rho(\tilde{\sigma}, \sigma)$ at the end of each iteration in Algorithm 2 (line 7). If the decrease is sufficiently large, we progress to the next iteration as Algorithm 1. Otherwise, we exploit the negative curvature at $\tilde{\sigma}^k$ by a power method instead.

Due to Lemma 2 in [33], we have

$$\begin{aligned} f(\tilde{\sigma}^{k+1}) &\leq f(\tilde{\sigma}^k) + t\langle u^k, \mathrm{grad} f(\tilde{\sigma}^k)\rangle + \frac{t^2}{2}\langle u^k, \mathrm{Hess} f(\tilde{\sigma}^k)[u^k]\rangle + \frac{5\|C\|_1}{2}t^3 \\ &\leq f(\tilde{\sigma}^k) + \frac{\lambda_H(\tilde{\sigma}^k)}{2}t^2 + \frac{5\|C\|_1}{2}t^3, \end{aligned} \tag{3.37}$$

given $\langle u^k, \mathrm{grad} f(\tilde{\sigma}^k)\rangle \leq 0$ and

$$\langle u^k, \mathrm{Hess} f(\tilde{\sigma}^k)[u^k]\rangle = \lambda_H(\tilde{\sigma}^k) = \lambda_{\min}(\mathrm{Hess} f(\tilde{\sigma}^k))/2 < 0.$$

From (3.37), we have

$$f(\tilde{\sigma}^k) - f(\tilde{\sigma}^{k+1}) \geq -\frac{\lambda_H(\tilde{\sigma}^k)}{2}t^2 - \frac{5\|C\|_1}{2}t^3 \geq \frac{\epsilon}{4}t^2 - \frac{5\|C\|_1}{2}t^3. \tag{3.38}$$

Setting $t = \frac{\epsilon}{15\|C\|_1}$ yields the last inequality above, i.e

$$f(\tilde{\sigma}^k) - f(\tilde{\sigma}^{k+1}) \geq \frac{\epsilon^3}{2700\,\|C\|_1^2}. \tag{3.39}$$

The analysis then relies on the following lemma:

Lemma 3.5.2 (Theorem 2 in [89]). *For any ϵ-approximate convex point $\sigma \in \mathcal{M}$ of the rank-r non-convex problem* (3.4)*, we have*

$$f(\sigma) \leq SDP(C) - \frac{1}{r-1}(SDP(C) + SDP(-C)) + \frac{n}{2}\epsilon,$$

where $SDP(C)$ is the optimum of (3.3)*.*

Note that, if we define $\hat{X}$ is the optimal solution of $\mathrm{SDP}(-C)$,

$$\mathrm{SDP}(C) \leq \langle C, \hat{X}\rangle = -\mathrm{SDP}(-C),$$

which implies $\mathrm{SDP}(C) + \mathrm{SDP}(-C) \leq 0$. Suppose we denote

$$f^* := SDP(C) - \frac{1}{r-1}(SDP(C) + SDP(-C)),\ g(\sigma) := f(\sigma) - f^*,$$

and assume $g(\tilde{\sigma}^0), \ldots, g(\tilde{\sigma}^{T+1}) > 0$. We can then obtain the following theorem, which is the main result of this section:

Theorem 3.5.1. *Suppose we choose an adaptive step $t = \frac{\epsilon}{15\|C\|_1}$ and the threshold $\Delta = \kappa\epsilon^2$ in Algorithm 2 with constant $\kappa > 0$. Algorithm 2 returns a point $\sigma \in \mathcal{M}$ with*

$$f(\sigma) \leq SDP(C) - \frac{1}{r-1}(SDP(C) + SDP(-C)) + \frac{n}{2}\epsilon, \tag{3.40}$$

within $T = T_1 + T_2$ iterations, where we set $T_1 = \left\lceil \frac{g(\tilde{\sigma}^0)}{\kappa\epsilon^2} \right\rceil$ and $T_2 = \lceil 675\|C\|_1^2 n/\epsilon^2 \rceil$.

Proof of Theorem 3.5.1.

Algorithm 2 has three possible updates for each iteration: we proceed as Algorithm 1 when the decrease of $G_\rho(\tilde{\sigma}^k, \sigma^k)$ is sufficiently large enough (case 1). Otherwise, we try to exploit the negative curvature along the most negative decreasing direction u^k when the smallest eigenvalue is negative (case 2), or return an $\epsilon-$approximate convex point (case 3).

Case 1 (line 7-8): $G_\rho(\tilde{\sigma}^k, \sigma^k) - G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) \geq \Delta$.
Suppose $\{k_i\}_{i=1,2,\ldots,T_1}$ is the set of iteration numbers when

$$G_\rho(\tilde{\sigma}^{k_i}, \sigma^{k_i}) - G_\rho(\tilde{\sigma}^{k_i+1}, \sigma^{k_i+1}) \geq \kappa\epsilon^2,$$

is satisfied.

Summing the decrease for every first-order step, we have

$$\begin{aligned} T_1\kappa\epsilon^2 &\leq \sum_{i=1}^{T_1} \left(G_\rho(\tilde{\sigma}^{k_i}, \sigma^{k_i}) - G_\rho(\tilde{\sigma}^{k_i+1}, \sigma^{k_i+1})\right) \leq G_\rho(\tilde{\sigma}^{k_1}, \sigma^{k_1}) - G_\rho(\tilde{\sigma}^{k_{T_1}+1}, \sigma^{k_{T_1}+1}) \\ &\leq G_\rho(\tilde{\sigma}^{k_1}, \sigma^{k_1}) - f^* - (G_\rho(\tilde{\sigma}^{k_{T_1}+1}, \sigma^{k_{T_1}+1}) - f^*) \leq G_\rho(\tilde{\sigma}^{k_1}, \sigma^{k_1}) - f^* - g(\tilde{\sigma}^{k_{T_1}+1}) \\ &\leq G_\rho(\tilde{\sigma}^0, \sigma^0) - f^* = g(\tilde{\sigma}^0), \end{aligned} \tag{3.41}$$

i.e. $T_1 \leq \frac{g(\tilde{\sigma}^0)}{\kappa\epsilon^2}$.

Case 2 (line 11-14): $G_\rho(\tilde{\sigma}^k, \sigma^k) - G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) < \Delta$ and $\lambda_H(\tilde{\sigma}^k) < -\frac{\epsilon}{2}$.
In this case, we have $\lambda_{\min}(\tilde{\sigma}^k) < \lambda_H(\tilde{\sigma}^k) < -\frac{\epsilon}{2} < 0$ and (3.39) holds. We then rewrite Lemma 3.5.2 as

$$g(\tilde{\sigma}^k) \leq \frac{n}{2}\epsilon,$$

and substituting it into (3.39) yields

$$g(\tilde{\sigma}^k) - g(\tilde{\sigma}^{k+1}) \geq \frac{2g(\tilde{\sigma}^k)^3}{675\|C\|_1^2 n^3}. \tag{3.42}$$

Then

$$\begin{aligned} \frac{1}{g(\tilde{\sigma}^{k+1})^2} - \frac{1}{g(\tilde{\sigma}^k)^2} &\geq \frac{(g(\tilde{\sigma}^k) - g(\tilde{\sigma}^{k+1}))(g(\tilde{\sigma}^k) + g(\tilde{\sigma}^{k+1}))}{g(\tilde{\sigma}^k)^2 g(\tilde{\sigma}^{k+1})^2} \\ &\geq \frac{2}{675\|C\|_1^2 n^3}\left(\frac{g(\tilde{\sigma}^k)}{g(\tilde{\sigma}^{k+1})} + \frac{g(\tilde{\sigma}^k)^2}{g(\tilde{\sigma}^{k+1})^2}\right) \\ &\geq \frac{4}{675\|C\|_1^2 n^3}. \end{aligned} \tag{3.43}$$

Case 3 (line 16): $G_\rho(\tilde{\sigma}^k, \sigma^k) - G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) < \Delta$ and $\lambda_H(\tilde{\sigma}^k) > -\frac{\epsilon}{2}$.
We can obtain

$$\lambda_{\min}(\text{Hess} f(\tilde{\sigma}^k)) \geq 2 * \lambda_H(\tilde{\sigma}^k) > -\epsilon,$$

which means $\tilde{\sigma}^k$ is already an ϵ-approximate convex point and (3.40) is satisfied due to Lemma 3.3.2.

Suppose $\{k_j\}_{j=1,2,...,T_2}$ is the subsequence of $\{1, \ldots, T+1\}$ such that the negative curvature is taken but not at an ϵ-approximate convex point. Note that $f(\tilde{\sigma}^{k_j+1}) = G_\rho(\tilde{\sigma}^{k_j+1}, \sigma^{k_j+1}), \forall j$ and $G_\rho(\tilde{\sigma}^{k_j+1}, \sigma^{k_j+1}) \geq G_\rho(\tilde{\sigma}^{k_{j+1}}, \sigma^{k_{j+1}}) \geq f(\tilde{\sigma}^{k_{j+1}})$ due to the non-decreasing property of $G_\rho(\tilde{\sigma}^k, \sigma^k)$, we can obtain $g(\tilde{\sigma}^{k_j+1}) \geq g(\tilde{\sigma}^{k_{j+1}})$. Therefore, summing up (3.43) yields

$$\frac{4T_2}{675\|C\|_1^2 n^3} \leq \sum_{j=1}^{T_2} \left(\frac{1}{g(\tilde{\sigma}^{k_j+1})^2} - \frac{1}{g(\tilde{\sigma}^{k_j})^2} \right) \leq \frac{1}{g(\tilde{\sigma}^{k_T+1})^2} - \frac{1}{g(\tilde{\sigma}^{k_1})^2} \leq \frac{1}{g(\tilde{\sigma}^{T+1})^2}, \tag{3.44}$$

which guarantees $g(\tilde{\sigma}^{T+1}) \leq n\epsilon/2$ as long as we set $T_2 = \lceil 675\|C\|_1^2 n/\epsilon^2 \rceil$. □

3.6 Extension to the product of Stiefel manifolds

The parallel implementation of Algorithm 1 can also be generalized to solve block-diagonally constrained SDPs (3.45),

$$\begin{aligned} \min \quad & \langle C, X \rangle \\ \text{s.t} \quad & X_{ii} = I_d, \text{for } i \in [\![q]\!] \\ & X \in \mathbb{S}_+^n, \end{aligned} \tag{3.45}$$

with $n = qd$. The rank-constrained counterpart of (3.45) is

$$\begin{aligned} \min \quad & f(\sigma) := \langle C, \sigma\sigma^\top \rangle \\ \text{s.t.} \quad & \sigma_i^\top \sigma_i = I_d, \text{for } i \in [\![q]\!], \end{aligned} \tag{3.46}$$

where $\sigma := [\sigma_1, \sigma_2, ..., \sigma_q]^\top \in \mathbb{R}^{qd \times r}$ with $r \geq d$ and $\sigma_i \in \mathbb{R}^{r \times d}$ is the i-th block of σ, and the manifold $\mathcal{M}$ is generalized to the product of Stiefel manifolds $\mathcal{M} = \{[\sigma_1, \sigma_2, \ldots, \sigma_q]^\top \in \mathbb{R}^{nd \times r} \mid \ \sigma_i^\top \sigma_i = I_d, \ i \in [q]\} := \mathcal{M}_1 \times \cdots \times \mathcal{M}_q$. To ensure the convergence of Algorithm 1, we add a proximal regularization to the update of $\tilde{\sigma}$.

When we add a proximal regularization to step (3.7a), we obtain a new prox-ADMM algorithm as follows,

$$\tilde{\sigma}^{k+1} = \underset{\tilde{\sigma}\in\mathcal{M}}{\operatorname{argmin}}\ L_\rho(\tilde{\sigma}, \sigma^k, y^k) + \frac{\mu}{2}\|\tilde{\sigma} - \tilde{\sigma}^k\|^2, \tag{3.47a}$$

$$\sigma^{k+1} = \underset{\sigma}{\operatorname{argmin}}\ L_\rho(\tilde{\sigma}^{k+1}, \sigma, y^k), \tag{3.47b}$$

$$y^{k+1} = y^k + \rho(\tilde{\sigma}^{k+1} - \sigma^{k+1}). \tag{3.47c}$$

The update of $\tilde{\sigma}$ can be written as $\tilde{\sigma}^{k+1} \leftarrow \mathrm{Proj}_{\mathcal{M}}(\gamma^k)$ if we define

$$\gamma^k := \frac{\mu}{\rho+\mu}\tilde{\sigma}^k + \frac{\rho}{\rho+\mu}\sigma^k - \frac{1}{\rho+\mu}(y^k + C\sigma^k),$$

which is equivalent to

$$\tilde{\sigma}_i^{k+1} \leftarrow \mathrm{Proj}_{\mathcal{M}_i}(\gamma_i^k),\ \forall i \in [\![n]\!],$$

with $\gamma^k = [\gamma_1^k, \ldots, \gamma_n^k]^\top, \gamma_i^k \in \mathbb{R}^{r\times d}$. The projection has an analytic solution by computing the SVD factorization of each γ_i^k, which we show in the following result:

Lemma 3.6.1 (Theorem 1 [74])**.** *The constrained quadratic problem*

$$P^* = \underset{P\in\mathbb{R}^{r\times d}}{\operatorname{argmin}} \frac{1}{2}\|P - X\|_F^2, \quad s.t.\ P^\top P = I_d,$$

which is the projection of X to the Stiefel manifold $P^\top P = I_d$, has closed-form solution $P^* = UI_{r\times d}V^\top$, where $U \in \mathbb{R}^{r\times r}, V \in \mathbb{R}^{d\times d}$ are two orthogonal matrices and $D \in \mathbb{R}^{r\times d}$ is a diagonal matrix satisfying the SVD factorization $X = UDV^\top$.

To summarize, we propose a proximal ADMM algorithm for the products of Stiefel manifolds shown in Algorithm 3. Compared with Algorithm 1, the only difference is an additional regularization for the update of $\tilde{\sigma}^{k+1}$ to guarantee the convergence of ADMM-BM theoretically. Meanwhile, the computation of Algorithm 3 can also be paralleled on GPU.

Algorithm 3 Proximal ADMM Burer-Monteiro algorithm (Prox-ADMM-BM)

1: Initialization: set $\sigma^0 = \tilde{\sigma}^0 \in \mathcal{M}, y^0 = C\tilde{\sigma}^0$.
2: **while** *termination criteria not satisfied* **do**
3: $\quad\gamma^k \leftarrow \frac{\mu}{\rho+\mu}\tilde{\sigma}^k + \frac{\rho}{\rho+\mu}\sigma^k - \frac{1}{\rho+\mu}(y^k + C\sigma^k)$
4: $\quad\gamma_i^k \leftarrow \mathrm{Proj}_{\mathcal{M}_i}(\gamma_i^k),\ \forall i \in [q]$
5: $\quad\sigma^{k+1} \leftarrow \tilde{\sigma}^{k+1} + \frac{1}{\rho}(y^k - C\tilde{\sigma}^{k+1})$
6: $\quad y^{k+1} \leftarrow y^k + \rho(\tilde{\sigma}^{k+1} - \sigma^{k+1})$
7: **end while**

Similar to Theorem 3.4.1, we can prove that Algorithm 3 converges to a critical point of f in (3.46).

Theorem 3.6.1. *Suppose $\forall i \in [q]$, $\|\gamma_i^k\|_F$ is nonzero for any iteration k. If we set $\rho, \mu \geq 0$ properly such that $\mu - \|C\|^2/\rho > 0$, then the sequence $(\tilde{\sigma}^k)_{k\in\mathbb{N}}$ generated by Algorithm 3 satisfies Lemma 3.4.1 and converges to a critical point $\bar{\sigma}$ of f in* (3.46). *Moreover, the sequence $(\tilde{\sigma}^k)_{k\in\mathbb{N}}$ has a finite length, i.e. $\sum_{k=0}^{+\infty} \|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\| < \infty$.*

Proof of Theorem 3.6.1. Similar to (3.18), we have the decrease

$$L_\rho(\tilde{\sigma}^k, \sigma^k, y^k) \geq L_\rho(\tilde{\sigma}^{k+1}, \sigma^k, y^k) + \frac{\mu}{2}\|\tilde{\sigma}^k - \tilde{\sigma}^{k+1}\|_F^2,$$

and the counterpart of Lemma 3.3.3 is

$$L_\rho(\tilde{\sigma}^k, \sigma^k, y^k) - L_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}, y^{k+1}) \geq \left(\mu - \frac{\|C\|^2}{\rho}\right)\|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\|_F^2 + \frac{\rho}{2}\|\sigma^{k+1} - \sigma^k\|_F^2,$$

which implies (C1) is valid for $G_\rho(\tilde{\sigma}, \sigma)$ since

$$\begin{aligned} G_\rho(\tilde{\sigma}^k, \sigma^k) - G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) &\geq \left(\mu - \frac{\|C\|^2}{\rho}\right) \cdot \|\tilde{\sigma}^{k+1} - \tilde{\sigma}^k\|_F^2 + \frac{\rho}{2}\|\sigma^{k+1} - \sigma^k\|_F^2, \\ &\geq \min\left\{\mu - \frac{\|C\|^2}{\rho}, \frac{\rho}{2}\right\} \cdot \left\|\begin{matrix}\tilde{\sigma}^{k+1} - \tilde{\sigma}^k \\ \sigma^{k+1} - \sigma^k\end{matrix}\right\|_F^2. \end{aligned} \tag{3.48}$$

In addition, the condition (3.28) becomes

$$C\sigma^k + y^k + \rho(\tilde{\sigma}^{k+1} - \sigma^k) + \mu(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k) + v^{k+1} = 0.$$

Then a subgradient of $G_\rho(\tilde{\sigma}, \sigma)$ at $(\tilde{\sigma}^{k+1}, \sigma^{k+1})$ is

$$\begin{aligned} &\partial_{\tilde{\sigma}} G_\rho(\tilde{\sigma}^{k+1}, \sigma^{k+1}) \\ &= 2C\tilde{\sigma}^{k+1} + \rho(\tilde{\sigma}^{k+1} - \sigma^{k+1}) + v^{k+1}, \\ &= 2C\tilde{\sigma}^{k+1} + \rho(\tilde{\sigma}^{k+1} - \sigma^{k+1}) - [C\sigma^k + y^k + \rho(\tilde{\sigma}^{k+1} - \sigma^k) + \mu(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k)], \\ &= 2C\tilde{\sigma}^{k+1} - C\sigma^k - C\tilde{\sigma}^k - \rho(\sigma^{k+1} - \sigma^k) - \mu(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k) \\ &= C(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k) + C(\sigma^{k+1} - \sigma^k) + C(\tilde{\sigma}^{k+1} - \sigma^{k+1}) - \rho(\sigma^{k+1} - \sigma^k) - \mu(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k) \\ &\overset{(3.7c)}{=} C(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k) + C(\sigma^{k+1} - \sigma^k) + \frac{C}{\rho}(y^{k+1} - y^k) - \rho(\sigma^{k+1} - \sigma^k) - \mu(\tilde{\sigma}^{k+1} - \tilde{\sigma}^k) \end{aligned}$$

$$\overset{(3.12)}{=}(C-\mu I)(\tilde{\sigma}^{k+1}-\tilde{\sigma}^{k})+C(\sigma^{k+1}-\sigma^{k})+\frac{C^2}{\rho}(\tilde{\sigma}^{k+1}-\tilde{\sigma}^{k})-\rho(\sigma^{k+1}-\sigma^{k}),$$

and

$$\partial_\sigma G_\rho(\tilde{\sigma}^{k+1},\sigma^{k+1}) = \rho(\sigma^{k+1}-\tilde{\sigma}^{k+1}) \overset{(3.7c)}{=} y^k - y^{k+1} \overset{(3.12)}{=} C(\tilde{\sigma}^k-\tilde{\sigma}^{k+1}),$$

which imply that

$$\left\|\begin{matrix}\partial_{\tilde{\sigma}} G_\rho(\tilde{\sigma}^{k+1},\sigma^{k+1})\\ \partial_\sigma G_\rho(\tilde{\sigma}^{k+1},\sigma^{k+1})\end{matrix}\right\|_F \le \left(2\|C\|+\max\{\rho,\mu\}+\frac{\|C\|^2}{\rho}\right)\left\|\begin{matrix}\tilde{\sigma}^{k+1}-\tilde{\sigma}^{k}\\ \sigma^{k+1}-\sigma^{k}\end{matrix}\right\|_F,\ \forall k,$$

and hence (C2) is satisfied. $L_\rho(\tilde{\sigma}^k,\sigma^k,y^k)=G_\rho(\tilde{\sigma}^k,\sigma^k)$ is lower-bounded due to the compact constraint over $\tilde{\sigma}^k$. In combination with (3.48), we can prove that $G_\rho(\tilde{\sigma}^k,\sigma^k)$ converges to a limit point, which implies

$$\left\|\begin{matrix}\tilde{\sigma}^{k+1}-\tilde{\sigma}^{k}\\ \sigma^{k+1}-\sigma^{k}\end{matrix}\right\|_F \to 0,$$

and then

$$\|\tilde{\sigma}^{k+1}-\sigma^{k+1}\| = \frac{1}{\rho}\|y^{k+1}-y^k\| \overset{(3.12)}{\le} \|C\|\cdot\|\tilde{\sigma}^{k+1}-\tilde{\sigma}^k\| \to 0.$$

The compactness of $\mathcal{M}$ implies that there is a subsequence of $\tilde{\sigma}^k$ that converges to a first-order stationary point, i.e. (C3) is valid. We hence prove Theorem 3.6.1 via Lemma 3.4.1. □

3.7 Experimental results

In this section we describe our computational results for Algorithm 1 and Algorithm 3 (named as ADMM-BM) with both CPU and GPU versions for them. All experiments are run on a laptop with Intel i7-8750H CPU @ 2.20GHz and a NVIDIA GeForce GTX1070 with Max-Q Design, with all code written in Julia. Foundamental operations like matrix additions and multiplications rely on NVIDIA CUDA but we need to write the kernel functions for the projection onto manifolds. For comparison, we also present computational results from the Riemannian gradient (RGD) method and the Riemannian trust-region (RTR) method from Manopt.jl [90]. The benchmarking metric is the relative optimality gap defined as:

$$\text{relative optimality gap} = \left|\frac{\langle\tilde{\sigma}^k, C\tilde{\sigma}^k\rangle - \langle\sigma^*, C\sigma^*\rangle}{\langle\sigma^*, C\sigma^*\rangle}\right|, \tag{3.49}$$

where $\tilde{\sigma}^k$ is generated at the k-th iteration by Algorithm 1 or Algorithm 3 that is feasible for (3.4) or (3.46), and σ^* is the optimal solution of $X^* = \sigma^* \sigma^{*\top}$ obtained via Mosek [40].

3.7.1 Max-cut

For Algorithm 1, we set $r = \left\lceil \sqrt{2n} \right\rceil$ and ρ to $\|C\|$ and test it on the dataset Gset[2], which contains a collection of max-cut problems of the form (3.3). An initial condition $\tilde{\sigma}^0 \in \mathbb{R}^{n \times r}$ is generated uniformly at random on $[0, 1]$ and then normalized row-wise. We also set $\sigma^0 = \tilde{\sigma}^0$ and $y^0 = C\tilde{\sigma}^0$ for ADMM-BM and $\tilde{\sigma}^0$ as the initial point for RGD and RTR. We test problems with sizes varying from $n = 800$ to $n = 10000$. The results are shown in Figure 3.1, where the x-axis denotes the computation time and the y-axis denotes the relative optimality defined in (3.49). ADMM-BM always performs better than RGD and RTR at moderate accuracy 10^{-4}. The computation time is within several seconds even for problem sizes up to $n = 10000$. While the GPU implementation of ADMM-BM is slower than the CPU version for small and medium size problems, it exhibits better performance when the dimension n becomes larger.

3.7.2 SO(3) synchronization

The SO(3) synchronization problem [89] is a typical problem of the form (3.45) where we choose $d = 3$. In order to test Algorithm 3, each entry of $\tilde{\sigma}^0$ is generated uniformly at random on $[0, 1]$ and then projected back to the product of Stiefel manifolds as in Lemma 3.6.1. We initialize $\sigma^0 = \tilde{\sigma}^0, y^0 = C\tilde{\sigma}^0$ in Prox-ADMM-BM and $\tilde{\sigma}^0$ as the start point for RTG and RTR. In addition, penalty parameters are set to $\rho = \mu = \|C\|$ in Prox-ADMM-BM. We change the dimension from $n = 300$ to $n = 15000$ and also choose two sparsity pattern, 0.02 and 0.002. As shown in Figure 3.2, the GPU-based Prox-ADMM-BM always performs better than RTG and RTR up to accuracy level 10^{-4}, and it is more suitable for large-scale problems compared to the multi-threaded CPU variant. In addition, we find that Prox-ADMM-BM

[2] `https://www.cise.ufl.edu/research/sparse/matrices/Gset/`

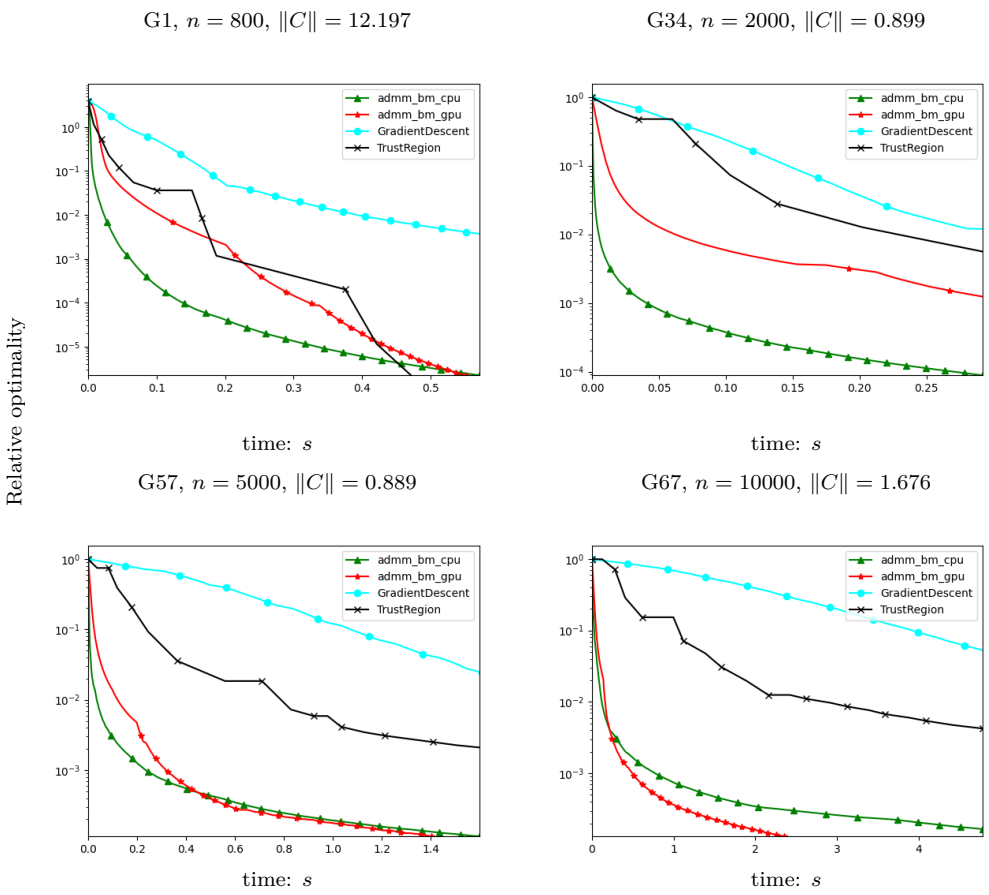


Figure 3.1: Tests of Algorithm 1 on max-cut problems.

converges faster when the cost matrix C is more sparse, which is reasonable since most of computational time is spent on matrix products when we update $\tilde{\sigma}$ and σ. This also applies to ADMM-BM.

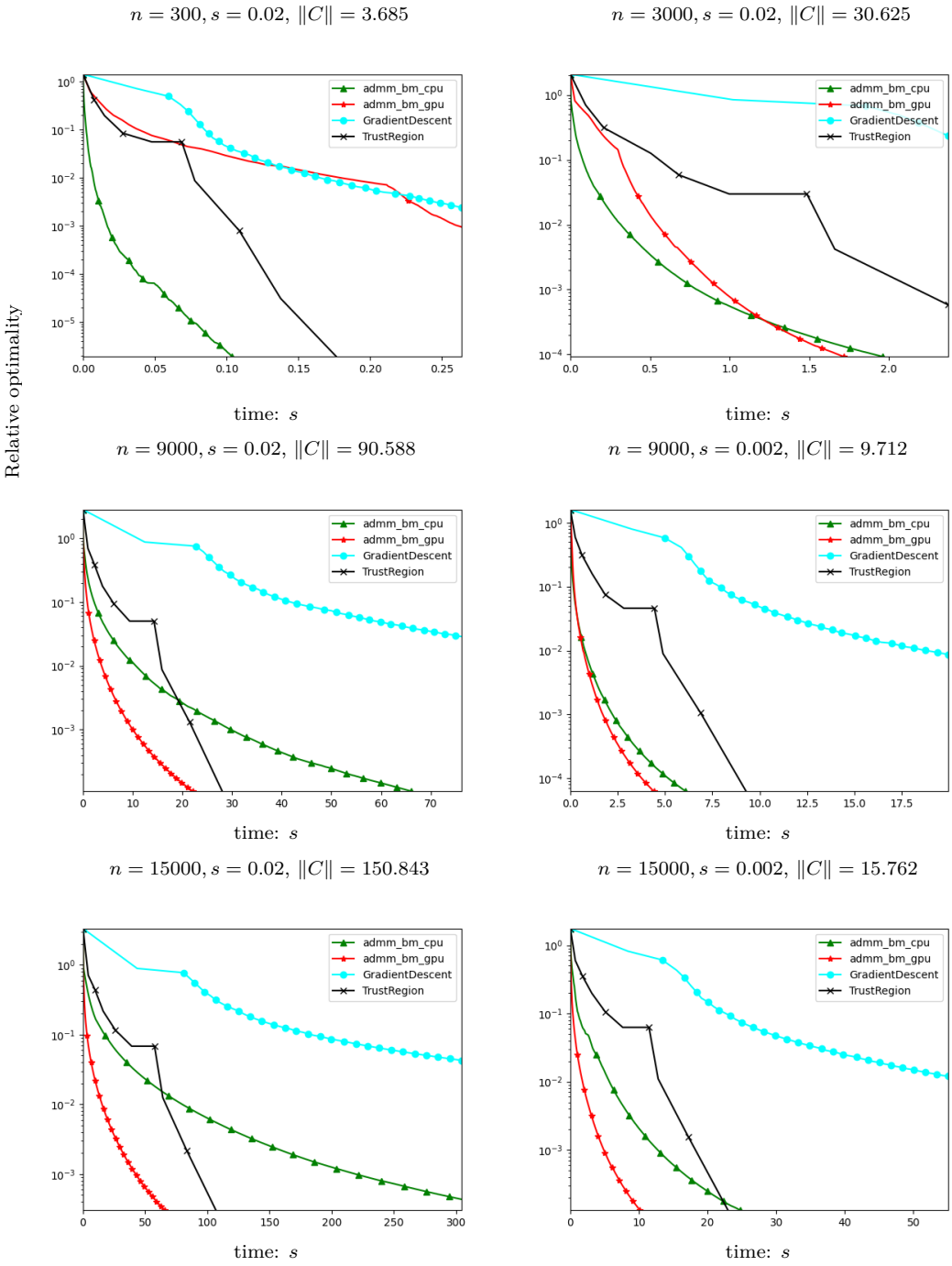


Figure 3.2: Tests of Algorithm 3 on SO(3) synchronization problems.

4

Clarabel: An interior point solver for conic programs

Contents

4.1 Introduction

Primal-dual interior point methods for conic optimization Conic constraints are the generalization of linear inequality constraint and are powerful tools for mathematical modelling. Traditional research focused on self-scaled [13] cones, such as nonnegative cone, second-order cone and positive semidefinite cone, which can be solved efficiently by primal-dual infeasible interior point methods with Nesterov-Todd (NT) scaling [13, 14]. Nonsymmetric cones that are not self-scaled have become a topic of more recent interest for interior point methods [15, 91], and an extension of the homogeneous self-dual embedding for nonsymmetric cones was proposed in [48]. Though it is attractive to formulate problems into some exotic nonsymmetric cones, like the relative entropy cone and the generalized power cone, which can be solved more efficiently by exploiting their special structures [17, 34, 35], the exponential cone and the power cone are the two most studied and are supported in the commercial solver Mosek [92], along with the symmetric second-order and positive semidefinite cones. Compared to symmetric cones, the NT scaling doesn't apply to nonsymmetric cones and different scaling strategies are proposed instead based on only primal or dual iterates [35, 48, 93] or both primal and dual iterates [34, 92].

Monotone Complementarity Problem: The homogeneous self-dual embedding was initially developed in the 1990s [38] for the primal-dual infeasible interior point methods. It is now the foundation of many interior point solvers [40, 41, 45] due to its convenience of infeasibility detection. The embedding was then developed for the linear complementarity problem (LCP) [94], and Anderson and Ye generalized the homogeneous embedding to the monotone complementarity problem (MCP) [95]. Later, Yoshise [96, 97] extended the homogeneous embedding for MCP over symmetric cones and Meszaros studied the practical performance of the homogeneous embedding for convex quadratically constrained quadratic programming (QCQP) problems [98]. The homogeneous self-dual embedding was also exploited for operator-splitting methods and implemented in the SCS solver [18].

Recently, [99] proposed a maximal monotone operator specialized for the LCP based on the monotone operator in [95], which can deal with the quadratic objective functions directly for operator splitting methods with homogeneous embedding.

Similar to [99], we regard the convex conic problem of quadratic objective (QCP) as a LCP that can rely on the homogeneous embedding for infeasible detection. We develop an interior point solver Clarabel [46] that can solve the problem

$$\begin{aligned} \min_{x,s} \quad & \tfrac{1}{2}x^\top Px + q^\top x \\ \text{subject to:} \quad & Ax + s = b, \\ & s \in \mathcal{K}. \end{aligned} \tag{$\mathcal{P}$}$$

Decision variables are $x \in \mathbb{R}^n$ and $s \in \mathbb{R}^m$, and problem parameters are $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $q \in \mathbb{R}^n$ and $P \in \mathbb{R}^{n\times n}$. We assume that P is symmetric and positive semidefinite (possibly zero) and that the set $\mathcal{K}$ is a closed and convex cone. We will denote the optimal value of this problem as p^* and an optimizer (when it exists) as (x^*, s^*). Numerical solution for convex problems in the form $(\mathcal{P})$ also underpins many nonconvex optimization methods, including those based on sequential quadratic programming [100, Ch. 18][101] and branch-and-bound methods within mixed-integer methods [102, 103].

Optimality and infeasibility of QCP As shown in Section 2.2.1, the problem dual to $\mathcal{P}$ is

$$\begin{aligned} \max_{x,z} \quad & -\tfrac{1}{2}x^\top Px - b^\top z \\ \text{subject to:} \quad & Px + A^\top z = -q, \\ & z \in \mathcal{K}^*, \end{aligned} \tag{$\mathcal{D}$}$$

where $\mathcal{K}^*$ is the dual cone of $\mathcal{K}$. We will denote its optimal value as d^* and its optimal solution (when it exists) as (x^*, z^*).

The standard KKT conditions for problem $(\mathcal{P})$ are

$$\begin{aligned} Ax + s &= b, \\ Px + A^\top z &= -q, \\ \langle s, z\rangle &= 0, \\ (s, z) &\in \mathcal{K} \times \mathcal{K}^*. \end{aligned} \tag{4.1}$$

We define the duality gap γ as

$$\begin{aligned}\gamma &:= (\frac{1}{2}x^\top Px + q^\top x) - (-\frac{1}{2}x^\top Px - b^\top z) \\ &= x^\top Px + q^\top x + b^\top z = \langle s, z\rangle\,,\end{aligned}$$

where the final equality follows from substitution in the KKT conditions (4.1). The duality gap is nonnegative for any feasible point since (s, z) are constrained to the dual pair of cones $(\mathcal{K}, \mathcal{K}^*)$, and zero at an optimal point (x^*, s^*, z^*) when strong duality holds.

This problem is of course infeasible whenever either $(\mathcal{P})$ or $(\mathcal{D})$ is infeasible or the pair is not strongly dual. As shown in Section 2.2.3, the KKT conditions are equivalent to an LCP problem,

$$\mathbb{R}^n \times \mathcal{K}^* \ni \begin{bmatrix} x \\ z \end{bmatrix} \perp \begin{bmatrix} P & A^\top \\ -A & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} q \\ b \end{bmatrix} \in \{0\}^n \times \mathcal{K}.$$

When the LCP is strongly infeasible, we can certify that either $(\mathcal{P})$ or $(\mathcal{D})$ is *strongly infeasible* in [99, Section 4.2] if (x, z) satisfies one of the following sets of infeasibility certificates for $(\mathcal{P})$ and $(\mathcal{D})$:

$$\begin{aligned}\mathbb{P} &:= \left\{ z \;\middle|\; A^\top z = 0, \quad z \in \mathcal{K}^*,\, \langle b, z\rangle < 0 \right\}, \\ \mathbb{D} &:= \left\{ x \;\middle|\; Px = 0,\, -Ax \in \mathcal{K},\; \langle q, x\rangle < 0 \right\}.\end{aligned} \tag{4.2}$$

The problem $(\mathcal{P})$ is strongly infeasible if and only if there exists some $\bar{z} \in \mathbb{P}$, while the problem $(\mathcal{D})$ is strongly infeasible if and only if there exists some $\bar{x} \in \mathbb{D}$. The same strong infeasibility conditions have appeared also in [19, 104] in the context of conic optimization with quadratic objectives.

4.2 Homogeneous embedding for conic problems of quadratic objective

The homogeneous self-dual embedding (HSDE) [38] is a popular technique that unifies optimality and infeasibility check of a convex problem. However, it only allows a linear cost in the objective and we need to eliminate the quadratic term in the objective function, replacing it with an epigraphical upper bound and an

additional second-order cone constraint in the objective. This amounts to rewriting $(\mathcal{P})$ as

$$\begin{aligned} \min_{x,s,w} \quad & w + q^\top x \\ \text{subject to:} \quad & Ax + s = b \\ & \left\| P^{\frac{1}{2}} x \right\|_2 \leq 2w \\ & s \in \mathcal{K},\ w \geq 0, \end{aligned} \tag{4.3}$$

which is a conic optimization problem with a purely linear cost and an additional second-order cone constraint. There are two drawbacks related to the transformation[1]:

1. We need to compute the matrix $P^{\frac{1}{2}}$ and use an additional second-order cone constraint that will increase the factorization time in an IPM;

2. The second-order cone programming is observed to be less numerically stable compared to the quadratic programming with only linear constraints in practice.

Recently, [99] treated the conic optimization problem of quadratic cost as a LCP and uses the homogeneous embedding that generalizes HSDE to problems with quadratic cost. It can tackle the infeasibility of (4.4) directly without the second-order cone transformation (4.3). The idea is mainly motivated by [95], which regarded the convex nonlinear optimization problem as an MCP and introduced the homogeneous embedding for MCP that can detect infeasibility. In Clarabel, we rely on this homogeneous embedding for infeasibility detection, which generalizes the ECOS solver [45] that can only solve conic problems with linear cost due to the use of HSDE.

[1]See Remark 4.4.1 for details.

4.2.1 Homogeneous embedding

Taking problems $(\mathcal{P})$ and $(\mathcal{D})$ together with an objective minimizing the duality gap, we can rewrite our problem in the form

$$\begin{aligned} \min_{(x,s,z)} \quad & \langle s, z \rangle \\ \text{subject to:} \quad & x^\top P x + q^\top x + b^\top z = 0 \\ & Px + A^\top z + q = 0 \\ & Ax + s - b = 0 \\ & (s, z) \in \mathcal{K} \times \mathcal{K}^*. \end{aligned} \tag{4.4}$$

The homogeneous embedding contains a nonnegative scalar τ that applies a change of variables $x \to x/\tau$, $z \to z/\tau$ and $s \to s/\tau$. Additionally, it introduces a slack variable $\kappa \in \mathbb{R}_+$ and we can rewrite the feasibility problem (4.4) as

$$\begin{aligned} \min_{(x,s,z,\tau,\kappa)} \quad & \langle s, z \rangle + \tau\kappa \\ \text{subject to:} \quad & \frac{1}{\tau} x^\top P x + q^\top x + b^\top z = -\kappa \\ & Px + A^\top z + q\tau = 0 \\ & Ax + s - b\tau = 0 \\ & (s, z, \tau, \kappa) \in \mathcal{K} \times \mathcal{K}^* \times \mathbb{R}_+ \times \mathbb{R}_+ \end{aligned} \tag{$\mathcal{H}$}$$

The problem $(\mathcal{H})$ amounts to the standard HSDE when $P = 0$.

Note that we keep the quadratic term in the objective and $(\mathcal{H})$ remains homogeneous but is no longer self-dual, in addition to featuring the seemingly awkward $\frac{1}{\tau} x^\top P x$ term in the first equality. Our approach will nevertheless amount to a direct solution of $(\mathcal{H})$ using a primal-dual type interior point method, and will show that significant performance improvements are possible with this approach in many cases.

4.2.2 Optimality and infeasibility detection

Before proceeding to the details of our algorithm, we first address briefly the existence and interpretation of solutions to $(\mathcal{H})$. As in the case of the HSDE, an advantage of the reformulation is that solutions will produce an optimal solution to

original problems $(\mathcal{P})$ and $(\mathcal{D})$, or (asymptotically) a certificate of either primal or dual infeasibility in (4.2).

Solving the problem $(\mathcal{H})$ amounts to finding a root of the nonlinear system of equations

$$G(x,z,s,\tau,\kappa) := \begin{bmatrix} 0 \\ s \\ \kappa \end{bmatrix} - \begin{bmatrix} P & A^\top & q \\ -A & 0 & b \\ -q^\top & -b^\top & 0 \end{bmatrix} \begin{bmatrix} x \\ z \\ \tau \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\tau}x^\top P x \end{bmatrix} = 0$$

subject to the conic constraint

$$(x,z,s,\tau,\kappa) \in \mathcal{D} := (\mathbb{R}^n \times \mathcal{K} \times \mathcal{K}^* \times \mathbb{R}_+ \times \mathbb{R}_+).$$

We will occasionally use the more compact notation $v := (x,z,s,\tau,\kappa)$ for convenience, so that $(\mathcal{H})$ is equivalent to

$$\begin{aligned} \min_{v} \quad & \langle s, z \rangle + \tau\kappa \\ \text{subject to:} \quad & G(v) = 0, \quad v \in \mathcal{D}. \end{aligned} \tag{4.5}$$

This problem is closely related to the homogeneous monotone complementarity problem (HMCP) of [95], but with constraints defined over the generalized cone $\mathcal{D}$ and G restricted to a specific monotone mapping. As in [95], we say that the problem $(\mathcal{H})$ (equivalently (4.5)) is *asymptotically feasible* if there exists a bounded sequence $\{v^k\} \subset \mathcal{D}$, $k = 1, 2, \ldots$, such that

$$\lim_{k\to\infty} G(v^k) = 0,$$

and call any limit point $\hat{v} := (\hat{x}, \hat{z}, \hat{s}, \hat{\tau}, \hat{\kappa})$ of such a sequence an *asymptotically feasible point*. We will call any such point with $\langle \hat{s}, \hat{z} \rangle + \hat{\tau}\hat{\kappa} = 0$ an *(asympototically) complementary* or optimal solution.

We can now develop some basic results about solutions to the problem $(\mathcal{H})$ in the spirit of [95, Theorem. 1]:

Lemma 4.2.1. *The problem $(\mathcal{H})$ is (asymptotically) feasible and every (asymptotically) feasible point is (asymptotically) complementary.*

Proof. We can construct an asympotically feasible sequence by defining $x^k = (\frac{1}{2})^k \mathbf{1}\frac{1}{\sqrt{n}}$, $s^k = (\frac{1}{2})^k e_{\mathcal{K}}$, $z^k = (\frac{1}{2})^k e_{\mathcal{K}^*}$, $\tau^k = (\frac{1}{2})^k$ and $\kappa^k = (\frac{1}{2})^k$, where $(e_{\mathcal{K}}, e_{\mathcal{K}^*})$ is any vector pair in $\mathcal{K} \times \mathcal{K}^*$. It is then straightforward to show that $G(v^k) \to 0$, noting in particular that the sequence

$$\frac{1}{\tau^k}(x^k)^\top P x^k \leq (\tfrac{1}{2})^k \bar{\sigma}(P) \to 0$$

where $\bar{\sigma}(P)$ is the maximum singular value of P, and is lower bounded by zero since P is positive semidefinite. To show the second part, observe that

$$\begin{bmatrix} x \\ z \\ \tau \end{bmatrix}^\top G(v) = \langle s, z \rangle + \tau\kappa$$

for any $v \in \mathcal{D}$, so $\left\langle s^k, z^k \right\rangle + \tau^k\kappa^k \to 0$ since $G(v^k) \to 0$ and the sequence $(x^k, s^k, z^k, \tau^k, \kappa^k)$ is bounded. □

Lemma 4.2.2. *Suppose that $v^* := (x^*, z^*, s^*, \tau^*, \kappa^*)$ is an asymptotically complementary solution to $(\mathcal{H})$. Then :*

1. *If $\tau^* > 0$ then $(x^*/\tau^*, s^*/\tau^*)$ is an optimal solution to $(\mathcal{P})$ and $(x^*/\tau^*, z^*/\tau^*)$ is an optimal solution to $(\mathcal{D})$.*

2. *If $\kappa^* > 0$ then at least one of the following holds:*

   - *$(\mathcal{P})$ is infeasible and $z^* \in \mathbb{P}$.*
   - *$(\mathcal{D})$ is infeasible and $x^* \in \mathbb{D}$.*

Proof. For (i), note that $\kappa^* \to 0$ since the solution is assumed asympototically complementary. Then any point satisfying $G(v^*) = 0$ also satisfies the KKT conditions (4.1) following rescaling of (x^*, s^*, z^*) by τ^*.

For (ii), suppose that $\{v^k\} \in \mathcal{D}, k = 1, 2, \ldots$, is any bounded sequence with limit v^*. Since $\tau^k \to 0$ by assumption, it follow that $(x^k)^\top P x^k \to 0$ since $\frac{1}{\tau^k} x^k P x^k$ must remain bounded. Hence $Px^* = 0$ since P is assumed positive semidefinite, and both $Ax^* + s^* = 0$ and $A^\top z^* = 0$. Since $\{s^k, z^k\} \in \mathcal{K} \times \mathcal{K}^*$ by assumption and both cones are closed, $z^* \in \mathcal{K}^*$ and $-Ax^* \in \mathcal{K}$. Since $\langle q, x^* \rangle + \langle b, z^* \rangle = -\kappa^*$ with $\kappa^* > 0$,

then at least one of the inner product terms must be negative. Consequently, if $\langle q, x^* \rangle < 0$ then $x^* \in \mathbb{D}$ and $(\mathcal{D})$ is infeasible, and if $\langle b, z^* \rangle < 0$ then $z^* \in \mathbb{P}$ and $(\mathcal{P})$ is infeasible. □

4.3 Barrier functions and the central path

In Clarabel, we implemented an infeasible primal-dual interior point method following ECOS [45] with the variation of dealing with the quadratic objective within the homogeneous embedding as shown in the previous section. Our goal is to solve the nonlinear system $(\mathcal{H})$, which yields the same optimal solution (infeasibility) as the original $(\mathcal{P})$ and $(\mathcal{D})$. The logarithmically-homogeneous self-concordant barriers are introduced to tackle conic constraints $\mathcal{K}, \mathcal{K}^*$ within the nonlinear system $(\mathcal{H})$ and build up connection between primal and dual variables.

4.3.1 Logarithmically-homogeneous self-concordant barrier in conic optimization

We support two types of cones in Clarabel. The first are symmetric cones where each cone is self-scaled [13], i.e. a cone is homogeneous and self-dual. It includes

- *Nonnegative cone* $\mathcal{K}_{\geq}^n := \{x \in \mathbb{R}^n : x \geq 0\}$.
- *Second order cone* $\mathcal{K}_q^n := \{(x, y) \in \mathbb{R}^n : x \geq \|y\|_2\}$.
- *Positive semidefinite cone* $\mathcal{K}_{\succeq}^n := \{X \in \mathbb{S}^n : X \succeq 0\}$.

The others are nonsymmetric cones that are not self-scaled. Two important classes are exponential cones and power cones:

- *Exponential cone* $\mathcal{K}_{\text{exp}} := \{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_2 \cdot e^{x_1/x_2} \leq x_3, x_2 > 0\} \cup \{(x_1, 0, x_3), x_1 \leq 0, x_3 \geq 0\}$.
- *Power cone* $\mathcal{K}_{\text{pow}} := \{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_1^{\alpha} x_2^{1-\alpha} \geq |x_3|, x_1, x_2 \geq 0, \alpha \in (0, 1)\}$.

They can be used to model constraints related to exponential or arbitrary power function with power $p > 1$ that can not be modelled by the symmetric cones

above [40]. For the two nonsymmetric cones above, the corresponding dual cones are

- *Dual exponential cone* $\mathcal{K}_{\text{exp}}^* := \{(z_1, z_2, z_3) \in \mathbb{R}^3 : z_3 \geq -z_1 \cdot e^{z_2/z_1 - 1},\ z_3 > 0, z_1 < 0\} \cup \{(0, z_2, z_3), z_2 \geq 0, z_3 \geq 0\}$.
- *Dual power cone* $\mathcal{K}_{\text{pow}}^* := \{(z_1, z_2, z_3) \in \mathbb{R}^3 : (\frac{z_1}{\alpha})^\alpha \cdot (\frac{z_2}{1-\alpha})^{1-\alpha} \geq |z_3|,\ z_1, z_2 \geq 0, \alpha \in (0, 1)\}$.

We then define the following logarithmically-homogeneous self-concordant barrier functions for the cones above:

1. *Nonnegative cone* $\mathcal{K}_{\geq}^n$ of degree n:

$$f(x) = -\sum_{i \in [\![n]\!]} \log(x_i),\ x \in \mathcal{K}_{\geq}^n.$$

2. *Second order cone* $\mathcal{K}_q^n$ of degree 1:

$$f(x) = -\frac{1}{2} \log\left(x_1^2 - \sum_{i=2}^n x_i^2\right),\ x \in \mathcal{K}_q^n.$$

3. *Positive semidefinite cone* $\mathcal{K}_{\succeq}^n$ of degree n:

$$f(x) = -\log\det\left(\text{mat}(x)\right),\ \text{mat}(x) \in \mathcal{K}_{\succeq}^n.$$

4. *Dual exponential cone* $\mathcal{K}_{\text{exp}}^*$ of degree 3:

$$f^*(z) = -\log\left(z_2 - z_1 - z_1 \log\left(\frac{z_3}{-z_1}\right)\right) - \log(-z_1) - \log(z_3),\ z \in \mathcal{K}_{\text{exp}}^*. \quad (4.6)$$

5. *Dual power cone* $\mathcal{K}_{\text{pow}}^*$ of degree 3:

$$f^*(z) = -\log\left(\left(\frac{z_1}{\alpha}\right)^{2\alpha} \left(\frac{z_2}{1-\alpha}\right)^{2(1-\alpha)} - z_3^2\right) - (1-\alpha)\log(z_1) - \alpha\log(z_2),\ z \in \mathcal{K}_{\text{pow}}^*. \quad (4.7)$$

Computation of derivatives for these cones is summarized in Appendix A.1. For symmetric cones, the barrier function f and its conjugate f^* can be computed analytically. In contrast, the barrier function f and its conjugate f^* for nonsymmetric cones can't have explicit forms simultaneously and one of them should be computed via numerical methods [105], see Appendix A.2.

4.3.2 Central path

We assume that $f : \mathcal{K} \to \mathbb{R}$ is a ν-LHSCB function on $\mathcal{K}$ with conjugate barrier f^* and degree $\nu > 0$. Given any initial $v^0 \in \mathcal{D}$, we define the *central path* $v^*(\mu)$ as the unique solution to

$$G(v) = \mu G(v^0), \tag{4.8a}$$

$$s = -\mu \nabla f^*(z), \tag{4.8b}$$

$$\tau\kappa = \mu, \tag{4.8c}$$

which implies that

$$\frac{\langle s, z \rangle + \tau\kappa}{\nu + 1} = \mu.$$

If the cone $\mathcal{K}$ is symmetric, then the condition (4.8b) is equivalent [41] to

$$s \circ z = \mu \mathbf{e}, \tag{4.8d}$$

where $\circ$ is the Jordan product and $\mathbf{e}$ is the idempotent of $\mathcal{K}$. Relevant properties and definitions of Jordan algebras are detailed in Appendix A.3.

The central path is uniquely defined by μ when $\mathcal{K}$ is a symmetric cone [96, Corollary 4.4] or when there is no quadratic cost, i.e. $P = 0$ [93, Theorem 5.1.4].

4.4 Infeasible primal-dual interior point methods

We implement an infeasible primal-dual interior point method over the homogeneous embedding in Clarabel. It is a path-following method that keeps iterates $(x^k, s^k, z^k, \tau^k, \kappa^k)$ close to the central path (4.8) and follows the direction of decreasing μ. In an optimization problem, $\mathcal{K}$ is usually the concatenation of multiple cones, i.e. $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_p$ and the degree of $\mathcal{K}$ is the sum of degrees of each individual cone, i.e. $\nu = \sum_{i \in [\![p]\!]} \nu_i$.

4.4.1 Computing initial points

The initial points of an interior point solver should be set to an interior point of conic constraints and should be close enough to the central path. We have two different strategies for initialization. One is specialized for symmetric cones and the other one is for problems with nonsymmetric cones.

Symmetric cones

We follow [41] for initialization when $\mathcal{K}$ is symmetric, i.e. $\mathcal{K} = \mathcal{K}^*$. Scalars τ^0, κ^0 are set to 1.

If $P \neq 0$, we solve the linear system

$$\begin{bmatrix} P & A^\top \\ A & -I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix} \tag{4.9}$$

first, which is the solution to the problem

$$\min \quad \frac{1}{2}x^\top Px + q^\top x + \frac{1}{2}\|Ax - b\|_2^2.$$

We set $x^0 = x$ and

$$s^0 = \begin{cases} -z, & \alpha_p < -\epsilon \\ -z + (\epsilon + \alpha_p)\mathbf{e} & \text{otherwise} \end{cases},$$

where $\alpha_p = \inf\{\alpha \mid -z + \alpha\mathbf{e} \in \mathcal{K}\}$. We introduce a threshold $\epsilon > 0$ to ensure s^0 is away from the boundary of cone $\mathcal{K}$. Likewise, z^0 is set to

$$z^0 = \begin{cases} z, & \alpha_d < -\epsilon \\ z + (\epsilon + \alpha_d)\mathbf{e} & \text{otherwise} \end{cases},$$

where $\alpha_d = \inf\{\alpha \mid z + \alpha\mathbf{e} \in \mathcal{K}\}$. The idempotents $\mathbf{e}$ for different cones are detailed in Appendix A.3.

If $P = 0$, we solve the linear system

$$\begin{bmatrix} 0 & A^\top \\ A & -I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}, \tag{4.10}$$

which tries to minimize the linear residuals. We set $x^0 = x$ and

$$s^0 = \begin{cases} -z, & \alpha_p < -\epsilon \\ -z + (\epsilon + \alpha_p)\mathbf{e} & \text{otherwise} \end{cases},$$

where $\alpha_p = \inf\{\alpha \mid -z + \alpha\mathbf{e} \in \mathcal{K}\}$. Then, we solve another linear system

$$\begin{bmatrix} 0 & A^\top \\ A & -I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} -q \\ 0 \end{bmatrix}, \tag{4.11}$$

which is equivalent to a least norm problem

$$\begin{aligned} \min_z \quad & \frac{1}{2}\|z\|_2^2 \\ s.t. \ & A^\top z + q = 0 \end{aligned}.$$

We set

$$z^0 = \begin{cases} z, & \alpha_d < -\epsilon \\ z + (\epsilon + \alpha_d)\mathbf{e} & \text{otherwise} \end{cases},$$

where $\alpha_d = \inf\{\alpha \mid z + \alpha\mathbf{e} \in \mathcal{K}\}$.

Nonsymmetric cones

When $\mathcal{K}$ contains a nonsymmetric cone, we instead apply unit initialization [48, 92]. In this case, we initialize both primal and dual variables on the central path satisfying $z = s = -\nabla f^*(z)$ ($\mu^0 = 1$), which is equivalent to solving the unconstrained optimization

$$\min_z \ \frac{1}{2}z^2 + f^*(z),$$

which is strictly convex and has a unique solution. It yields $s^0_{sym} = z^0_{sym} = \mathbf{e}$ for symmetric cones,

$$s^0_{\text{exp}} = z^0_{\text{exp}} = (-1.051383945322714, 0.556409619469370, 1.258967884768947)$$

for exponential cones and

$$s^0_{\text{pow}} = z^0_{\text{pow}} = (\sqrt{1+\alpha}, \sqrt{2-\alpha}, 0)$$

for power cones of parameter α.

4.4.2 Scaling matrices

The interior point method can be regarded as a Newton-type method that tries to solve the nonlinear system (4.8) with $\mu \to 0$. The nonlinearity comes from (4.8b) and (4.8c), which is the approximation of complementary slackness in the KKT condition (2.6). We know $s = -\mu \nabla f^*(z)$ and $z = -\mu \nabla f(s)$ hold simultaneously for the pair (s, z) on the central path, and this condition can also be expressed as (4.8d) for symmetric cones. Hence, there are several ways to linearize the central path (4.8b) (or (4.8c)), i.e. the choice of *scaling matrix* H^k, defined later in (4.17), changes at every iteration k and is not unique in an interior point method.

For a symmetric cone, the most common choice is the NT scaling [13, 14]. The great majority of research on IPMs, such as linear programming (LP), quadratic programming (QP), second order cone programming (SOCP) and semidefinite programming (SDP), belongs to this class. For nonsymmetric cones which are not self-scaled, the central path is (4.8b) and both the primal-dual symmetric scaling [92] and the nonsymmetric scaling [93] are implemented in the state-of-the-art solvers like Mosek and ECOS respectively. We assume $\mathcal{K}$ is a single cone in this section to simplify the analysis.

Symmetric cones

The NT scaling is based on the self-scaled property of symmetric cone $\mathcal{K}$ that, for $s, z \in \mathcal{K}$, there is a unique scaling point $w \in \mathcal{K}$ satisfies [13, Theorem 3.2]

$$H(w)s = z,$$

where $H(w)$ is the Hessian of the barrier function $f(\cdot)$ and can be formed as $H^{-1}(w) = W^\top W$ and we set $H^k = H^{-1}(w^k)$ at iteration k later in (4.17). The values of w, W are detailed in Appendix A.4.

Nonsymmetric cones

The self-scaled property doesn't hold for nonsymmetric cones and the central path can't be formulated as (4.8d). Several other scaling strategies have therefore been proposed.

Symmetric scaling: A general primal-dual symmetric scaling strategy was proposed in [106] and used later in [92]. It relies on the satisfaction of two secant equations for the set of scaling matrix ($\mathcal{W}_1(s,z)$) instead of a scaling point for $\mathcal{K}$,

$$\mathcal{W}_1(s,z) := \{W : W \succ 0, W^2 z = s, W^2 \nabla f(s) = \nabla f^*(z)\}. \tag{4.12}$$

Suppose we define shadow iterates as

$$\tilde{z} := -\nabla f(s), \quad \tilde{s} := -\nabla f^*(z), \tag{4.13}$$

with

$$\tilde{\mu} = \langle \tilde{s}, \tilde{z} \rangle / \nu. \tag{4.14}$$

A scaling matrix $H^k = {W^k}^\top W^k$ can be obtained from the rank-4 Broyden-Fletcher-Goldfarb-Shanno (BFGS) update, which is commonly used in quasi-Newton methods,

$$H_{\text{BFGS}} := Y(Y^\top S)^{-1} Y^\top + H_q - H_q S (S^\top H_q S)^{-1} S^\top H_q,$$

where $Y = [z, \tilde{z}], S = [s, \tilde{s}], \tilde{z} = -\nabla f(s), \tilde{s} = -\nabla f^*(z)$ and $H_q \succ 0$ is an approximation of the Hessian. In our implementation, we choose $H_q = \mu \nabla^2 f^*(z)$ following [92] and the update of H_{BFGS} reduces to a rank-3 update,

$$\begin{aligned} H_{\text{BFGS}} =& \mu \nabla^2 f^*(z) + \frac{1}{2\mu\nu} \delta_s \left(s + \mu\tilde{s} + \frac{1}{\mu\tilde{\mu} - 1} \delta_s \right)^T + \frac{1}{2\mu\nu} \left(s + \mu\tilde{s} + \frac{1}{\mu\tilde{\mu} - 1} \delta_s \right) \delta_s^T \\ & - \mu \frac{\left(\nabla^2 f^*(z)\tilde{z} - \tilde{\mu}\tilde{s}\right) \left(\nabla^2 f^*(z)\tilde{z} - \tilde{\mu}\tilde{s}\right)^T}{\langle \tilde{z}, \nabla^2 f^*(z)\tilde{z} \rangle - \nu\tilde{\mu}^2}, \end{aligned} \tag{4.15}$$

where we define $\delta_s := s - \mu\tilde{s}, \delta_z := z - \mu\tilde{z}$. More details about the primal-dual symmetric scaling of nonsymmetric cones can be found in [92].

Nonsymmetric scaling: Nonsymmetric scaling has been used for various nonsymmetric cones within an IPM such as exponential cone [93], power cone [15], sparse semidefinite cone [16] and the sum-of-square cone [17]. It chooses the scaling matrix H^k as

$$H^k = \mu^k \nabla^2 f^*(z^k), \tag{4.16}$$

which appears in the linearization of the central path $s = -\mu \nabla f^*(z)$ and μ^k is computed via $\mu^k = \langle s^k, z^k \rangle / \nu.$. The detailed implementation for the nonsymmetric scaling of nonsymmetric cones can be found in [93].

In Clarabel, we support two nonsymmetric cones, the exponential cone and the power cone with the symmetric scaling plus 3rd-order corrections, which requires fewer iterations to converge empirically compared to the nonsymmetric scaling as noticed in [92]. The nonsymmetric scaling will be exploited for high-dimensional power cones in Chapter 5.

4.4.3 Computing step directions

Our interior point method computes Newton-like search directions using a linearization of (4.8) given some right-hand side residual $d := (d_x, d_z, d_\tau, d_s, d_\kappa)$. This produces a linear system in the form

$$\begin{bmatrix} 0 \\ \Delta s \\ \Delta \kappa \end{bmatrix} - \begin{bmatrix} P & A^\top & q \\ -A & 0 & b \\ -(q + 2P\xi)^\top & -b^\top & \xi^\top P \xi \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta z \\ \Delta \tau \end{bmatrix} = - \begin{bmatrix} d_x \\ d_z \\ d_\tau \end{bmatrix} \tag{4.17a}$$

$$H \Delta z + \Delta s = -d_s, \quad \kappa \Delta \tau + \tau \Delta \kappa = -d_\kappa, \tag{4.17b}$$

where $\xi := x\tau^{-1}$ and $H \in \mathbb{S}^n_{++}$ is the scaling matrix detailed in Section 4.4.2.

Our approach to solving (4.17) now follows that of [41, 45], and differs only in the fact that some of the blocks in the coefficient matrix of (4.17a) include an additional term when $P \neq 0$. We first eliminate the variables $(\Delta s, \Delta \kappa)$ to obtain the reduced system

$$\begin{bmatrix} P & A^\top & q \\ -A & H & b \\ -(q + 2P\xi)^\top & -b^\top & \xi^\top P \xi + \frac{\kappa}{\tau} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta z \\ \Delta \tau \end{bmatrix} = \begin{bmatrix} d_x \\ d_z - d_s \\ d_\tau - d_\kappa / \tau \end{bmatrix} \tag{4.18a}$$

$$\Delta s = -d_s - H\Delta z, \quad \Delta\kappa = -(d_\kappa + \kappa\Delta\tau)/\tau. \tag{4.18b}$$

To solve (4.18) we first solve a pair of linear systems with a common left-hand side

$$\underbrace{\begin{bmatrix} P & A^\top \\ A & -H \end{bmatrix}}_{K:=} \left[\begin{array}{c|c} \Delta x_1 & \Delta x_2 \\ \Delta z_1 & \Delta z_2 \end{array}\right] = \left[\begin{array}{c|c} d_x & -q \\ -(d_z - d_s) & b \end{array}\right] \tag{4.19}$$

and then recover $\Delta\tau$ by

$$\begin{aligned} \Delta\tau &= \frac{d_\tau - d_\kappa/\tau + (2P\xi + q)^\top \Delta x_1 + b^\top \Delta z_1}{\kappa/\tau + \xi^\top P\xi - (2P\xi + q)^\top \Delta x_2 - b^\top \Delta z_2} \\ &= \frac{d_\tau - d_\kappa/\tau + q^\top \Delta x_1 + b^\top \Delta z_1 + 2\xi^\top P\Delta x_1}{\|\Delta x_2 - \xi\|_P^2 - \|\Delta x_2\|_P^2 - q^\top \Delta x_2 - b^\top \Delta z_2}, \\ \Delta x &= \Delta x_1 + \Delta\tau \cdot \Delta x_2, \quad \Delta z = \Delta z_1 + \Delta\tau \cdot \Delta z_2, \end{aligned}$$

and $\Delta s, \Delta\kappa$ via (4.18).

Linear solve methods

Nearly all of the computational cost of computing step directions comes when solving the symmetric linear system (4.19). When solving this system via a direct factorization method, we add a small regularization term ϵ_s and compute an LDL factorization

$$LDL^\top = (K + \Delta K) := \begin{bmatrix} P + \epsilon_s I & A^\top \\ A & -(H + \epsilon_s I) \end{bmatrix}.$$

This *static regularization* ensures that the matrix is quasidefinite [107] even if P is rank deficient, and consequently that the factor D is diagonal with $D_{ii} \neq 0$ [108]. When computing the LDL factorization we also employ a *dynamic regularization* strategy by lower bounding the magnitude of the pivots away from zero by an amount ϵ_d to ensure that the factorization is numerically stable. We have extended the open-source package QDLDL, originally developed for OSQP and based on [109], to support this regularization strategy.

Due to the regularization for numerical stability in the LDL factorization, we are solving a perturbed system of the form

$$(K + \Delta K)y = r$$

for (4.19) and the error will be $r - Ky$. We apply *iterative refinement*, i.e. solving

$$(K + \Delta K)\Delta y = r - Ky$$

and then updating $y \leftarrow y + \Delta y$ iteratively with y will converge to the true solution of $Ky = r$ [100].

Remark 4.4.1. The improved numerical efficiency of our scheme relative to the standard HSDE method can be explained by consideration of the linear system in (4.19). If we had started instead from the epigraphical reformulation (4.3), then the coefficient matrix in (4.19) would have been

$$\begin{bmatrix} 0 & A^T & [P^{\frac{1}{2}}]^T \\ A & -H & \\ P^{\frac{1}{2}} & & -H_{\mathcal{K}} \end{bmatrix} \tag{4.20}$$

with a scaling matrix $H_{\mathcal{K}}$ for the additional second order cone constraint introduced in (4.3). Direct factorization of (4.20) will produce significantly more fill-in than for coefficient matrix (4.19), particular when the matrix factor $P^{\frac{1}{2}}$ already has substantial fill-in.

Remark 4.4.2. We have two different implementations for the scaling matrix H w.r.t. the second-order cones. When the dimension n of a second-order cone is $n > 5$, we follow the same sparse augmentation of the scaling matrix as in ECOS, which is detailed in Section A.5. However, we use the dense form for the scaling matrices of second-order cones when $n \leq 5$ since it has less fill-in and is also more numerically stable compared to ECOS's augmentation. Indeed, there are many of applications that only require second-order cones of $n = 3$ or $n = 4$, e.g. lossless convexification in the space landing problems [110] and SOCPs for optimal power flow problems [111].

4.4.4 Affine and centering step directions

Our IPM needs to solve (4.19) with two different sets of $(d_x, d_z, d_\tau, d_s, d_\kappa)$. The first one is called the *affine step* where we estimate $(\Delta x, \Delta s, \Delta z, \Delta\kappa, \Delta\tau)$ trying to eliminate the linear residuals. The second is called the *combined step*, which

is the affine step plus a *centering step* toward the central path. In practice, the estimated direction $(\Delta x, \Delta s, \Delta z, \Delta\kappa, \Delta\tau)$ from the affine step is used to compute the higher-order correction for acceleration in $(d_x, d_z, d_\tau, d_s, d_\kappa)$ of the combined step. In the view of predictor-corrector algorithms, the affine step is the *predictor* and the centering step is the *corrector*.

In our work, the specification for $(d_x, d_z, d_\tau, d_s, d_\kappa)$ are

$$d_x = r_x, d_z = r_z, d_\tau = r_\tau, d_\kappa = \kappa\tau, d_s = s,$$

in the affine step, and

$$d_x = (1-\sigma)r_x, d_z = (1-\sigma)r_z, \tag{4.21a}$$

$$d_s = \begin{cases} W^\top(\lambda \backslash (\lambda \circ \lambda + \eta - \sigma\mu\mathbf{e})) & \text{(symmetric cone)} \\ s + \sigma\mu\nabla f^*(z) + \eta & \text{(nonsymmetric cone)} \end{cases}, \tag{4.21b}$$

$$d_\tau = (1-\sigma)r_\tau, d_\kappa = \kappa\tau + \Delta\kappa\Delta\tau - \sigma\mu, \tag{4.21c}$$

in the combined step, with λ detailed in Appendix A.3. There are two parameters, σ and η, closely related to the convergence of an interior point method. The *centering parameter* σ controls the decreasing rate of linear residuals and the duality gap (related to μ). We estimate σ from the affine step α_{aff} [112]

$$\sigma = (1 - \alpha_{\text{aff}})^3.$$

A larger step size α_{aff} implies the current iterate is away from the boundary of conic constraints and we can decrease μ more aggressively along the central path, i.e. a smaller σ.

Taking the affine step can only eliminate errors in the linearized model (4.17) but will inevitably introduce higher-order error given the nonlinearity of the central path (4.8). The higher-order correction η tries to take the higher-order error (from the affine step) into account when we determine the direction for the centering step. We will detail it in the next section.

4.4.5 Higher-order corrections

The higher-order correction η tries to compensate the nonlinear error introduced after taking the affine step. It is a heuristic technique that can accelerate the convergence of IPMs empirically. Considering the nonlinear path w.r.t. (τ, κ) in (4.18b), the affine step tries to offset the error in the linearized model, i.e.

$$\kappa\Delta\tau_a + \tau\Delta\kappa_a = -\kappa\tau,$$

where $d_\kappa = \kappa\tau$. However, moving along the affine search direction introduces an additional higher-order error at the new pair $(\tau + \Delta\tau_a, \kappa + \Delta\kappa_a)$,

$$(\tau + \Delta\tau_a)(\kappa + \Delta\kappa_a) = \Delta\tau_a\Delta\kappa_a.$$

The idea of higher-order correction η is to compensate the nonlinear error in the correction step (4.4.4).

We select two different higher-order corrections for symmetric and nonsymmetric cones separately. The Mehrotra correction is effective for symmetric cones [112],

$$\eta = (W^{-1}\Delta s) \circ (W\Delta z). \tag{4.22}$$

The 3rd-order correction proposed in [92] is effective for nonsymmetric cones,

$$\eta(\Delta s_a, \Delta z_a) := -\frac{1}{2}\nabla^3 f^*(z)\left[\Delta z_a, \left(\nabla^2 f^*(z)\right)^{-1}\Delta s_a\right]. \tag{4.23}$$

The computations related to higher-order correction are detailed in Appendix A.6.

4.4.6 Proximity measurement

The path-following interior point method aims to keep iterates $(x^k, s^k, z^k, \tau^k, \kappa^k)$ in the neighbourhood of the central path (4.8). Following [92, 106], we choose the neighbourhood based on *shadow iterates* defined in (4.13) and (4.14) for $(s, z) \in \mathcal{K}\times\mathcal{K}^*$, where $\mu\tilde{\mu} \geq 1$ with equality only on the central path [14]. The neighbourhood is defined as

$$\mathcal{N}(\beta) = \{(s, z) \in \mathcal{K} \times \mathcal{K}^* \mid \beta\mu\tilde{\mu}_i \leq 1, i = 1, \ldots, k+1\}, \tag{4.24}$$

for $\beta \in (0, 1]$. We use the neighborhood $\mathcal{N}(\beta)$ defined in (4.24) following the use of 3rd-order correction in [92]. Both ∇f and ∇f^* can be computed analytically for symmetric cones while we need to compute ∇f numerically if we define barriers f^* for dual exponential and dual power cones as in Section 4.3.1, see Appendix A.2 for details of computation. Different choices of proximity measurement can be found in [35, 93].

4.4.7 Termination check

We check the termination of our interior point method when we obtain a new iterate v^k. The optimality metrics are defined as

$$r_p^k := \frac{\|A\underline{x}^k + \underline{s}^k - b\|}{\max(1, \|b\|_\infty + \|\underline{x}^k\| + \|\underline{s}^k\|)}, \tag{4.25a}$$

$$r_d^k := \frac{\|P\underline{x}^k + A^\top \underline{z}^k + q\|}{\max(1, \|q\|_\infty + \|\underline{x}^k\| + \|\underline{z}^k\|)}, \tag{4.25b}$$

$$g_p^k := \frac{1}{2}\underline{x}^{k\top} P\underline{x}^k + \langle q, \underline{x}^k \rangle, \tag{4.25c}$$

$$g_d^k := -\frac{1}{2}\underline{x}^{k\top} P\underline{x}^k - \langle b, \underline{z}^k \rangle, \tag{4.25d}$$

$$r_g^k := \frac{|g_p^k - g_d^k|}{\max(1, \min(|g_p^k|, |g_d^k|))}, \tag{4.25e}$$

where $\underline{x}^k = x^k/\tau^k, \underline{s}^k = s^k/\tau^k, \underline{z}^k = z^k/\tau^k$ are the normalized variables. r_p^k, r_d^k are the primal and the dual residuals and r_g^k is the duality measurement. We claim optimality if

$$\max\{r_p^k, r_d^k, r_g^k\} \leq \epsilon,$$

where we set the default values $\epsilon = 1e^{-8}$. Likewise, we define the infeasibility metrics

$$r_{pinf}^k := \frac{\|A^\top z^k\|}{\max(1, \|x^k\| + \|z^k\|)}, \tag{4.26a}$$

$$r_{dinf}^k := \max\left(\frac{\|Px^k\|}{\max(1, \|x^k\|)}, \frac{\|Ax^k + s^k\|}{\max(1, \|x^k\| + \|s^k\|)}\right), \tag{4.26b}$$

We claim primal infeasibility if

$$\left\langle b, z^k \right\rangle < -\epsilon_{ainf}, \text{ and } \frac{r_{pinf}^k}{-\langle b, z^k \rangle} < \epsilon_{rinf},$$

and dual infeasibility if

$$\left\langle q, x^k \right\rangle < -\epsilon_{ainf}, \text{ and } \frac{r^k_{dinf}}{-\left\langle q, x^k \right\rangle} < \epsilon_{rinf}.$$

The default values for $\epsilon_{ainf}, \epsilon_{rinf}$ are $1e^{-8}$.

4.4.8 Sketch of the interior point algorithm

We outline the general algorithm inside each iteration, after the initialization step in Section 4.4.1:

1. *Update residuals, gap and check optimality or infeasibility:* We compute

$$\begin{aligned} r^k_x &= -A^\top z^k - q\tau^k, \\ r^k_z &= s^k + Ax^k - b\tau^k, \\ r^k_\tau &= \kappa^k + q^\top x^k + b^\top z^k, \\ \mu^k &= \tfrac{{s^k}^\top z^k + \kappa^k \tau^k}{\nu+1}, \end{aligned}$$

   and check the optimality and infeasibility as in Section 4.4.7.

2. *Update scaling matrix H^k* As discussed in Section 4.4.2, we update the scaling matrix H^k by

$$H^k = \begin{cases} {W^k}^\top W^k & \text{(symmetric cone)} \\ H^k_{\text{BFGS}} \text{ in (4.15)} & \text{(nonsymmetric cone)} \end{cases}.$$

3. *Affine step:* The affine direction (predictor) is computed via

$$\begin{bmatrix} 0 \\ \Delta s^k_a \\ \Delta \kappa^k_a \end{bmatrix} - \begin{bmatrix} P & A^\top & q \\ -A & 0 & b \\ -(q + 2P\xi^k)^\top & -b^\top & {\xi^k}^\top P \xi^k \end{bmatrix} \begin{bmatrix} \Delta x^k_a \\ \Delta z^k_a \\ \Delta \tau^k_a \end{bmatrix} = - \begin{bmatrix} r^k_x \\ r^k_z \\ r^k_\tau \end{bmatrix}, \tag{4.27a}$$

$$H^k \Delta z^k_a + \Delta s^k_a = -s^k, \quad \kappa^k \Delta \tau^k_a + \tau^k \Delta \kappa^k_a = -\kappa^k \tau^k, \tag{4.27b}$$

   where $\xi^k := x^k/\tau^k$ and H^k is the scaling matrix. It tries to remove residuals of the linearized model at iteration k. We can compute the maximal step size α_a such that $(s^k + \alpha_a \Delta s^k_a, z^k + \alpha_a \Delta z^k_a, \tau^k + \alpha_a \Delta \tau^k_a, \kappa^k + \alpha_a \Delta \kappa^k_a)$ resides in $\mathcal{F} := \mathcal{K} \times \mathcal{K}^* \times \mathbb{R}_+ \times \mathbb{R}_+$.

4. *Combined step:* The weight of centrality σ is set to $(1 - \alpha_a)^3$ empirically and then used for the computation of the centering direction (corrector) via

$$\begin{bmatrix} 0 \\ \Delta s^k_c \\ \Delta \kappa^k_c \end{bmatrix} - \begin{bmatrix} P & A^\top & q \\ -A & 0 & b \\ -(q + 2P\xi^k)^\top & -b^\top & {\xi^k}^\top P \xi^k \end{bmatrix} \begin{bmatrix} \Delta x^k_c \\ \Delta z^k_c \\ \Delta \tau^k_c \end{bmatrix} = -(1-\sigma) \begin{bmatrix} r^k_x \\ r^k_z \\ r^k_\tau \end{bmatrix}, \tag{4.28a}$$

$$H^k \Delta z_c^k + \Delta s_c^k = \begin{cases} {W^k}^\top \left(\lambda^k \backslash \left(\lambda^k \circ \lambda^k + \eta(\Delta s_a^k, \Delta z_a^k) - \sigma\mu^k \mathbf{e}\right)\right) & \text{(symmetric)} \\ s^k + \sigma^k \mu^k \nabla f^*(z^k) + \eta(\Delta s_a^k, \Delta z_a^k) & \text{(nonsymmetric)} \end{cases}, \tag{4.28b}$$

$$\kappa^k \Delta\tau_c^k + \tau^k \Delta\kappa_c^k = -\tau^k\kappa^k + \sigma^k\mu^k - \Delta\tau_a^k \Delta\kappa_a^k, \tag{4.28c}$$

where $\Delta\tau_a^k\Delta\kappa_a^k$ and $\eta(\Delta s_a^k, \Delta z_a^k)$ are higher-order corrections given information from the affine directions. We use the Mehrotra's correction (4.22) for symmetric cones and the 3rd-order correction (4.23) for nonsymmetric cones. Likewise, we compute the largest step size α_c ensuring $(s^k + \alpha_c\Delta s_c^k, z^k + \alpha_c\Delta z_c^k, \tau^k + \alpha_c\Delta\tau_c^k, \kappa^k + \alpha_c\Delta\kappa_c^k)$ stays inside conic constraints $\mathcal{F}$ and the neighbourhood $\mathcal{N}(\beta)$ (4.24) of the central path.

5. *Update iterates:* At the end of each iteration k, we obtain the new iterate $(x^{k+1}, s^{k+1}, z^{k+1}, \kappa^{k+1}, \tau^{k+1})$ by

$$(x^{k+1}, s^{k+1}, z^{k+1}, \kappa^{k+1}, \tau^{k+1}) := (x^k, s^k, z^k, \kappa^k, \tau^k) + \alpha_c(\Delta x_c^k, \Delta s_c^k, \Delta z_c^k, \Delta\kappa_c^k, \Delta\tau_c^k).$$

4.5 Implementation: the Clarabel solver

We have implemented our proposed approach in the *Clarabel* solver [46], an open-source and liberally licensed software package with separate implementations in both the Rust [113] and Julia [114] programming languages. Both implementations are publicly available[2][3] under the Apache v2.0 license.

Our Rust implementation is intended for most academic and industrial end users and as an implementation that can be accessed via other common languages through standard foreign function interfaces (FFIs) and wrappers that we provide. We currently provide such wrappers for Python, R and C/C++. We further provide interfaces in Python to the standard modelling package CVXPY [115, 116]. Clarabel is installed as part of the standard CVXPY distribution [117] as of version 1.4.

The Rust version of Clarabel provides its own internal implementation of most linear algebra functionality, including a stand-alone Rust reimplementation of the

[2]Julia: https://github.com/oxfordcontrol/Clarabel.jl
[3]Rust: https://github.com/oxfordcontrol/Clarabel.rs

quasidefinite linear solver QDLDL, which was first implemented for OSQP [118] with additional features to support the dynamic regularization method described in §4.4.3. We use Rust interfaces to user specified BLAS [119] implementations for solving conic programs on the semidefinite cone.

The Julia implementation is intended to be used both as a standalone solver for users of the Julia language and as as prototyping and development platform for further algorithmic development. The Julia implementation relies heavily on native Julia functions for most linear algebra functionality, with the exception of a Julia implementation of QDLDL which we provide as a standalone package. In Julia we also provide the option of using alternative linear solve methods including CHOLMOD [120], Pardiso [121] and the HSL MA57 [122] solver. All numerical results are based on our QDLDL implementation for simplicity of comparison between implementations.

Both implementations provide identical functionality and support the same set of conic constraints. We also provide support for different floating point data types in both languages, e.g. for standard 32- or 64-bit single or double precision floating point types [123] or for extended precision types such as the Julia BigFloat type. Our implementation is inspired by the modular design pattern of the interior point solver OOQP [124], in the sense that all internal data types are defined as abstract types that can be extended or customised by end users to specific problem classes to exploit domain-specific structure. In the Rust implementation this functionality relies heavily on Rust's trait-based type system and generics, while in Julia we instead rely on Julia's dynamic dispatch and "duck typing" [125].

4.6 Numerical experiments

We have benchmarked our implementation of Clarabel against a variety of open-source and commercial solvers: the open-source interior point solver ECOS [45], the open-source dual-simplex base solver HiGHS [126], and the commercial interior point solvers Mosek [40] and Gurobi [127]. We execute all benchmarks with all

default settings for all solvers enabled, but with pre-solve disabled where applicable to ensure that the solvers are solving equivalent problems. We do not impose any iteration limits other than those specified within each solver's internal defaults. In all cases we set the maximum solve time to 300 seconds.

Our implementation of Clarabel relies on a Rust-language implementation of the direct LDL linear solver QDLDL, which is the same method used in the ADMM-based solvers COSMO [19] (implemented in Julia) and OSQP [118] (implemented in C). The QDLDL solver is relatively unsophisticated relative to the multi-threaded methods used in commercial solvers, but is lightweight, simple to implement and does not rely on any external libraries such as BLAS.

All experiments were carried out on the Oxford University Advanced Research Computing (ARC) Facility[128]. For each test problem in our benchmarks results, solver were run single threaded on an Intel Xeon Platinum 8268 CPU @ 2.90GHz with 64GB RAM. All benchmarks tests are scripted in Julia and access solver interfaces via JuMP [129]. We use Rust compiler version 1.72.0 for Clarabel and the Julia version 1.9.2. The code for all numerical examples is publicly available [130].

For each set of benchmark problems in our results, we exclude problems for which *none* of the benchmarked solvers produced a valid solution. We provide a summary of the results for all benchmarked solvers appropriate to each problem class in the form of shifted geometric means and performance profiles in the remainder of this section.

For all benchmark tests sets, we provide more detailed numerical results for our Rust and Julia implementations as well as the solvers ECOS and Mosek in Appendix A.7, including solve times and iteration counts. We include only this subset of solvers in our detailed reporting since all are interior point based methods and therefore have iteration counts that are directly comparable.

Shifted geometric means We follow the standard benchmarking convention [131] of using a normalised shifted geometric mean for comparison of solve time across different solvers. For a set of N test problems, we define the shifted geometric mean solve time g_s for solver s as

$$g_s := \left[\prod_{p=1}^{N}(t_{p,s}+k)\right]^{\frac{1}{N}} - k,$$

where $t_{p,s}$ is the time in seconds for solver s to solve problem p, and $k = 1$ is the shift. The normalised shifted geometric mean is then defined as

$$r_s := \frac{g_s}{\min_s g_s},$$

so that the solver with least overall shifted geometric mean solve time has a normalised score of 1. For those problems for which a given solver fails, we assign a solve time $t_{p,s}$ equal to the maximum allowable solve time for the relevant benchmark.

Performance profiles We also provide performance profiles [132] to compare both the relative and absolute performance of different solvers. For a set of N test problems, we define the *relative performance ratio* for solver s and problem p as

$$u_{p,s} = \frac{t_{p,s}}{\min_s t_{p,s}}.$$

The performance profile for the solver s is then a plot of the function $f_s^r : \mathbb{R}_+ \mapsto [0,1]$ defined as

$$f_s^r(\tau) := \frac{1}{N}\sum_p \mathcal{I}_{\leq\tau}(u_{p,s}).$$

where $\mathcal{I}_{\leq\tau} = 1$ if $\tau \leq u_{p,s}$ and $\mathcal{I}_{\leq\tau} = 0$ otherwise. The relative performance profile therefore shows, at each level τ, the fraction of problems solved by solver s in time within a factor τ of the solve time of the best solver.

Since the relative performance profile for a given solver can change depending on the overall collection of solvers being benchmarked, we further compute an *absolute performance profile* by plotting a function $f_s^a : \mathbb{R}_+ \mapsto [0,1]$ as

$$f_s^a(\tau) := \frac{1}{N}\sum_p \mathcal{I}_{\leq\tau}(t_{p,s}).$$

The absolute performance profile then shows, at each level τ, the fraction of problems solved by solver s within τ seconds and is independent of the other solvers being benchmarked.

4.6.1 Benchmark problems with quadratic objectives

In this section we present benchmark results for quadratic programming (QP) problems in the standard form ($\mathcal{P}$) with the set $\mathcal{C}$ restricted to the composition of the zero cone (i.e. modelling equalities) and the nonnegative orthant. We consider example problems taken or generated from standard open-source problem collections and covering a wide range of problem dimensions.

The Maros-Meszaros test set

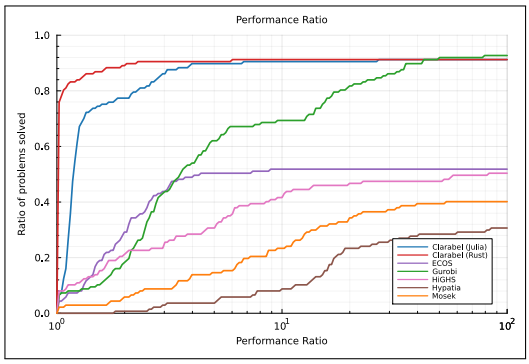
We consider first the standard benchmark collection of 138 quadratic programs from the Maros-Meszaros test set [133]. This collection of QPs includes a wide range of problem sizes and contains a number of difficult test cases due to numerical ill-conditioning, rank deficiency or poor scaling.

Results Results for this benchmark set are shown in Figure 4.1 for all solvers. Clarabel is the fastest overall solver on this benchmark set, with the Rust implementation marginally faster as expected. Note that the seemingly large gap in the relative performance profile of our Rust and Julia implementations is almost entirely attributable to faster solve times among the small examples in this test set. All solvers fail on at least some subset of these benchmarks, with Gurobi the lowest failure rate (at full accuracy) and Clarabel the lowest failure rate (at reduced accuracy).

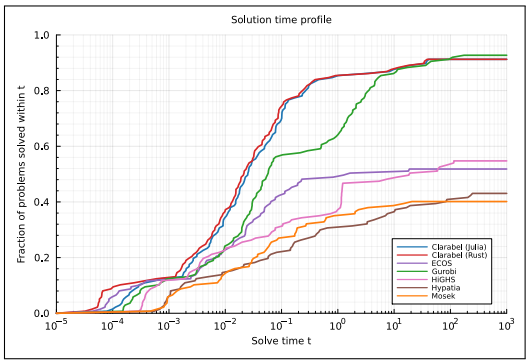
Of particular note in this test is the high failure rate of the ECOS and Mosek solvers, since both are interior point methods broadly similar to Clarabel. The reason for these failures in the case of ECOS is partly attributable to the solver's requirement to reformulate QP problems in the conic form (4.20), since it does not support quadratic objectives natively. This leads to immediate failures in a substantial number of cases due to ill-conditioning of the matrix P, resulting in

a failed attempt to compute the Cholesky factor $P^{\frac{1}{2}}$ in (4.20) when P is either semidefinite or contains very small negative eigenvalues. Mosek handles this case more robustly, but is still not able to solve a substantial number of problems to full accuracy within the benchmark time limit.

Figure 4.1: Performance profiles for the Maros-Meszaros problem set



(a) Relative performance profile



(b) Absolute performance profile

		ClarabelRs	Clarabel	ECOS	Gurobi	HiGHS	Hypatia	Mosek
Shifted GM	Full Acc.	1.0	1.02	15.49	1.64	17.92	36.61	32.67
	Low Acc.	1.0	1.1	19.8	2.9	39.99	42.99	4.07
Failure Rate (%)	Full Acc.	8.8	8.8	48.2	7.3	45.3	56.9	59.9
	Low Acc.	2.2	2.9	38.0	3.6	45.3	42.3	10.2

(c) Benchmark timings as shifted geometric mean and failure rates

Least-squares problems with SuiteSparse matrices

We next consider a collection of 23 sparse least-squares problems $Ax \approx b$ derived from matrices taken from the SuiteSparse Matrix Collection [134], following the equivalent set of benchmark examples from [118]. For each case we compute an approximate solution in by solving a constrained QP using two standard methods:

Huber Problem: The *Huber fitting* [135, 136] or *robust least squares* problem for a given matrix A and vector b is defined as

$$\min_{x} \quad \sum_{i=1}^{m} \phi(a_i^T x - b_i), \tag{4.29}$$

where the *Huber loss function* $\phi : \mathbb{R} \to \mathbb{R}$ is defined as

$$\phi(w) = \begin{cases} w^2 & |w| \leq M \\ M(2|w| - M) & |w| > M. \end{cases} \tag{4.30}$$

We set $M = 1$ for all test cases. This problem is equivalent [137] to the following quadratic program:

$$\begin{aligned} \min_{x,u,r,s} \quad & u^\top u + 2M\mathbf{1}^\top(r+s) \\ \text{subject to:} \quad & Ax - b - u = r - s \\ & (r,s) \geq 0. \end{aligned} \tag{4.31}$$

LASSO Problem: The *least absolute shrinkage and selection operator (LASSO)* problem [8, 9] for a given matrix A and vector b is defined as

$$\min_x \quad \|Ax - b\|_2^2 + \lambda \|x\|_1 . \tag{4.32}$$

We set $\lambda = \left\|A^\top b\right\|_\infty$ for all test cases. This problem is equivalent [137] to the following quadratic program:

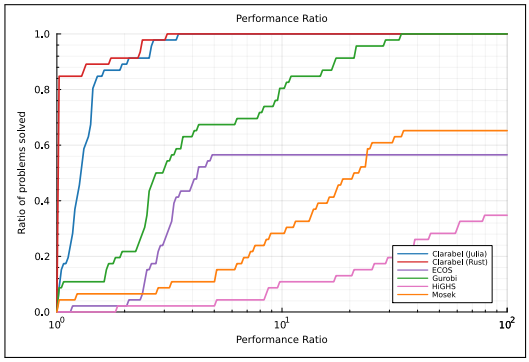
$$\begin{aligned} \min_{y,x,t} \quad & y^\top y + \lambda\mathbf{1}^\top t \\ \text{subject to:} \quad & Ax - b = y \\ & -t \leq x \leq t. \end{aligned} \tag{4.33}$$

Results Results for this benchmark set of 46 problems are shown in Figure 4.2. Clarabel is again the fastest solver overall with the Rust implementation marginally faster. In this test set only the Clarabel and Gurobi solvers are able to solve all cases to full accuracy.

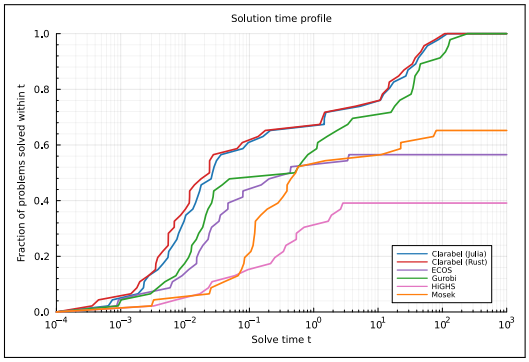
Constrained optimal control

Finally, we consider finite-horizon constrained optimal control problems with quadratic objectives. Problems of this type are of particular interest in embedded control systems, since the repeated online solution of such problems is the basis of the model predictice control (MPC) method. We consider a collection of 72 such problems taken from the benchmark collection of industrial and academic applications in [138]. Problems in this collection are in the form

Figure 4.2: Performance profiles for the SuiteSparse least-squares problem set



(a) Relative performance profile



(b) Absolute performance profile

		ClarabelRs	Clarabel	ECOS	Gurobi	HiGHS	Mosek
Shifted GM	Full Acc.	1.0	1.06	7.12	1.79	21.5	6.31
	Low Acc.	1.0	1.06	5.43	1.79	21.5	1.68
Failure Rate (%)	Full Acc.	0.0	0.0	43.5	0.0	60.9	34.8
	Low Acc.	0.0	0.0	39.1	0.0	60.9	0.0

(c) Benchmark timings as shifted geometric mean and failure rates

$$
\begin{aligned}
\min_{y,x,u} \quad & \sum_{i=0}^{N-1} \begin{pmatrix} y_i - y_i^r \\ u_i - u_i^r \end{pmatrix} \begin{pmatrix} Q_k & S_k \\ S_k^T & R_k \end{pmatrix} \begin{pmatrix} y_i - y_i^r \\ u_i - u_i^r \end{pmatrix} + \begin{pmatrix} g_k^y \\ g_u^k \end{pmatrix}^\top \begin{pmatrix} y_i - y_i^r \\ u_i - u_i^r \end{pmatrix} \\
& + (x_N - x_N^r)^\top P (x_N - x_N^r)
\end{aligned}
$$

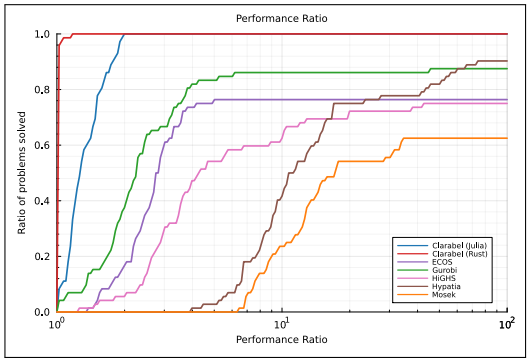
$$
\text{subject to:} \quad \left.\begin{aligned}
& x_{k+1} = A_k x_k + B_k u_k + f_k \\
& y_k = C_k x_k + D_k u_k + e_k \\
& d_k^\ell \leq M_k x_k + N_k u_k \leq d_k^u \\
& u_k \in \mathcal{U}_k, \;\; y_k \in \mathcal{Y}_k
\end{aligned}\right\} \quad k = 0 \ldots N-1
$$

$$
T x_N \in \mathcal{T}, \tag{4.34}
$$

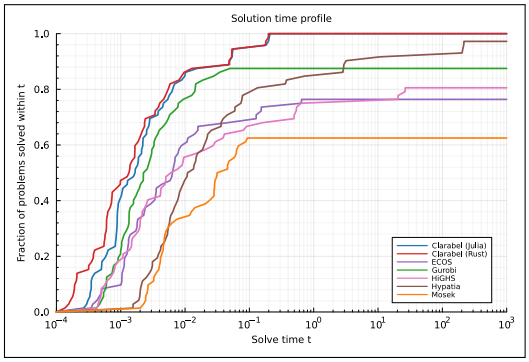
where the constraint sets $\mathcal{U}_k$, $\mathcal{Y}_k$ and $\mathcal{T}$ are interval constraints. All problems have $Q_k \succeq 0$, $R_k \succeq 0$ and $P \succ 0$, which ensures that the problems are all convex QPs. As is typical of optimal control problems for embedded systems, the dimension of the states x_k and inputs u_k are relatively small (max 12 and 4, respectively), with horizons N up to 100.

Results Results for this benchmark set are shown in Figure 4.3. Clarabel is the fastest solver overall, and is the only solver tested with a 100% success rate in solving problems to full accuracy.

Figure 4.3: Performance profiles for the optimal control problem set



(a) Relative performance profile



(b) Absolute performance profile

		ClarabelRs	Clarabel	ECOS	Gurobi	HiGHS	Hypatia	Mosek
Shifted GM	Full Acc.	1.0	1.06	197.57	70.48	185.45	54.95	511.01
	Low Acc.	1.0	1.06	14.97	60.07	185.45	37.36	8.26
Failure Rate (%)	Full Acc.	0.0	0.0	23.6	12.5	19.4	2.8	37.5
	Low Acc.	0.0	0.0	2.8	11.1	19.4	0.0	0.0

(c) Benchmark timings as shifted geometric mean and failure rates

4.6.2 Benchmark problems with linear objectives

In this section we present benchmark results for optimization problems *without* quadratic objective terms, i.e. with $P = 0$ in $(\mathcal{P})$. We again consider example problems taken or generated from standard open-source problem collections and covering a wide range of problem dimensions. Our test set covers cases with both constraints on the positive orthant (i.e. linear programs), as well as second-order and exponential cone programs.

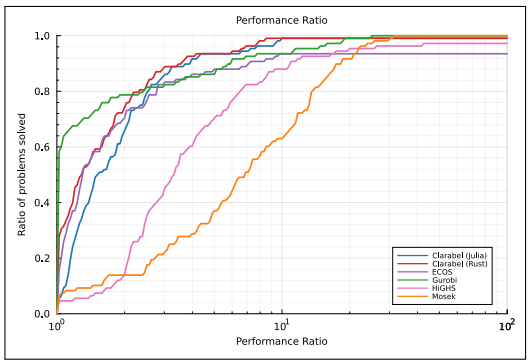
NETLIB LP problems

We first consider LP problems taken from the NETLIB collection, a standard collection of benchmark LPs. Our benchmark test set includes 117 feasible and 29 infeasible test cases, representing all NETLIB LP problem instances with source files not exceeding 2.5MB.

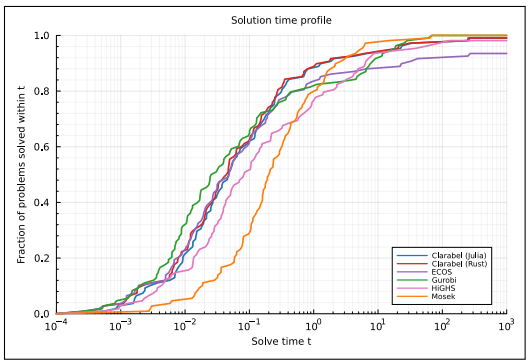
Results for the feasible and infeasible test sets are shown in Figures 4.4 and 4.5, respectively. For these cases Clarabel has performance broadly similar to both Mosek and Gurobi for the feasible set, and somewhat slower for the infeasible set. This result is to be expected since most of the potential performance advantage of

our method arises from improved handling of quadratic objectives, but illustrates that our implementation is, in the LP case, still broadly comparable.

Figure 4.4: Performance profiles for the NETLIB Feasible LP problem set



(a) Relative performance profile

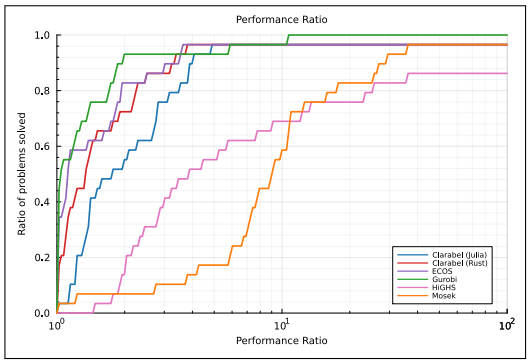


(b) Absolute performance profile

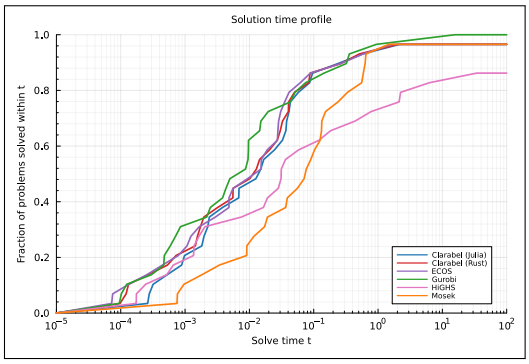
		ClarabelRs	Clarabel	ECOS	Gurobi	HiGHS	Mosek
Shifted GM	Full Acc.	1.0	1.04	2.13	1.3	1.79	1.2
	Low Acc.	1.0	1.04	1.08	1.3	1.79	1.2
Failure Rate (%)	Full Acc.	0.9	0.9	6.5	0.0	1.9	0.0
	Low Acc.	0.9	0.9	0.9	0.0	1.9	0.0

(c) Benchmark timings as shifted geometric mean and failure rates

Figure 4.5: Performance profiles for the NETLIB Infeasible LP problem set



(a) Relative performance profile



(b) Absolute performance profile

		ClarabelRs	Clarabel	ECOS	Gurobi	HiGHS	Mosek
Shifted GM	Full Acc.	1.89	1.92	1.85	1.0	12.19	2.49
	Low Acc.	1.89	1.92	1.85	1.0	12.19	2.49
Failure Rate (%)	Full Acc.	3.4	3.4	3.4	0.0	13.8	3.4
	Low Acc.	3.4	3.4	3.4	0.0	13.8	3.4

(c) Benchmark timings as shifted geometric mean and failure rates

Optimal power flow

We next consider a variety of optimal power flow problems based on power networks from IEEE PLS PGLib-OPF benchmark library [111] and constructed using the `PowerModels.jl` benchmark test framework [139]. This framework allows for the generation of optimal power flow problems with various modelling assumptions and convex relaxations applied [140, 141]. We consider in particular linear programming problems based on linearized (i.e. direct current (DC)) power flow models [142], and SOCPs arising from second-order cone relaxations of AC models [143]. Results from these benchmarks are shown in Figures 4.6 and 4.7, respectively.

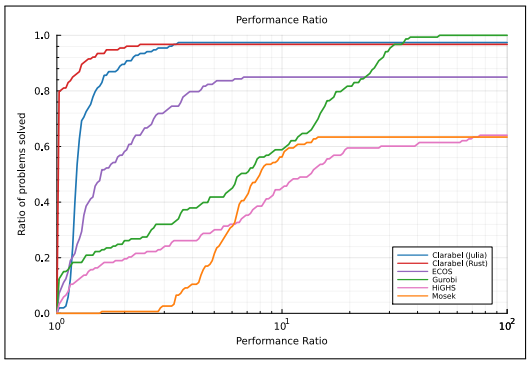
For both of these test sets Clarabel outperforms the other solvers tested, albeit with a slightly higher failure rate for the LP benchmark tests relative to Gurobi (i.e. 3.3% vs 0% at full accuracy). For the SOCP benchmark tests in particular our success rate is substantially better than the other solvers tested. All solvers in our benchmark group struggled to some extent in solving large scale SOCPs to full accuracy. Our Rust implementation is able to solve to at least its reduced accuracy level for more than 99% of test cases though, a success rate considerably higher than ECOS or Mosek, both of which failed even at reduced accuracy on more than 50% of cases.

We note also that our success rates differ slightly between our Julia and Rust implementations, even though the implementation of our algorithm is (nearly) identical between the cases. We believe that this difference is attributable to minor differences in compiled code vectorisations and optimizations, which in some very difficult problems lead to slight differences in behaviour at high accuracy.

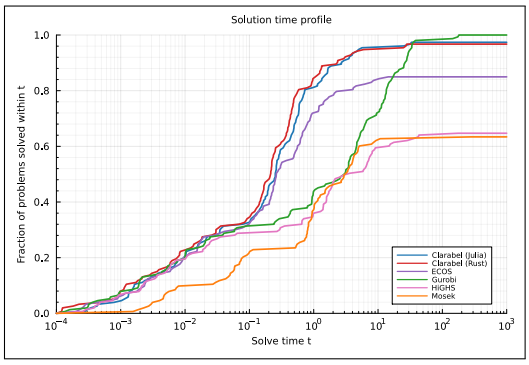
CBLIB exponential cone problems

Finally, we consider a collection of 39 exponential cone programs from the CBLIB benchmark collection [144] in order to test performance on nonsymmetric cone programs. For these problems our implementation has broadly similar performance

Figure 4.6: Performance profiles for the LP Optimal Power Flow problem set



(a) Relative performance profile

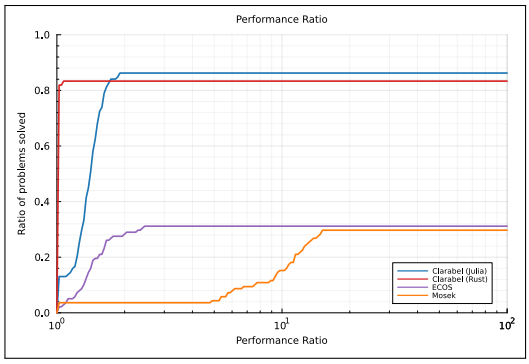


(b) Absolute performance profile

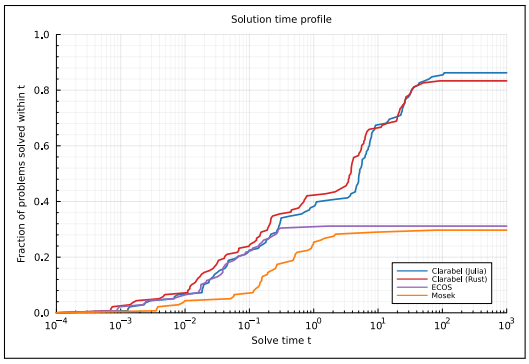
		ClarabelRs	Clarabel	ECOS	Gurobi	HiGHS	Mosek
Shifted GM	Full Acc.	1.0	1.04	3.27	4.48	17.73	17.58
	Low Acc.	1.0	1.07	2.89	5.57	22.04	21.85
Failure Rate (%)	Full Acc.	3.3	2.6	15.0	0.0	35.3	36.6
	Low Acc.	1.3	0.7	9.8	0.0	35.3	36.6

(c) Benchmark timings as shifted geometric mean and failure rates

Figure 4.7: Performance profiles for the SOCP Optimal Power Flow problem set



(a) Relative performance profile



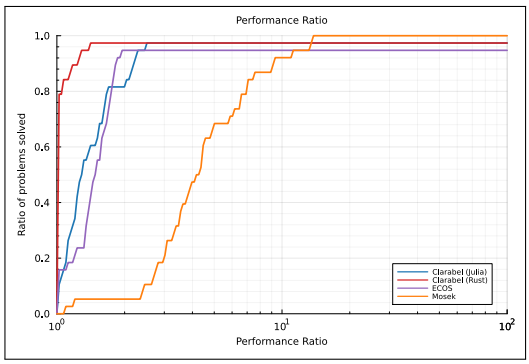
(b) Absolute performance profile

		ClarabelRs	Clarabel	ECOS	Mosek
Shifted GM	Full Acc.	1.0	1.07	8.25	10.02
	Low Acc.	1.0	1.33	8.62	20.48
Failure Rate (%)	Full Acc.	16.7	13.8	68.8	70.3
	Low Acc.	0.7	2.2	55.8	70.3

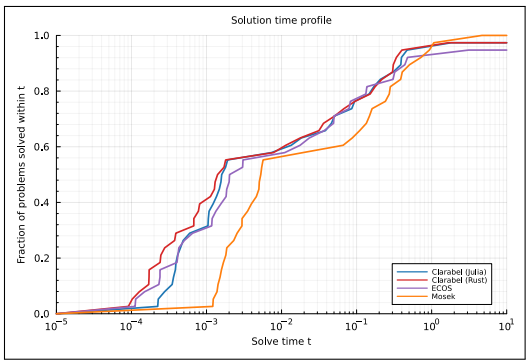
(c) Benchmark timings as shifted geometric mean and failure rates

to Mosek, with slightly faster solve times in the Rust implementation despite a generally higher iteration count.

Figure 4.8: Performance profiles for the CBLIB Exponential Cone problem set



(a) Relative performance profile



(b) Absolute performance profile

		ClarabelRs	Clarabel	ECOS	Mosek
Shifted GM	Full Acc.	1.45	1.49	2.68	1.0
	Low Acc.	1.0	1.08	3.28	2.14
Failure Rate (%)	Full Acc.	2.6	2.6	5.3	0.0
	Low Acc.	0.0	0.0	2.6	0.0

(c) Benchmark timings as shifted geometric mean and failure rates

5

An efficient implementation of interior-point methods for a class of nonsymmetric cones

Contents

5.1 Introduction

The great majority of research on efficient IPMs has focussed on problems with conic constraints that are *self-scaled* (homogeneous and self-dual), e.g. nonnegative, second order and positive semidefinite cones. Cones that do *not* possess the self-scaled property are called *nonsymmetric* and are a topic of more recent interest interior-point methods [15]. The extension of the homogeneous self-dual model for the nonsymmetric case was proposed in [48] with a variant implemented in ECOS [93], and then a complementary proof for the convergence to ϵ-optimality of complexity $\mathcal{O}(\sqrt{\nu}\ln(1/\epsilon))$ was provided in [145], where ν is the degree of a cone. The exponential cone and the power cone are the two most commonly studied, and are supported in the commercial solver Mosek [40] and some open-source solvers like ECOS [45] and Alfonso [146]. It has also been shown that some existing conic optimization problems can be solved more efficiently by exploiting their special structure through the lens of nonsymmetric conic optimization, like sparse SDPs [16] and sum of squares (SOS) programs [17]. A Matlab-based DDS [34] and Julia-based Hypatia [35] solvers introduce some nonsymmetric cones that solve many optimization problems more efficiently than their extended formulations based on 3-dimensional exponential cones or power cones.

However, nonsymmetric cones do not possess the self-scaled property and so cannot exploit NT scaling points as is possible in the symmetric case [13]. Instead a nonsymmetric strategy, where the scaling point is chosen to be the primal (dual) iterate in each iteration, is commonly used for nonsymmetric cones in solvers like ECOS and Hypatia. Recently, a primal-dual scaling algorithm motivated by [106] was proposed in [92] and implemented in Mosek, which claims to be the fastest solver for exponential and power cones at present. Both the nonsymmetric scaling and the primal-dual symmetric scaling require conjugate gradients of conic barrier functions, which generally do not have closed-form representations for nonsymmetric cones. Numerical methods for computing conjugate gradients of some nonsymmetric cones are detailed in [105].

The 3-*dimensional power cone* is a powerful tool to model various cones such as the p-norm cone, the power mean cone and the generalized power cone [91]. The thesis of Chares [91] also conjectured an $(n+1)$-self-concordant barrier for the $(n+1)$-dimensional power cone $\mathcal{K}_\alpha^{(n)} := \left\{(x,z) \in \mathbb{R}^n_+ \times \mathbb{R} : \prod_{i=1}^n x_i^{\alpha_i} \geq |z|\right\}$, where $\alpha \in \mathbb{R}^n_\geq$ and $\sum_{i=1}^n \alpha_i = 1$. This was finally validated in the context of a more general (d_1, d_2)-generalized power cone $\mathcal{K}_{\text{gpow}(\alpha,d_1,d_2)}$ [147].

5.2 Augmented sparsity

We will consider the primal-dual pair $(\mathcal{P})$ and $(\mathcal{D})$ with P set to 0 and $\mathcal{K}$ assumed nonsymmetric throughout this chapter. For a class of nonsymmetric cones, the scaling matrix H appears as the lower right-hand block in (4.19) is the sum of a sparse matrix plus a few low-rank dense terms, and satisfies a certain quasidefiniteness condition.

Definition 5.2.1. Suppose the scaling matrix H is of the form

$$H := D + \sum_{i=1}^{n_1} u_i u_i^\top - \sum_{j=1}^{n_2} v_i v_i^\top, \tag{5.1}$$

where $u_i, v_i \in \mathbb{R}^n, D \in \mathbb{S}^{n\times n}$ is sparse and $n_1, n_2 \ll n$. We will say that H is *augmented-sparse* if $D - \sum_{j=1}^{n_2} v_i v_i^\top \succ 0$. The corresponding *augmented sparse matrix* is

$$H_{\text{aug}} := \begin{bmatrix} D & V & U \\ V^\top & I_{n_2} & \\ U^\top & & -I_{n_1} \end{bmatrix}$$

with $V = [v_1, \ldots, v_{n_2}], U = [u_1, \ldots, u_{n_1}]$.

Lemma 5.2.1. *If H is augmented-sparse then H_{aug} is quasidefinite.*

Proof. A sufficient condition is for the upper left 2×2 block of H to be positive definite [148]. This follows immediately since the Schur complement $D - VV^T$ is positive definite by assumption. □

If H is augmented-sparse, we can solve (4.19) via an equivalent linear system. For example, the last row of the left equation in (4.19), i.e. $Ax - Hz_1 = d_s - d_z$ becomes

$$\begin{bmatrix} A \\ 0 \\ 0 \end{bmatrix} \Delta x - \underbrace{\begin{bmatrix} D & V & U \\ V^\top & I_{n_2} & \\ U^\top & & -I_{n_1} \end{bmatrix}}_{H_{\text{aug}}} \begin{bmatrix} \Delta z_1 \\ t_v \\ t_u \end{bmatrix} = \begin{bmatrix} d_s - d_z \\ 0 \\ 0 \end{bmatrix} \tag{5.2}$$

with additional variables $t_u \in \mathbb{R}^{n_1}, t_v \in \mathbb{R}^{n_2}$, which is both larger and sparser than the equivalent term in (4.19). The same applies to the last row of the right equation in (4.19). For (5.2), the number of nonzero entries in H_{aug} is $[\text{nnz}(D) + n(n_1 + n_2 + 2)]$, which is much smaller than the number of entries in a dense Hessian form if $n_1, n_2 \ll n$. We therefore expect that an LDL factorization of H_{aug} will have significantly sparser factors than would be obtained by direct factorization of H. Such a property has been exploited for SOCPs in ECOS under the NT (primal-dual) scaling strategy [45]. We will show how this approached can also be utilized in the nonsymmetric strategy for nonsymmetric cones.

5.3 Sparse Hessian decompositions for nonsymmetric cones

In this section we describe two nonsymmetric cones that can be shown to have *augmented-sparse* structure in the sense of Definition 5.2.1. We introduce fundamental definitions and barrier functions for these nonsymmetric cones in Section 5.3.1, and then detail the underlying augmented-sparse structures in Section 5.3.2.

5.3.1 Nonsymmetric power cones

We consider two power cones that can exploit sparse structure: generalized power cones and power mean cones. They are defined as follows:

Definition 5.3.1 (Generalized Power Cone)**.** The *generalized power cone* is defined as

$$\mathcal{C}_{\text{gpow}(\alpha,d_1,d_2)} = \left\{ (u, w) \in \mathbb{R}^{d_1}_+ \times \mathbb{R}^{d_2} : \prod_{i \in [\![d_1]\!]} u_i^{\alpha_i} \geq \|w\| \right\}, \tag{5.3a}$$

$\alpha \in \mathbb{R}^{d_1}_{++}$ such that $\sum_{i\in[\![d_1]\!]} \alpha_i = 1$. Its dual cone is

$$\mathcal{C}^*_{\text{gpow}(\alpha,d_1,d_2)} = \left\{(u,w) \in \mathbb{R}^{d_1}_+ \times \mathbb{R}^{d_2} : \prod_{i\in[\![d_1]\!]} \left(\frac{u_i}{\alpha_i}\right)^{\alpha_i} \geq \|w\|\right\}. \tag{5.3b}$$

Note that the dual generalized power cone is a linear transformation of the primal generalized power cone.

Definition 5.3.2 (Power Mean Cone)**.** The *power mean cone* is parametrized by $\alpha \in \mathbb{R}^{d_1}_{++}$ such that $\sum_{i\in[\![d]\!]} \alpha_i = 1$ and is defined as

$$\mathcal{C}_{\text{powm}(\alpha,d)} = \left\{(u,w) \in \mathbb{R}^d_+ \times \mathbb{R} : \prod_{i\in[\![d]\!]} u_i^{\alpha_i} \geq w\right\}. \tag{5.4a}$$

Its dual cone is

$$\mathcal{C}^*_{\text{powm}(\alpha,d)} = \left\{(u,w) \in \mathbb{R}^d_+ \times \mathbb{R}_- : \prod_{i\in[\![d]\!]} \left(\frac{u_i}{\alpha_i}\right)^{\alpha_i} \geq -w\right\}. \tag{5.4b}$$

Note that the geometric mean cone [35] is a special case of the power mean cone where $\alpha_1 = \cdots = \alpha_d = \frac{1}{d}$.

A generalized power cone reduces to a second-order cone when $d_1 = 1$ while it looks similar to a power mean cone when $d_2 = 1$. This motivates us to consider whether we can exploit sparsity of these nonsymmetric cones using an approach similar to the one that was described for second-order cones in [45]. The answer is *yes* if we choose the nonsymmetric scaling strategy for these cones in a IPM.

We therefore choose the nonsymmetric scaling strategy in our IPM, and will formulate our problems such that the dual cone $\mathcal{K}^*$ in our general primal-dual problem pair $(\mathcal{P})$–$(\mathcal{D})$ is either $\mathcal{C}_{\text{gpow}(\alpha,d_1,d_2)}$ or $\mathcal{C}_{\text{powm}(\alpha,d)}$. We consequently require ν-LHSCB conjugate barrier functions for $\mathcal{C}_{\text{gpow}(\alpha,d_1,d_2)}$ and $\mathcal{C}_{\text{powm}(\alpha,d)}$, which have been defined already in [34, 35]:

Definition 5.3.3. For generalized power cones and power mean cones:

1. The function

$$f_{\text{gpow}}(u,w) = -\ln\left(\prod_{i\in[\![d_1]\!]} u_i^{2\alpha_i} - \|w\|^2\right) - \sum_{i\in[\![d_1]\!]} (1-\alpha_i)\ln(u_i) \tag{5.5}$$

is a (d_1+1)-LHSCB function of $\mathcal{C}_{\text{gpow }(\alpha,d_1,d_2)}$ where $\alpha, u \in \mathbb{R}^{d_1}_{++}, w \in \mathbb{R}^{d_2}$.

2. The function

$$f_{\text{powm}}(u,w) = -\ln\left(\prod_{i\in[\![d]\!]} u_i^{\alpha_i} - w\right) - \sum_{i\in[\![d]\!]} \ln(u_i) \tag{5.6}$$

is a $(d+1)$-LHSCB function of $\mathcal{C}_{\text{powm}(\alpha,d)}$ where $\alpha, u \in \mathbb{R}^{d}_{++}, w \in \mathbb{R}$.

5.3.2 Sparsity exploitation

For the nonsymmetric scaling strategy at iteration k, the scaling matrix H is set to $H^k = \mu^k H^*(z^k)$, where $\mu^k := \langle s^k, z^k\rangle/\nu > 0$ is the centering parameter and $H^*(z^k)$ is the Hessian of a barrier function at $z^k \in \mathcal{K}^*$ with degree ν. Hence, *we can exploit the augmented-sparse structure of the scaling matrix H as long as $H^*(z^k)$ is augmented-sparse.* For the remainder of this section, we assume that $\mathcal{K}^*$ is either $\mathcal{C}_{\text{gpow }(\alpha,d_1,d_2)}$ or $\mathcal{C}_{\text{powm}(\alpha,d)}$, i.e. $f^*(z) = f_{\text{gpow}}(\cdot)$ or $f^*(z) = f_{\text{powm}}(\cdot)$, and will show that the Hessians $H^*(z)$ of each of the barrier functions defined in Definition 5.3.3 are augmented-sparse, and thus so are the related scaling matrices H.

Generalized power cone

We start by establishing the augmented-sparse property for generalized power cones:

Theorem 5.3.1. *The Hessian of the dual barrier function $f^*(z) = f_{gpow}(u,w)$ defined in* (5.5) *for the generalized power cone satisfies Definition 5.2.1 with $n_1 = 1, n_2 = 2$, i.e.*

$$H^*(z) = D + pp^\top - qq^\top - rr^\top,$$

where $z =: (u,w)$ and $D - qq^\top - rr^\top \succ 0$. The parameters D, p, q, r are given by

$$D = \left[\begin{array}{c|c} \underbrace{\begin{array}{c}\ddots \\ \frac{\tau_i\varphi}{\zeta u_i} + \frac{1-\alpha_i}{u_i^2} \\ \qquad\qquad \ddots\end{array}}_{D_1} & \\ \hline & \underbrace{\frac{2}{\zeta}\cdot I_{d_2}}_{D_2} \end{array}\right], \; p = \begin{bmatrix} p_0 \cdot \frac{\tau}{\zeta} \\ p_1 \cdot \frac{w}{\zeta} \end{bmatrix}, \; q = \begin{bmatrix} q_0 \cdot \frac{\tau}{\zeta} \\ 0 \end{bmatrix}, \; r = \begin{bmatrix} 0 \\ r_1 \cdot \frac{w}{\zeta} \end{bmatrix}, \tag{5.7a}$$

with

$$p_0 = \sqrt{\frac{\varphi(\varphi + \|w\|^2)}{2}}, \qquad p_1 = -2\sqrt{\frac{2\varphi}{\varphi + \|w\|^2}}, \tag{5.7b}$$
$$q_0 = \sqrt{\frac{\zeta\varphi}{2}}, \qquad\qquad r_1 = 2\sqrt{\frac{\zeta}{\varphi + \|w\|^2}},$$

where $\varphi = \prod_{i\in[\![d_1]\!]} u_i^{2\alpha_i}, \tau_i = \frac{2\alpha_i}{u_i}, \forall i \in [\![d_1]\!]$, *and* $\zeta = \prod_{i\in[\![d_1]\!]} u_i^{2\alpha_i} - \|w\|^2$.

Proof of Theorem 5.3.1. The gradient and Hessian for $f_{\text{gpow}}(u, w)$ (5.5) are given by

$$\begin{aligned}
\nabla_{u_i} f &= -\frac{\tau_i\varphi}{\zeta} - \frac{1-\alpha_i}{u_i}, \quad \forall i \in [\![d_1]\!], \\
\nabla_{w_i} f &= \frac{2w_i}{\zeta}, \quad \forall i \in [d_2], \\
\nabla^2_{u_i,u_j} f &= \frac{\tau_i\tau_j\varphi}{\zeta}\left(\frac{\varphi}{\zeta} - 1\right) + \delta_{i,j}\left(\frac{\tau_i\varphi}{\zeta u_i} + \frac{1-\alpha_i}{u_i^2}\right), \quad \forall i, j \in [\![d_1]\!], \\
\nabla^2_{u_i,w_j} f &= -\frac{2\tau_i\varphi w_j}{\zeta^2}, \quad \forall i \in [\![d_1]\!], \forall j \in [d_2], \\
\nabla^2_{w_i,w_j} f &= \frac{4w_i w_j}{\zeta^2} + \delta_{i,j}\frac{2}{\zeta}, \quad \forall i, j \in [d_2].
\end{aligned}$$

p_0, p_1, q_0, r_1 should satisfy the conditions

$$p_0^2 - q_0^2 = \varphi(\varphi - \zeta) = \varphi\|w\|^2, \quad p_0 p_1 = -2\varphi, \quad p_1^2 - r_1^2 = 4,$$

to match coefficients in the Hessian. Given the parameter choices in (5.7), we find $D - qq^\top - rr^\top$ is 2×2 block diagonal and the condition $D - qq^\top - rr^\top \succ 0$ becomes

$$D_1 - q_0^2\left(\frac{\tau}{\zeta}\right)\cdot\left(\frac{\tau}{\zeta}\right)^\top \succ 0 \text{ and } D_2 - r_1^2\left(\frac{w}{\zeta}\right)\cdot\left(\frac{w}{\zeta}\right)^\top \succ 0,$$

which is equivalent to

$$1 - q_0^2\left(\frac{\tau}{\zeta}\right)^\top D_1^{-1}\left(\frac{\tau}{\zeta}\right) = 1 - \sum_{i\in[\![d_1]\!]} \frac{q_0^2\cdot\frac{\tau_i^2}{\zeta^2}}{\frac{\tau_i\varphi}{\zeta u_i} + \frac{1-\alpha_i}{u_i^2}} = 1 - \sum_{i\in[\![d_1]\!]} \frac{4\alpha_i^2 q_0^2}{\zeta(2\alpha_i\varphi + (1-\alpha_i)\zeta)} > 0, \tag{5.8a}$$

and

$$1 - r_1^2\left(\frac{w}{\zeta}\right)^\top D_2^{-1}\left(\frac{w}{\zeta}\right) = 1 - r_1^2\frac{\|w\|^2}{\zeta^2}\cdot\frac{\zeta}{2} > 0, \tag{5.8b}$$

by Schur complement. If we set $q_0 = \sqrt{\frac{\zeta\varphi}{2}}, r_1 = 2\sqrt{\frac{\zeta}{\varphi+\|w\|^2}}$, (5.8a) is satisfied by

$$1 - \sum_{i\in[\![d_1]\!]} \frac{4\alpha_i^2 q_0^2}{\zeta(2\alpha_i\varphi + (1-\alpha_i)\zeta)} > 1 - \sum_{i\in[\![d_1]\!]} \frac{4\alpha_i^2 q_0^2}{2\zeta\alpha_i\varphi} = 1 - \frac{2q_0^2}{\zeta\varphi} = 0,$$

and (5.8b) is satisfied by

$$1 - r_1^2 \frac{\|w\|^2}{\zeta^2} \cdot \frac{\zeta}{2} = 1 - \frac{2\|w\|^2}{\varphi + \|w\|^2} = \frac{2\zeta}{\varphi + \|w\|^2} > 0.$$

Hence, $D - qq^\top - rr^\top \succ 0$. □

Since the scaling matrix $H = \mu H^*(z)$ is augmented-sparse, we can employ an expanded sparse linear system

$$\begin{bmatrix} A \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta x - \mu \underbrace{\begin{bmatrix} D & q & r & p \\ q^\top & 1 & 0 & 0 \\ r^\top & 0 & 1 & 0 \\ p^\top & 0 & 0 & -1 \end{bmatrix}}_{K_{\text{aug}}} \begin{bmatrix} \Delta z_1 \\ t_q \\ t_r \\ t_p \end{bmatrix} = \begin{bmatrix} d_s - d_z \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{5.9}$$

in our interior-point step equation (4.19), where K_{aug} is quasidefinite and strongly factorizable. Instead of (5.9), we solve the linear system

$$\begin{bmatrix} A \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta x - \begin{bmatrix} \mu D & \sqrt{\mu}q & \sqrt{\mu}r & \sqrt{\mu}p \\ \sqrt{\mu}q^\top & 1 & 0 & 0 \\ \sqrt{\mu}r^\top & 0 & 1 & 0 \\ \sqrt{\mu}p^\top & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \Delta z_1 \\ t'_q \\ t'_r \\ t'_p \end{bmatrix} = \begin{bmatrix} d_s - d_z \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{5.10}$$

which yields the same $\Delta x, \Delta z_1$ as (5.9). During direct factorization we find that the pivoting for (5.10) is more numerically stable than for (5.9) since the magnitude of diagonal terms of K_{aug} increases from μ to 1 and the ratio of off-diagonal to diagonal terms also decreases, e.g. q to $\sqrt{\mu}q$. In addition, the memory requirement for the rank-3 update of generalized power cones is equivalent to a rank-2 update due to the presence of zeros in q, r.

Power mean cone

We next establish the augmented-sparse property for the power mean cone:

Theorem 5.3.2. *The Hessian of the dual barrier function* $f^*(z) = f_{powm}(u, w)$ *defined in* (5.6) *for the power mean cone satisfies Definition 5.2.1 with* $n_1 = 1, n_2 = 1$*, i.e.*

$$H^*(z) = D + pp^\top - qq^\top,$$

where $z =: (u, w)$ *and* $D - qq^\top \succeq 0$*. The parameters* D, p, q, r *are given by*

$$D = \left[\begin{array}{ccc|c} \ddots & & & \\ & \frac{\tau_i\varphi}{u_i} + \frac{1}{u_i^2} & & \\ & & \ddots & \\ \hline & & & 0 \end{array}\right], \; p = \begin{bmatrix} p_0 \cdot \tau \\ p_1 \cdot \frac{1}{\zeta} \end{bmatrix}, \; q = \begin{bmatrix} q_0 \cdot \tau \\ 0 \end{bmatrix}, \tag{5.11a}$$

with

$$p_0 = \varphi, \quad p_1 = -1, \quad q_0 = \sqrt{\zeta\varphi}, \tag{5.11b}$$

where we define $\varphi = \prod_{i\in[\![d]\!]} u_i^{\alpha_i}, \zeta = \varphi - w$ *and* $\tau \in \mathbb{R}^d$ *with* $\tau_i = \frac{\alpha_i}{u_i\zeta}, \forall i \in [\![d]\!]$*.*

Proof of Theorem 5.3.2. The gradient and Hessian of (5.6) are

$$\begin{aligned} \nabla_w f &= \frac{1}{\zeta} \\ \nabla_{u_i} f &= -\varphi\tau_i - \frac{1}{u_i}, \\ \nabla^2_{w,w} f &= \frac{1}{\zeta^2} \\ \nabla^2_{w,u_i} f &= -\frac{\varphi\tau_i}{\zeta} \\ \nabla^2_{u_i,u_j} f &= \varphi\tau_i\tau_j w + \delta(i,j)\left(\frac{\varphi\tau_i}{u_i} + \frac{1}{u_i^2}\right), \end{aligned}$$

According to the Schur complement, the condition $D - qq^\top \succeq 0$ is equivalent to

$$\begin{aligned} & 1 - q_0^2\tau^\top\left[\mathrm{Diag}\left(\frac{\tau_i\varphi}{u_i} + \frac{1}{u_i^2}\right)\right]^{-1}\tau \succeq 0. \\ \iff & 1 - \sum_{i\in[\![d]\!]} \frac{\varphi\zeta\tau_i^2}{\frac{\tau_i\varphi}{u_i} + \frac{1}{u_i^2}} = 1 - \sum_{i\in[\![d]\!]} \frac{\alpha_i^2}{\alpha_i + \frac{\zeta}{\varphi}} > 0. \end{aligned}$$

□

Note that the Hessian of the barrier function for the power mean cone will become augmented-sparse after adding regularization into diagonal terms, which is commonly

used in the factorization step [149]. Note that the augmented-sparse structure of power mean cones is similar to the case of generalized power cones in Theorem 5.3.2, but with r to zeros. We can therefore use an expanded sparse linear system as (5.9) with a numerically stable variant similar to (5.10) for interior-point step equations for the power mean cone.

5.4 An IPM for nonsymmetric conic optimization

We implement an IPM algorithm akin to that described in Section 4.4.8, with the following necessary modifications:

Initial point: Initialization of symmetric cones and 3-dimensional nonsymmetric cones, i.e. power and exponential cones, follows the Mosek setting as discussed in [92], where $x^0 = 0, \kappa^0 = \tau^0 = 1$ and s^0, z^0 are set to

$$s^0 = z^0 = -g^*(z^0), \tag{5.12}$$

which are on the central path with $\mu^0 = 1$. This strategy holds for $\mathcal{C}_{\text{gpow}(\alpha,d_1,d_2)}$, and we initialize s^0, z^0 by

$$s^0 = z^0 = \left(\left(\sqrt{1+\alpha_i}\right)_{i\in[\![d_1]\!]}, 0_{d_2}\right).$$

For $\mathcal{C}_{\text{powm}(\alpha,d)}$, we can not easily find s^0, z^0 satisfying (5.12), so we initialize s^0, z^0 as in Hypatia [35], where s^0, z^0 satisfy $s^0 = -g^*(z^0)$. Note that $\mu^0 = 1$ still holds in this setting.

Scaling matrix: Instead of primal-dual scaling strategy, we choose the nonsymmetric scaling (4.16) for H^k to exploit the augmented sparsity from the Hessians of barrier functions.

Higher-order correction: We use the same 3rd-order correction (4.23) for cones in this chapter. Although the 3rd-order correction was proposed for use with the primal-dual symmetric scaling [92], we find it also works effectively for other nonsymmetric cones when we keep the iterates close to the central path. The computation of 3rd-order derivatives and the inverse of Hessians are detailed in Appendix A.6.

5.5 Experiments

In our experiments we choose two examples to show the effectiveness of our sparse implementation for generalized power cones; the maximum likelihood estimation of a convex distribution [150] and the maximum volume hypercube problems [151]. We test these examples with 4 different solver configurations:

1. `GenPow` is the proposed sparse implementation of generalized power cones;
2. `Clarabel-Pow` uses Clarabel but transforms the generalized power cone to a product of 3-dimensional power cones as formulated in [40, 91];
3. `Hypatia` uses the Hypatia solver, which supports the generalized power cone constraints directly;
4. `Mosek` denotes the use of Mosek with transforming the generalized power cone to a product of 3-dimensional power cones.

Our code is publicly available[1]. The convergence tolerance is set to $\epsilon = 10^{-7}$ for all solvers in all tests. In order to compare performance of our method against solvers that do not directly support generalized power cone constraints, we apply a standard transformation to reformulate the constraint for those solvers into a collection of 3-dimensional power cone constraints. Following [40], we can rewrite a generalized power cone $(x, t) \in \mathcal{C}_{\text{gpow}(\alpha,n,1)}$ constraint as a collection of constraints over 3-dimensional power cones

$$\begin{aligned} & x_1^{1-p_2} x_2^{p_2} \geq |z_3|\,, \\ & z_3^{1-p_3} x_3^{p_3} \geq |z_4|\,, \\ & \cdots \\ & z_{n-1}^{1-p_{n-1}} x_{n-1}^{p_{n-1}} \geq |z_n|\,, \\ & z_n^{1-p_n} x_n^{p_n} \geq |t|, \end{aligned} \tag{5.13}$$

where $p_i = \alpha_i / (\sum_{1 \leq j \leq i} \alpha_j)$. This is the form used by `Clarabel-Pow` and `Mosek`.

[1] `https://github.com/yuwenchen95/Clarabel-genpowcone.jl.git`

We also test the sparse implementation of the power mean cone, called `PowMean`, and compare it with `Hypatia` on these same two examples, since it produces the same results as the generalized power cone for both cases.

5.5.1 Maximum likelihood estimator of a density function

Given an ordered sample $y_1 < y_2 < \ldots < y_n$ of n outcomes of an unknown distribution with a convex density function $g : \mathbb{R}_+ \to \mathbb{R}_+$, we can form an estimate of g using a piecewise linear function estimator $\hat{g} : [y_1, y_n] \to \mathbb{R}_+$ with break points at $(y_i, x_i), i = 1, \ldots, n$, where the variables $x_i > 0$ are estimators for $g(y_i)$ [150]. The slope of the i-th linear segment of $\hat{g}$ is

$$\frac{x_{i+1} - x_i}{y_{i+1} - y_i}$$

and the convexity requirement leads to the non-decreasing slope constraints

$$\frac{x_{i+1} - x_i}{y_{i+1} - y_i} \leq \frac{x_{i+2} - x_{i+1}}{y_{i+2} - y_{i+1}}, i = 1, \ldots, n-2.$$

The discrete density function $\hat{g}$ must sum to 1, i.e.

$$\sum_{i=1}^{n-1} (y_{i+1} - y_i) \left(\frac{x_{i+1} + x_i}{2} \right) = 1.$$

The maximum likelihood estimation can then be formulated as a conic optimization problem

$$\begin{aligned} \max_{x,t} \quad & t \\ s.t. \quad & \frac{x_{i+1} - x_i}{y_{i+1} - y_i} - \frac{x_{i+2} - x_{i+1}}{y_{i+2} - y_{i+1}} \leq 0, \quad i = 1, \ldots, n-2, \\ & \sum_{i=1}^{n-1} (y_{i+1} - y_i) \left(\frac{x_{i+1} + x_i}{2} \right) = 1, \\ & (x, t) \in \mathcal{K}, \; x \geq 0, \end{aligned} \tag{5.14}$$

where $\mathcal{K}$ can be either a generalized power cone or a power mean cone. We test implementation of `GenPow` and compare them with the counterparts implemented in `Hypatia`, `Clarabel-Pow` and `Mosek`, which are tested by replacing the constraint $(x, t) \in \mathcal{K}$ with the equivalent reformulation shown in (5.13). We take $(\alpha^{-1}x, t) \in \mathcal{C}^*_{\text{gpow}}$ as an equivalent constraint in our proposed implementation. This enables us to

take advantage of the sparsity exploitation methods of Section-5.3 with higher-order correction, treating $\mathcal{C}_{\text{gpow}}$ as the dual cone.

Computational results are given in Table 5.1 and Table 5.2 which show the total computational time and the number of iterations for different dimensions n. We generate sample points from an exponential distribution and prune to ensure that the minimum distance between samples is at least 10^{-2} to avoid clustering.

Table 5.1: Tests with the generalized power cone

	GenPow		Hypatia (GenPow)		Clarabel-Pow		Mosek	
n	time	iter	time	iter	time	iter	time	iter
465	0.042	41	1.037	57	0.103	45	0.078	24
910	0.088	49	8.175	137	0.217	52	0.297	44
1370	0.223	87	19.144	138	0.390	61	0.625	63

Table 5.2: Tests with the power mean cone

	PowMean		Hypatia (PowMean)	
n	time	iter	time	iter
465	0.083	62	0.905	45
910	0.169	60	5.304	84
1370	0.274	80	15.274	111

Table 5.1 shows that our sparse LDL implementation for generalized power cones performs better than other methods on problems of moderate size. Although the total iteration number of `GenPow` is larger than the implementation of `Clarabel-Pow` or `Mosek`, it takes less time overall for convergence due to improved linear solve efficiency enabled by the augmented sparse decomposition we proposed in Section 5.3.2.

Table 5.2 summarizes results for the same problem, but choosing $\mathcal{K}$ to be a power mean cone in (5.14). Since `Mosek` doesn't support power mean cones directly we compare our sparse implementation of power mean cones, named `PowMean`, only with Hypatia. We vary the dimension n as for the test of generalized power cones. Table 5.2 shows that our sparse implementation for power mean cones is significantly faster than Hypatia which is based on the Cholesky decomposition on the reduced

normal system, both in total time and the averaged time per iteration. Although the results of our power mean cones are slower than our sparse implementation of generalized power cones, it still performs better than other solvers on the discrete maximum likelihood problem.

5.5.2 Maximum volume of a hypercube

The problem of finding the maximum volume of a hypercube that fits inside the intersection of 1-norm and ∞-norm balls is given as follows,

$$\begin{aligned} & \max \quad t \\ \text{s.t.} \quad & (x,t) \in \mathcal{K}_{\text{gpow }(\alpha,n,1)}, \alpha = \left[\frac{1}{n}, \ldots, \frac{1}{n}\right], \\ & Ax \in \mathcal{B}_1(\gamma_1),\ Ax \in \mathcal{B}_\infty(\gamma_2), \end{aligned} \tag{5.15}$$

where $\mathcal{B}_1(\gamma_1) := \{x \mid \|x\|_1 \leq \gamma_1\}$ is a 1-norm ball, $\mathcal{B}_\infty(\gamma_2) := \{x \mid \|x\|_\infty \leq \gamma_2\}$ is an ∞-norm ball and γ_1, γ_2 are given constants. Both norm constraints are easily converted into a collection of linear inequalities. The Matrix A is set to the identity matrix and the solver settings are kept the same as in the previous example, along with the same transformation of the generalized power cone to a product of 3-dimensional power cones as in (5.13).

Table 5.3: Tests with the generalized power cone

	GenPow		Hypatia (GenPow)		Clarabel-Pow		Mosek	
n	time	iter	time	iter	time	iter	time	iter
100	0.014	20	0.075	13	0.017	20	0.016	10
500	0.079	49	0.722	18	0.062	24	0.046	12
2500	0.648	80	31.243	21	0.431	26	0.297	14

Table 5.4: Tests with the power mean cone

	PowMean		Hypatia (PowMean)	
n	time	iter	time	iter
100	0.019	25	0.091	17
500	0.087	49	1.228	16
2500	1.865	63	33.297	21

We vary the dimension n from 100 to 2500 and summarize the computational results in Table 5.3 and Table 5.4. Although `GenPow` and `PowMean` takes more time than `Clarabel-Pow` and `Mosek` in total time, each iteration is much faster in our sparse implementation. This demonstrates the efficacy of our sparse augmentation approach compared with `Hypatia` that based on the dense Cholesky factorization for solving linear systems. Since both our IPM and `Hypatia` need to solve the same linear system at each iteration and the sparse implementation of the Hessian matrix is isolated from the remaining parts of an IPM, it could be exploited in Hypatia for acceleration.

6

Early termination technique for primal-dual algorithms in mixed integer conic programs

Contents

6.1 Introduction

Many techniques have been developed to speed up MIP computation. Cutting plane methods are widely used and can significantly reduce the number of nodes that a B&B solver must visit. Presolving [26] is a collection of problem reduction operations applied before solving an MIP, including bound strengthening, coefficient strengthening, constraint reduction and conflict analysis. In addition to presolving an MIP, one can also apply many heuristic methods to accelerate the computation. Most acceleration methods can be broadly classified into two types, start and improvement heuristics [152], both of which are crucial for pruning nodes in B&B algorithms. Start heuristics aim to find a feasible solution as early as possible when the B&B algorithm starts, e.g. feasibility pump methods [27]. On the other hand, improvement heuristics search for feasible points of better objective value based on information from feasible points already obtained, e.g. RINS [153] and the crossover method [154].

Pruning is usually an effective method to reduce the total number of nodes to be solved in B&B. Suppose U is the upper bound corresponding to the value of the best integer feasible solution so far. After updating the upper bound U with a new integer feasible point, one can prune any unevaluated nodes that are known to have an optimal value or a lower bound that is greater than U. Consequently, if a dual feasible point of a relaxed problem within a B&B search can be generated prior to convergence with its dual objective already larger than the current upper bound U, then one can stop the node computation immediately before solving it to optimality. This is called *early termination* and has been implemented in dual feasible algorithms like dual active-set methods [28, 36, 155, 156]. However, many key ideas in dual feasible methods, such as the use of basic feasible solutions, are not easily generalizable to conic programming.

At the heart of any B&B method is an optimization algorithm for solving convex problems. Many state-of-the-art conic optimization algorithms are primal-dual methods, and most can be classified into two types: second-order methods such

as the IPM [157], and first-order methods such as the OSM [72]. Both of them start from an infeasible initial point, and attain a feasible point when the algorithm converges to a global optimum and generate a certificate of infeasibility otherwise. This makes early termination difficult since primal-dual methods do not typically reach a dual feasible point until the algorithm converges at optimality. Recently, [29] proposed a heuristic method to generate a dual feasible point for a specialized primal-dual IPM, but the feasibility of dual iterates is still not theoretically guaranteed and it applies only to mixed-integer quadratic programming.

In this chapter we propose an early termination strategy for MICP to primal-dual optimization methods including both OSMs and IPMs [158, 159]. We develop efficient methods to find a dual feasible point for early termination at each iteration under the boundedness assumption. Then, we relax the boundedness assumption to a more general rank condition on the problem data that is applicable to many real-world scenarios. We propose a simple correction step that costs $\mathcal{O}(n)$ flops for bounded problems, and a more general optimization-based one costing $\mathcal{O}(n^2)$ flops at each iteration once we obtain a factorization at the start of an MICP. Both costs are relatively small compared to the factorization time $\mathcal{O}(n^3)$ per iteration in IPMs and no worse than the per iteration cost of OSMs. We also show that mixed-integer model predictive control (MIMPC) with bounded input satisfies the condition for the optimization-based correction.

6.2 Problem description

We will consider MICPs in the general form:

$$\begin{aligned} \min_{x,s} \quad & \frac{1}{2}x^\top Px + q^\top x \\ \text{s.t.} \quad & Gx = h \\ & Ax + s = b,\ s \in \mathcal{K}, \\ & \bar{l} \le x \le \bar{u},\ x_{\mathbb{I}} \in \mathcal{Z}, \end{aligned} \tag{6.1}$$

where $G \in \mathbb{R}^{p\times n}, A \in \mathbb{R}^{m\times n}, h \in \mathbb{R}^p, b \in \mathbb{R}^m$ and $\mathcal{K}$ is a proper cone. The vector $x \in \mathbb{R}^n$ is the decision variable with interval bounds defined by $\bar{l}, \bar{u} \in \mathbb{R}^n$, and $\mathbb{I}$

denotes the entries of x constrained to a finite integer set $\mathcal{Z}$. The objective function is convex quadratic with $P \in \mathbb{S}^n_+$ and vector $q \in \mathbb{R}^n$.

6.2.1 Branch and bound

We denote the continuous relaxation of (6.1) within a B&B solver as

$$\begin{aligned} \min_{x,s} \quad & \frac{1}{2}x^\top Px + q^\top x \\ \text{s.t.} \quad & Gx = h \\ & Ax + s = b,\ s \in \mathcal{K}, \\ & l \le x \le u, \end{aligned} \qquad \text{CP}(l,u) \tag{6.2}$$

where the integer relaxation of $\mathcal{Z}$ is incorporated into the box constraint $\bar{l} \le l \le x \le u \le \bar{u}$.

The B&B method computes an optimal solution x in (6.1) by exploring different integer combinations in a tree. It repeatedly branches on entries of x in the integer index set $\mathbb{I}$ and solves continuous convex relaxation subproblems in the form of (6.2) until a global optimizer is found. Meanwhile, B&B always maintains a global upper bound U corresponding to the value of the best integer feasible solution of (6.1) found so far. This upper bound is very useful in pruning unsolved nodes, and we will explore early evaluation of this bound inside each convex subproblem in the rest of this chapter.

6.2.2 Dual form for OSMs

Following [158], the dual of the continuous relaxation (6.2) is

$$\begin{aligned} \max_{x,y,y_b,z} \quad & -\frac{1}{2}x^\top Px - h^\top z + b^\top y - \sigma_{[l,u]}(y_b) \\ \text{s.t.} \quad & Px + q + G^\top z - A^\top y + y_b = 0, \\ & x \in \mathbb{R}^n, y \in \mathcal{K}^\circ, y_b \in \mathbb{R}^n, z \in \mathbb{R}^p, \end{aligned} \tag{6.3}$$

where the support function $\sigma_{[l,u]}(y_b)$ is explicit, i.e.

$$\sigma_{[l,u]}(y_b) = u^\top y_b^+ + l^\top y_b^-, \tag{6.4}$$

where $y_b^+ = \max\{y_b, 0\}, y_b^- = \min\{y_b, 0\}$, which is suitable to generate a correction for early termination of any MIP based on an OSM solver, e.g. OSQP [160] and PDHG [161].

6.2.3 Dual form for primal-dual IPMs

For IPMs that rely on LHSCB functions [157], there is no standard explicit barrier function for box constraints. We instead reformulate the box constraint $l \leq x \leq u$ into two nonnegative inequalities $x \geq l, x \leq u$ that have well-defined barrier functions, and obtain the alternative dual formulation:

$$\begin{aligned} \max_{x,y,y_+,y_-,z} \quad & -\frac{1}{2}x^\top Px - h^\top z - b^\top y - u^\top y_+ + l^\top y_- \\ \text{s.t.} \quad & Px + q + G^\top z + A^\top y + y_+ - y_- = 0 \\ & x \in \mathbb{R}^n, y \in \mathcal{K}^*,\ y_- \geq 0, y_+ \geq 0, z \in \mathbb{R}^p, \end{aligned} \tag{6.5}$$

where $\mathcal{K}^* = -\mathcal{K}^\circ$ for a proper cone $\mathcal{K}$. If we define $y_b := y_+ - y_-$ for (6.5), then we find that the dual form for IPMs (6.5) is equivalent to its counterpart (6.3) for OSMs. We can therefore design a unified dual correction mechanism for both IPMs and OSMs, which we will describe in Section 6.3.

The primal-dual IPM typically requires factorization of a matrix in the form

$$K^k := \begin{bmatrix} P & G^\top & A^\top \\ G & 0 & 0 \\ A & 0 & -H^k \end{bmatrix} \tag{6.6}$$

to compute the search direction for every iteration k, where H^k is a scaling matrix that depends on the choice of cones but which is always positive semidefinite. By adding small perturbation to diagonals of K^k, the matrix can become quasi-definite and be factorized by LDL decomposition with complexity $\mathcal{O}((n+p+m)^3)$ [148]. Both OSMs [160], [161] and IPMs [45, 46] always generate a sequence $(x^k, s^k, z^k, y^k, y_b^k)$ such that $s^k \in \mathcal{K}$ and $y^k \in \mathcal{K}^*$.

6.3 Early termination for primal-dual algorithms

The key to our proposed early termination method is to utilize the current dual iterate, which has a conic feasible y^k from a primal-dual algorithm (either an

OSM or an IPM), and then *remove linear dual residuals by adding corrections to unconstrained dual variables y_b in* (6.3) *or* (6.5). We thereby obtain a dual feasible solution for (6.3) or (6.5) and generate the corresponding dual cost V for early termination. If $V \geq U$, we can terminate the node computation before solving it to optimality since the optimum of the convex relaxation is not better than the current best feasible solution (with the objective value U).

To ensure our early termination always works, we first make the following boundedness assumption:

Assumption 6.3.1. *The domain of x in the MICP relaxation* (6.2) *is bounded, i.e. $l, u \in \mathbb{R}^n$ are both finite.*

Since y_b is the dual multiplier of the box constraint $[l, u]$, it becomes unconstrained under Assumption 6.3.1, i.e. $y_b \in \mathbb{R}^n$. The assumption works for many real world scenarios, e.g. x is an 0-1 switching signal or subjected to some physical limitations, like in some QP problems where $\|x\|$ is bounded. We will also show how to relax this assumption in Section 6.4.

6.3.1 Correction for OSMs

Suppose OSMs can always generate iterates $y^k \in \mathcal{K}^\circ$, $\forall k \geq 0$. For any dual iterates (x^k, y^k, y_b^k, z^k) generated by an OSM, we can offset the linear residual

$$r^k := Px^k + q + G^\top z^k - A^\top y^k + y_b^k \tag{6.7}$$

by setting $\Delta y_b^k = -r^k$ so that $(x^k, y^k, y_b^k + \Delta y_b^k, z^k)$ is a dual feasible point for (6.3), which is suitable for the early termination check. We find the conic feasibility $y^k \in \mathcal{K}^\circ$ always holds for any OSM because we always tackle a conic constraint $s \in \mathcal{K}$ in (6.2) by either projection to the polar cone $\mathcal{K}^\circ$, i.e. $\Pi_{\mathcal{K}^\circ}(v^k)$, or projection to $\mathcal{K}$, i.e. $\Pi_{\mathcal{K}}(v^k)$, w.r.t. the intermediate iterates v^k. The former is what we want for early termination directly, as in a primal-dual hybrid gradient (PDHG) solver [161]. For the latter, due to the Moreau decomposition [162, §2.5],

$$v = \Pi_{\mathcal{K}}(v) + \Pi_{\mathcal{K}^\circ}(v), \ \forall v, \tag{6.8}$$

we can generate an *equivalent* dual iterate $(I - \Pi_{\mathcal{K}})(v) \in \mathcal{K}^\circ$, which gives the y^k we obtain in ADMM [158]. Therefore, the early termination method we proposed is applicable to any OSM within a B&B solver.

6.3.2 Correction for primal-dual IPMs

The main idea behind our correction strategy is to adjust the iterate $(x^k, y^k, y_+^k, y_-^k, z^k)$ to be dual feasible via the correction on the dual of box constraint. A similar idea can be applied to primal-dual IPMs, which also generate dual-feasible conic iterates y^k for every iteration k. Suppose we define $\Delta y_b := \Delta y_+ - \Delta y_-$ with $\Delta y_+, \Delta y_- \geq 0$ for the IPM dual formulation (6.5). We can verify Δy_b is an unconstrained variable for the dual correction. If we only make corrections on y_- and y_+, leaving other variables fixed, then the change of dual cost in (6.5) becomes

$$\begin{aligned} -\Delta y_+^\top u + \Delta y_-^\top l &= \Delta y_+^\top (l-u) + (\Delta y_- - \Delta y_+)^\top l \\ &= \Delta y_+^\top (l-u) - \Delta y_b^\top l. \end{aligned} \tag{6.9}$$

Note that we have $\Delta y_+^\top (l-u) \leq 0$ due to $\Delta y_+ \geq 0$, $l - u \leq 0$. Meanwhile, the linear residual is

$$r^k := Px^k + q + G^\top z^k + A^\top y^k + y_+^k - y_-^k \tag{6.10}$$

before the correction. To maximize the dual objective in (6.5) given $\Delta y_b^k = -r^k$, we set $\Delta y_+^k, \Delta y_-^k$ as

$$\Delta y_+^k = \max\{0, \Delta y_b^k\}, \quad \Delta y_-^k = \Delta y_+^k - \Delta y_b^k. \tag{6.11}$$

Hence, $(x^k, y^k, y_+^k + \Delta y_+^k, y_-^k + \Delta y_-^k, z^k)$ is a dual feasible point and we can enable early termination checking via (6.5).

6.4 Optimization-based correction

In Section 6.3.1 and 6.3.2 we applied a box-like correction to every entry of y_b^k to ensure dual feasibility, which explains the need for Assumption 6.3.1. However, the crux of our early termination strategy is to offset the linear residual r^k in (6.10) via

corrections on unconstrained dual variables, which means we can exploit other dual variables beyond box constraints. If we allow for corrections to the unconstrained dual variables x, y_b, z in early termination, then Assumption 6.3.1 can be generalized to the following:

Assumption 6.4.1. *$[P, I_{\mathcal{B}}^\top, G^\top]$ has rank n, i.e. full row-rank, where $\mathcal{B}$ is the set of entries that have explicit bounded constraints $l_{\mathcal{B}} \leq x_{\mathcal{B}} \leq u_{\mathcal{B}}$ and $I_{\mathcal{B}}$ is the incidence matrix from the span of x to entries in $\mathcal{B}$, i.e. $x_{\mathcal{B}} = I_{\mathcal{B}} x$.*

We can always generate a dual feasible correction $(\Delta x^k, \Delta y_b^k, \Delta z^k)$, given Assumption 6.4.1, since the linear system

$$P\Delta x^k + I_{\mathcal{B}}^\top \Delta y_{\mathcal{B}}^k + G^\top \Delta z^k = -r^k. \tag{6.12}$$

always has a solution. It is also a generalization for setting $\Delta y_b^k = -r^k$ discussed in Section 6.3.2, which is useful if some entries of l, u for box constraints are infinite or the difference $u - l$ is so large that the corrected dual cost is excessively sensitive to the correction Δy_b^k.

Due to the existence of different coefficients for the support function $\sigma_{[l,u]}(y_b)$ in (6.3), or $-u^\top y_+ + l^\top y_-$ in (6.5), we divide the optimization-based correction into two steps. For the first step, we solve the optimization problem

$$\begin{aligned} \min_{\Delta x^k, \Delta z^k, \Delta y_{\mathcal{B}}^k} \; & \frac{1}{2}\Delta x^{k\top} P \Delta x^k + (Px^k)^\top \Delta x^k + h^\top \Delta z^k \\ & + \frac{\eta}{2}\|\Delta y_{\mathcal{B}}^k\|^2 + \frac{\gamma}{2}\|\Delta z^k\|^2 \\ \text{s.t.} \quad & P\Delta x^k + I_{\mathcal{B}}^\top \Delta y_{\mathcal{B}}^k + G^\top \Delta z^k = -r^k, \end{aligned} \tag{6.13}$$

which produces a correction $(\Delta x^k, \Delta y_{\mathcal{B}}^k, \Delta z^k)$ while maximizing the corrected dual cost w.r.t. $\Delta x^k, \Delta z^k$ with regularizations for $\Delta y_{\mathcal{B}}^k, \Delta z^k$. The corresponding KKT condition of (6.13) is

$$\begin{bmatrix} P & I_{\mathcal{B}}^\top & G^\top \\ I_{\mathcal{B}} & -\eta I & 0 \\ G & 0 & -\gamma I \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y_{\mathcal{B}}^k \\ \Delta z^k \end{bmatrix} = \begin{bmatrix} -r^k \\ -I_{\mathcal{B}} x^k \\ h - Gx^k \end{bmatrix} \tag{6.14}$$

if we set $\lambda^k = x^k + \Delta x^k$, which is the dual multiplier of (6.13). The matrix on the left-hand side does not depend on the active node, and hence only needs to

be factored once at the initialization of an MIP solver and can be reused later for any node's computation. Meanwhile, solving (6.14) is computationally efficient given the factors of the matrix on the LHS, or not worse than computation of an OSM per iteration. For the second step, we complete Δy_b^k by setting $\Delta y_j = 0$ for any index $j \notin \mathcal{B}$. If an IPM is used, we compute $\Delta y_+^k, \Delta y_-^k$ from Δy_b^k as in (6.11), and $(x^k + \Delta x^k, y^k, y_+^k + \Delta y_+^k, y_-^k + \Delta y_-^k, z^k + \Delta z^k)$ is a dual feasible point for early termination.

6.5 Applications in model predictive control

A common type of MIP arising in control engineering is optimal control with discrete-valued inputs as encountered in hybrid MPC problems, which takes the form:

$$\begin{aligned} \min_{x,u} \ & \sum_{t=0}^{T-1} (x_t^\top Q_t x_t + u_t^\top R_t u_t) + x_T^\top Q_T x_T + 2 q_T^\top x_T \\ \text{s.t.} \ & x_{t+1} = \bar{A} x_t + \bar{B} u_t, \ x_0 = x_{init}, \\ & u_t \in \mathcal{U}_t, \quad \forall t = 0, 1, \ldots, T-1, \end{aligned} \tag{6.15}$$

where $x_{init} \in \mathbb{R}^{n_x}$ is the initial state, $(\bar{A}, \bar{B})$ models the system dynamics, and $\mathcal{U}_t$ describes the constraints for each input $u_t \in \mathbb{R}^{n_u}$. The set $\mathcal{U}_t$ can be composed entirely or in part by integer constraints. We assume the input constraints $\mathcal{U}_t$ are bounded for each time horizon.

The box-like correction in Section 6.3 is directly applicable when the problem (6.15) reduces to the form that only has input variables u_t without state variables x_t. However, the sparse formulation (6.15) allows us to use sparse linear algebra that is more suitable for large MPC problems. Though the box-like correction is not directly applicable for (6.15), the optimization-based correction in Section 6.4 is suitable for the hybrid-MPC (6.15) due to the following theorem:

Theorem 6.5.1. *Assumption 6.4.1 is satisfied in the hybrid MPC problem* (6.15) *when* $\mathcal{U}_t$ *is bounded for* $t = 0, \ldots, T-1$.

Proof. Suppose $x := [x_0; \ldots; x_T; u_0; \ldots; u_{T-1}]$. The corresponding block components of $[P, I_{\mathcal{B}}^\top, G^\top]$ become

$$P = \begin{bmatrix} Q & \\ & R \end{bmatrix}, I_{\mathcal{B}} = \begin{bmatrix} 0_{n_u T \times n_x (T+1)} & I_{n_u T} \end{bmatrix},$$

$$G = \left[\begin{array}{ccccccc|ccccc} I & & & & & & & 0 & & & & \\ \bar{A} & -I & & & & & & \bar{B} & & & & \\ & \bar{A} & -I & & & & & & \ddots & & & \\ & & \ddots & \ddots & & & & & & \ddots & & \\ & & & & \bar{A} & -I & & & & & & \bar{B} \end{array}\right],$$

where $Q = \mathrm{Diag}(Q_0, \ldots, Q_T), R = \mathrm{Diag}(R_0, \ldots, R_{T-1})$ are block diagonal. Hence, $[P, I_{\mathcal{B}}^\top, G^\top]$ can be reordered into an upper triangular matrix in the form

$$\left[\begin{array}{ccccccc|ccc} I_{n_x} & & & \bar{A}^\top & & & & & & \\ & I_{n_u} & & \bar{B}^\top & & & & \vdots & \vdots & \\ & & & -I_{n_x} & & & & \vdots & \vdots & \\ & & & & \ddots & & \bar{A}^\top & \vdots & \vdots & \\ & & & & & I_{n_u} & \bar{B}^\top & \vdots & \vdots & \\ & & & & & & -I_{n_x} & & & \end{array}\right].$$

The matrix above is full row rank since every diagonal term is either 1 or -1. Hence, $[P, I_{\mathcal{B}}^\top, G^\top]$ is full row rank and the Assumption 6.4.1 is satisfied. □

Note that the system (6.14) is also banded for the sparse formulation (6.15) and we can exploit its structure to accelerate the computation as in [163], which reduces the factorization time from $\mathcal{O}(T^3)$ to $\mathcal{O}(T)$. Moreover, our method can be applied directly to outer approximation (OA) [21] if the OA only replaces the conic constraint $\mathcal{K}$ with some linear constraints without introducing new variables. Otherwise, we can combine it with bound strengthening techniques [26] to satisfy Assumption 6.4.1.

6.6 Algorithm and complexity of computation

We next summarize how to implement early termination in a B&B method, which corresponds to steps 3-17 in Algorithm 4. For every iteration k in a node CP($\underline{x}, \bar{x}$), we can obtain a primal-dual iterate $(x^k, s^k, y^k, y_b^k, z^k)$ from an OSM or an IPM

with an approximate dual cost D^k. Note that this iterate is conic feasible but doesn't satisfy the dual linear constraint, i.e. (6.3) or (6.5). We then check whether the algorithm finds an optimal solution $\hat{x}$ or detects the infeasibility of CP($\underline{x}, \bar{x}$) (steps 5-10). These steps are inherent to a primal-dual algorithm even without early termination and do not incur any additional cost. We then activate early termination when we find the approximate dual cost is larger than the current upper bound, i.e. $D^k \geq U$ (step 11). This heuristic follows [29] since D^k is close to the optimal solution of CP($\underline{x}, \bar{x}$) when the dual linear residual r^k is small enough, and can save computation time on early termination.

Once early termination is enabled, we compute a feasible correction $(\Delta x^k, \Delta y_b^k, \Delta z^k)$ using one of the correction methods discussed in Section 6.3.1, 6.3.2 or Section 6.4 and obtain the dual cost $\underline{D}^k$ at the dual feasible point $(x^k + \Delta x^k, y^k, y_b^k + \Delta y_b^k, z^k + \Delta z^k)$ for OSMs or $(x^k, y^k, y_+^k + \Delta y_+^k, y_-^k + \Delta y_-^k, z^k)$ for IPMs (step 12). If $\underline{D}^k > U$, we know the optimum of CP($\underline{x}, \bar{x}$) is larger than $\underline{D}^k$ due to weak duality, and hence larger than U, which indicates we can stop the node computation and prune this node immediately (step 14). Otherwise, we continue computing until we solve CP($\underline{x}, \bar{x}$) and proceed with the standard B&B method (steps 18-27).

Algorithm 4 B&B for MICP with early termination

Require:

Initialization: $U \leftarrow +\infty$, node tree $\mathcal{T} \leftarrow$ CP(l, u)

1: **while** $\mathcal{T} \neq \emptyset$ **do**
2: Pick and remove CP($\underline{x}, \bar{x}$) from $\mathcal{T}$
3: **for** k = 1, 2 ... **do**
4: Generate $(x^k, s^k, y^k, y_b^k, z^k)$ and an estimated dual cost D^k from OSMs or IPMs
5: **if** *termination criteria is satisfied* **then**
6: return optimal solution $\hat{x} = x^k$ and $f(\hat{x})$
7: **end if**
8: **if** *infeasibility of* CP($\underline{x}, \bar{x}$) *is detected* **then**
9: return CP($\underline{x}, \bar{x}$) infeasible

10: **end if**
11: **if** $D^k \geq U$ **then**
12: Compute the dual cost $\underline{D}^k$ at the dual feasible point $(x^k + \Delta x^k, y^k, y_b^k + \Delta y_b^k, z^k + \Delta z^k)$
13: **if** $\underline{D}^k \geq U$ **then**
14: return CP$(\underline{x}, \bar{x})$ terminates early
15: **end if**
16: **end if**
17: **end for**
18: **if** CP$(\underline{x}, \bar{x})$ *terminates early* or is *infeasible* **then**
19: prune current node
20: **else if** $f(\hat{x}) > U$ **then**
21: prune current node
22: **else if** $\hat{x}$ is *integer feasible* **then**
23: $U \leftarrow f(\hat{x}), x^* \leftarrow \hat{x}$
24: prune nodes in $\mathcal{T}$ with lower bound $> U$
25: **else**
26: branch node CP$(\underline{x}, \bar{x})$
27: **end if**
28: **end while**

Let us now consider the computational complexity of early termination. The estimated dual cost D^k, the iterate (x^k, y^k, y_b^k, z^k) and the residual r^k (6.10) are already computed from a primal-dual algorithm and therefore do not incur any extra cost for checking early termination. We recompute corrections $\Delta x, \Delta z, \Delta y_b$ and the corrected cost $\underline{D}^k$ at every time we check for early termination. Indeed, we compute the cost change $\Delta D^k := \underline{D}^k - D^k$ first and then the corrected cost via $\underline{D}^k = D^k + \Delta D^k$, which is more efficient than computing $\underline{D}^k$ directly. The box-like correction (6.9) requires an additional $\mathcal{O}(n)$ flops to generate a feasible dual cost. For an optimization-based correction (6.13), we need no more than $\mathcal{O}((2n+p)^2)$ flops to solve the linear system (6.14) if we save the factorization of the matrix

in (6.14) from the start of an MICP. Both correction costs are relatively small compared to the $\mathcal{O}(n+p+m)^3$ flops per IPM iteration. For an OSM, each early termination check is no more costly than the original computation per iteration, so that its computational time is negligible inside every M iterations, e.g. $M = 25$ in OSQP [118]. In addition, we empirically found that the early termination is usually activated immediately when the heuristic check $D^k \geq U$ is true (step 11).

6.7 Numerical results

We implement Algorithm 4 and a counterpart without early termination, i.e. removing steps 3-17 in Algorithm 4. Both were written in Julia with every convex relaxation solved by the IPM solver `Clarabel.jl` [46][1]. Tests are implemented on Intel Core i7-9700 CPU @3.00GHz, 16GB RAM. Experiments for OSMs can be found in [158].

6.7.1 Mixed integer model predictive control

We next consider the current reference tracking problem in power electronics [24]:

$$\begin{aligned} \min_{x,u} \quad & \sum_{t=0}^{T-1} \gamma^t l(x_t) + \gamma^T V(x_T) \\ \text{s.t.} \quad & x_0 = x_{\text{init}}, \\ & x_{t+1} = \bar{A}x_t + \bar{B}u_t, \ ||u_t - u_{t-1}||_\infty \leq 1, \\ & u_t \in \{-1, 0, 1\}^6, \ \forall t = 0, 1, \ldots, T-1, \end{aligned} \tag{6.16}$$

where γ is a discount factor and T is the time horizon. The quadratic state penalty cost $l(x_t)$ is for current tracking and $V(x_T)$ is a final stage quadratic cost computed using approximate dynamic programming. The initial state is x_{init} and the system dynamics is $x_{t+1} = \bar{A}x_t + \bar{B}u_t$ with $x_t \in \mathbb{R}^{12}$ representing the internal motor currents, voltages and the input $u_t \in \mathbb{R}^6$ including three semiconductor devices positions with integer values $\{-1, 0, 1\}$ and three additional binary components required to model the system. The ramp rate constraint $\|u_t - u_{t-1}\|_\infty \leq 1$ avoids shoot-through in

[1]Additional tests for OSMs can be found in [158] based on COSMO solver [19].

the inverter positions (changes from -1 to 1 or vice-versa) that can damage the components.

By eliminating $x_t, t \in \{1, \ldots, T\}$ via the state dynamics, (6.16) reduces to a problem depending only on the input variables $u_0, \ldots, u_{T-1}$ and the initial state x_0; we refer readers to [24] for details. We set $T = 8$ for the time horizon and simulate closed-loop MIMPC for 100 consecutive intervals. Figure 6.1 compares the performance of B&B with and without early termination. We apply the simple early termination introduced in Section 6.3.2. We start to count time only after the first feasible solution of (6.16), and consequently a finite upper bound U is found. This ensures that the reductions shown in Figures 6.1 and 6.2 represent the reduction in computation cost relative to an idealized omniscient early termination scheme. For all 100 intervals, early termination has produced a noticeable reduction in computational time, averaging to about 20%.

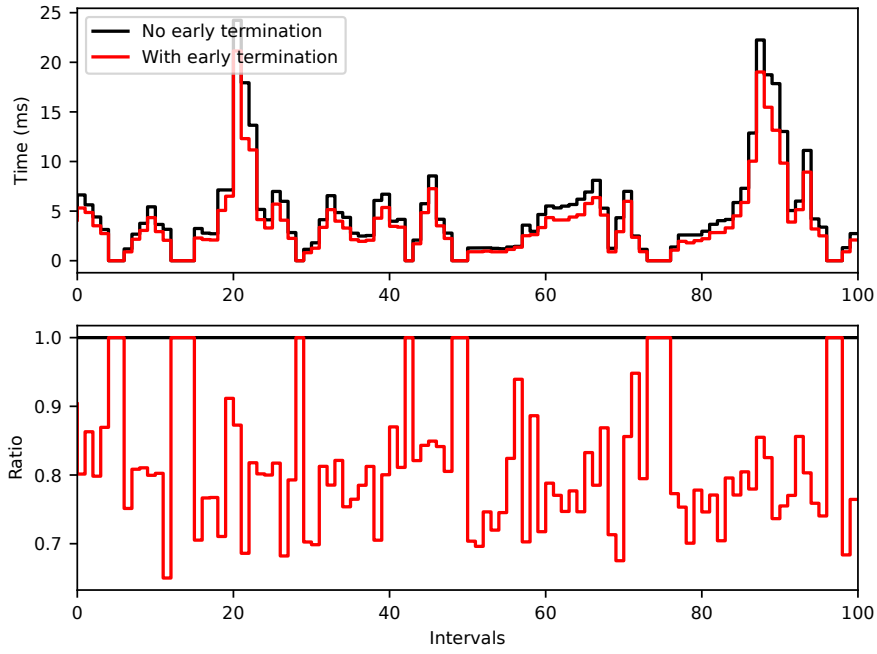


Figure 6.1: MIMPC $T = 8$, reduced dense form

We implement another experiment for (6.16) using the sparse form discussed in Section 6.5 with the optimization-based correction of $\eta = \gamma = 1$. Figure 6.2 shows

it also reduces the computational time about 15% to 20% over 100 intervals.

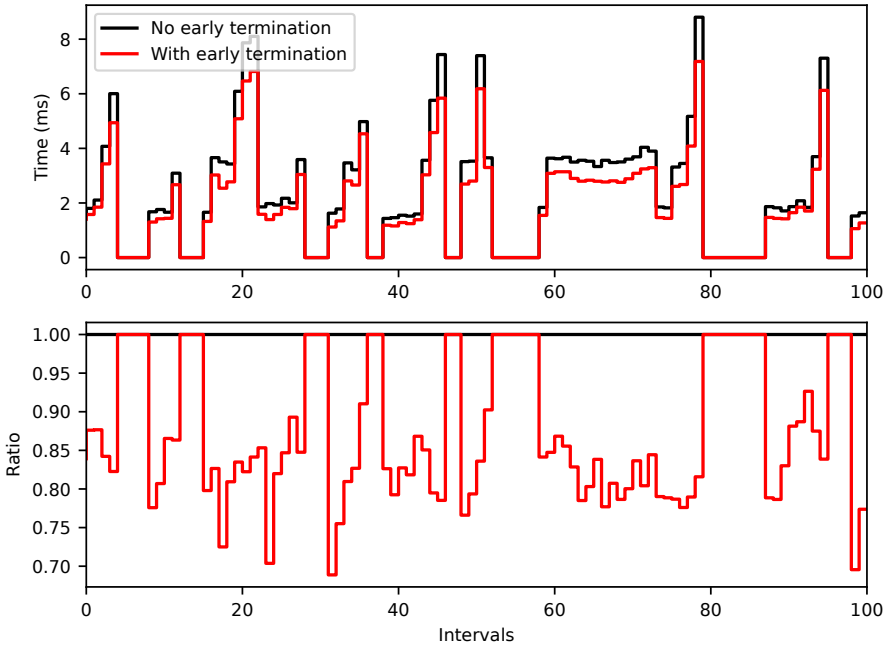


Figure 6.2: MIMPC $T = 8$, sparse form

6.7.2 Portfolio optimization

We also test our proposed early termination technique on a portfolio optimization problem, which can be formulated as a mixed integer second-order cone program [23],

$$\begin{aligned}
\min_{x,b,l} \quad & r^\top x \\
\text{s.t.} \quad & x^\top \Lambda x \leq \rho, \\
& \textstyle\sum_{i=1}^n x_i = 1, \quad \sum_{i=1}^n b_i \leq K, \\
& L_{min} \leq \textstyle\sum_{i=1}^n l_i \leq L_{max}, \; b \leq Hl, \; l \leq H^\top b, \\
& l_j \in \{0,1\}, \text{ for } j \in \{1,...,L\} \\
& -b_i \leq x_i \leq b_i, b_i \in \{0,1\}, i \in \{1,...,n\}.
\end{aligned}$$

There are n assets in total, categorized into L industry sectors, with the mapping from assets to sectors captured by matrix $H \in \mathbb{R}^{n \times L}$. We define $x \in \mathbb{R}^n$ as the fractions of portfolio value held in each asset: $x_i > 0$ and $x_i < 0$ denote buying and selling (i.e. shorting) respectively, and must sum to unity. The vector $r \in \mathbb{R}^n$ is the expected return for n assets, and Λ is the covariance for market volatility

and restricted below a certain level ρ, which is formulated as a second-order cone constraint. The binary vectors $b \in \mathbb{R}^n$ denotes whether we invest in an asset or not, and $l \in \mathbb{R}^L$ for investment in a sector respectively. The number of assets we can invest in is upper-bounded by K and the number of sectors is limited to $[L_{min}, L_{max}]$.

We use the early termination strategy as in Section 6.3.2 and choose $n = 20, L = 3$, $L_{min} = 1, L_{max} = L, \rho = 100, K = 10$ and simulate the portfolio problem over 100 consecutive days. We use data on returns from the S&P 500 for the period 2015-2020 grouped according to GICS sector. Estimates for r and Λ were computed using standard statistical methods; see [164, §13]. Figure 6.3 shows the early termination can reduce computation time about 10%-15% after we find the first integer feasible solution, which shows that our early termination method can also be effective for MICPs.

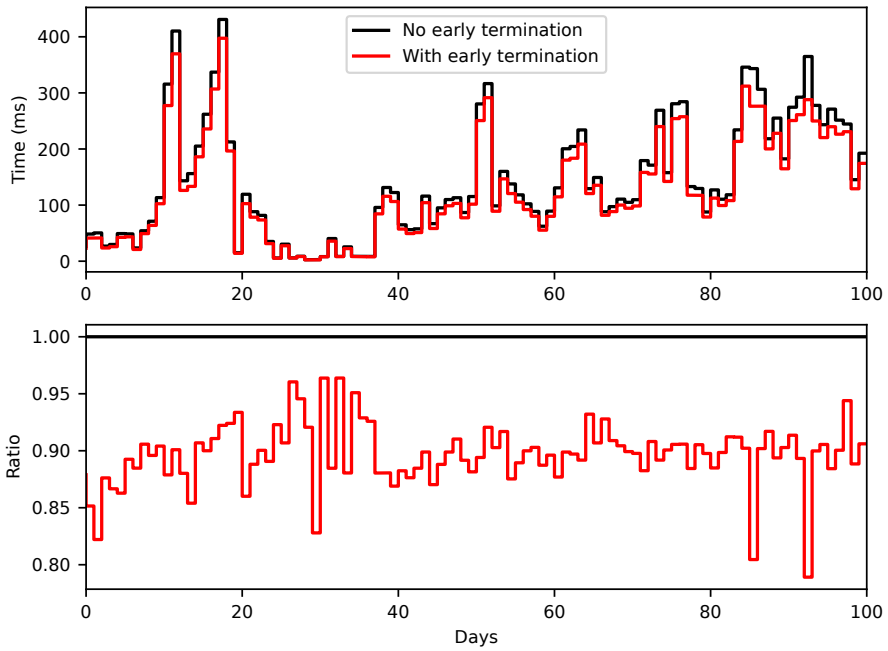


Figure 6.3: Portfolio optimization over 100 days

7 Conclusion

Contents

7.1 Contributions of this dissertation

The dissertation is focused on the analysis and implementation of various acceleration techniques for conic optimization. The contributions can be summarized as follows:

GPU acceleration for semidefinite programming

Semidefinite programming is known to be computationally expensive. Sparsity and low-rank information are exploited in the design of efficient algorithms for SDPs in Chapter 3. For diagonally constrained SDPs, we exploited low-rank information via Burer-Monteiro decomposition and developed an ADMM algorithm where each subproblem is equivalent to a parallel forward operation well-suited for GPU computation. The proximal variant can solve block-diagonally constrained SDPs and also benefits from parallel computation on GPU. We proved that the algorithms converge to a first-order stationary point and further to $O(1/r)$ global optimality with information from negative curvature.

Implementation of Clarabel solver

In Chapter 4, we presented the interior-point solver, Clarabel, for conic optimization with quadratic objectives by using the homogeneous embedding [95, 96] for infeasibility detection, which avoids the issue of transforming QPs into SOCPs. We also support exponential cones, power cones and positive definite cones in Clarabel. Our solver is competitive with the cutting-edge conic solvers on various benchmarks and outperforms them in terms of time and numerical stability on QPs and SOCPs.

Efficient factorization in interior point methods for high-dimensional power cones

A high-dimensional nonsymmetric cones can be written as an extended form with products of multiple basic nonsymmetric cones. However, using the extended formulation instead of the original high-dimensional nonsymmetric cone will enlarge the dimension of a optimization problem and increase the computational time for

it, which mainly comes from increased factorization time for the underlying linear systems.

In Chapter 5, we propose an augmented decomposition for the nonsymmetric scaling matrices of a class of nonsymmetric cones that ensures the augmented linear system is quasidefinite and the use of sparse LDL factorization, which saves factorization time compared to the existing dense linear factorization for those nonsymmetric cones.

Early termination for primal-dual algorithms within mixed integer conic programming

Branch-and-bound framework is the foundation of many solvers for mixed integer programming, and it relies on a convex optimization solver to solve the convex relaxation problems within it. Primal-dual algorithms such as primal-dual interior point methods and operator splitting methods are efficient for solving stand-alone convex problems. However, they can not easily exploit early termination check or warm-starting due to the infeasibility of generated iterates, which makes dual feasible algorithms, like dual simplex and active-set methods, more favourable in combination with branch-and-bound methods.

In Chapter 6, we proposed an early termination technique that can generate a point from iterates of primal-dual algorithms with (dual) feasibility guarantees. We showed it can be directly applied to general mixed integer conic programming and MIMPC. Generating a dual feasible point is also very efficient and usually succeeds at the first check when it is combined with a heuristic functional value check beforehand. Numerical results showed the proposed early termination can reduce about 10%-20% total computational time in MICPs.

7.2 Directions for future research

Finally, there are some possible directions for future research:

GPU acceleration for conic optimization

GPUs are becoming more prevalent in scientific computing recently due to its success on training deep neural networks, where computation is mainly on forward operations like matrix (tensor) multiplication. In Chapter 3, we developed an efficient ADMM algorithm that exploits both sparsity and the low-rank information and can be run in parallel on a GPU. It is also worth considering whether GPU acceleration is possible for general semidefinite programming. Moreover, GPU acceleration has been successfully implemented for convex optimization solvers based on first-order methods, e.g. SCS [18], OSQP [165] and PDLP [166]. It would be interesting to investigate whether our Clarabel solver can benefit from computation on GPUs.

Linear complementarity problems with nonsymmetric cones

For Clarabel solver, the proof of convergence is still missing for problems of nonsymmetric conic constraints and a quadratic cost. The main issue is the lack of proof for the existence of the central path, i.e. whether (4.8) has a solution $\forall \mu \in (0, +\infty)$ when both nonsymmetric conic constraints and a quadratic cost exist. If the central path for problems of nonsymmetric cones with quadratic cost is validly defined under the homogeneous embedding, we can follow a similar derivation in [145] to prove the convergence of the IPM.

Numerically stable algorithm for nonsymmetric cones

In Chapter 5, we proposed an efficient sparse LDL factorization for a nonsymmetric-scaling IPM with a class of high-dimensional nonsymmetric cones. Although the factorization time can be reduced dramatically, we observe that the number of iterations for convergence of high-dimensional nonsymmetric cones is higher compared to the equivalent problem solved by Mosek due to the smaller step size in each iteration, which is also found in tests of Hypatia. Future work would be designing more efficient higher-order corrections, improving the numerical stability

of IPMs and exploiting the low-rank property in the primal-dual scaling IPM for high-dimensional nonsymmetric cones.

Appendices

A

Appendix

Contents

A.1 Derivatives of LHSCB functions for cones

A.1.1 Symmetric cones

1. Nonnegative cone, $x \in \mathbb{R}^n_+$: for $f(x) = -\sum_{i\in[\![n]\!]} \log(x_i)$,
$$\nabla f(x) = -\mathrm{diag}(x^{-1})\mathbf{1}^n,$$
$$\nabla^2 f(x) = \mathrm{diag}(x^{-2}).$$

2. Second-order cone, $x \in \mathcal{K}^n_{\mathrm{soc}}$: for $f(x) = -\frac{1}{2}\log\left(x_1^2 - \sum_{i=2}^n x_i\right)$,
$$\nabla f(x) = -\frac{Jx}{x^\top Jx},$$
$$\nabla^2 f(x) = \frac{1}{(x^\top Jx)^2}\left(2Jxx^\top J - (x^\top Jx)J\right),$$
where $J := \mathrm{diag}([1; -\mathbf{1}^{n-1}])$.

3. Positive semidefinite cone, $\mathrm{mat}(x) \in \mathbb{S}^n_+$: for $f(x) = -\log\det(\mathrm{mat}(x))$,
$$\nabla f(x) = -\mathrm{vec}(\mathrm{mat}(x)^{-1}),$$
$$\nabla^2 f(x)[v] = \mathrm{vec}\left(\mathrm{mat}(x)^{-1}\mathrm{mat}(v)\mathrm{mat}(x)^{-1}\right).$$

A.1.2 Nonsymmetric cones

The barrier functions of the exponential cone (4.6) and the power cone (4.7) can both be written in the form:

$$f^*(z) = -\log(\psi(z)) + h(z). \tag{A.1}$$

Hence, the gradient and Hessian of dual exponential cones and dual power cones can be computed via

$$\nabla f^*(z) = -\frac{\nabla\psi(z)}{\psi(z)} + \nabla h(z), \quad \nabla^2 f^*(z) = \frac{\nabla\psi(z)\nabla\psi(z)^\top}{\psi^2(z)} - \frac{\nabla^2\psi(z)}{\psi(z)} + \nabla^2 h(z). \tag{A.2}$$

The third order directional derivatives used in the 3rd-order correction (4.23) are

$$\begin{aligned}\nabla^3 f^*(z)[u] = &-\frac{2\langle\nabla\psi(z), u\rangle}{\psi(z)^3}\cdot\nabla\psi(z)\nabla\psi(z)^\top + \frac{\langle\nabla\psi(z), u\rangle}{\psi(z)^2}\cdot\nabla^2\psi(z)\\ &+\frac{1}{\psi(z)^2}\cdot\nabla\psi(z)u^\top\nabla^2\psi(z) + \frac{1}{\psi(z)}\cdot\nabla^2\psi(z)u\cdot\nabla\psi(z)^\top \qquad \text{(A.3a)}\\ &-\frac{1}{\psi(z)}\cdot\nabla^2\psi(z)[u] + \nabla^3 h(z)[u],\end{aligned}$$

$$\begin{aligned}\nabla^3 f^*(z)[u,v] = &-\frac{2\langle\nabla\psi(z),u\rangle\cdot\langle\nabla\psi(z),v\rangle}{\psi(z)^3}\cdot\nabla\psi(z)+\frac{\langle\nabla\psi(z),u\rangle}{\psi(z)^2}\cdot\nabla^2\psi(z)v\\ &+\frac{\langle u,\nabla^2\psi(z)\nu\rangle}{\psi(z)^2}\cdot\nabla\psi(z)+\frac{\langle\nabla\psi(z),v\rangle}{\psi(z)^2}\cdot\nabla^2\psi(z)u\\ &-\frac{1}{\psi(z)}\nabla^3\psi(z)[u,v]+\nabla^3 h(z)[u,v]\\ =&\frac{\langle u,\nabla^2\psi(z)v\rangle\psi(z)-2\langle\nabla\psi(z),u\rangle\langle\nabla\psi(z),v\rangle}{\psi(z)^3}\nabla\psi(z)\\ &+\frac{\langle\nabla\psi(z),u\rangle}{\psi(z)^2}\nabla^2\psi(z)v+\frac{\langle\nabla\psi(z),v\rangle}{\psi(z)^2}\nabla^2\psi(z)u\\ &-\frac{1}{\psi(z)}\nabla^3\psi(z)[u,v]+\nabla^3 h(z)[u,v].\end{aligned} \tag{A.3b}$$

The specifications of $\psi(z)$ and $h(z)$ in dual exponential cones and dual power cones are given as follows:

Exponential cone

$\psi(z) := z_2 - z_1 - z_1\log(-z_3/z_1)$ and $h(z) := -\log(-z_1) - \log(z_3)$, and

$$\nabla\psi(z) = \begin{bmatrix} -\log(z_3)+\log(-z_1)\\ 1\\ -\frac{z_1}{z_3}\end{bmatrix},\ \nabla^2\psi(z) = \begin{bmatrix}\frac{1}{z_1} & 0 & -\frac{1}{z_3}\\ 0 & 0 & 0\\ -\frac{1}{z_3} & 0 & \frac{z_1}{z_3^2}\end{bmatrix}, \tag{A.4}$$

$$\nabla h(z) = \begin{bmatrix} -\frac{1}{z_1}\\ 0\\ -\frac{1}{z_3}\end{bmatrix},\ \nabla^2 h(z) = \begin{bmatrix}\frac{1}{z_1^2} & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & \frac{1}{z_3^2}\end{bmatrix}. \tag{A.5}$$

Power cone

$\psi(z) := \varphi(z) - z_3^2, \varphi(z) = \left(\frac{z_1}{\alpha}\right)^{2\alpha}\left(\frac{z_2}{1-\alpha}\right)^{2-2\alpha}$ and $h(z) := -(1-\alpha)\log(z_1) - \alpha\log(z_2)$, and

$$\begin{aligned}\nabla\psi(z) &= \begin{bmatrix}\frac{2\alpha\varphi(z)}{z_1}\\ \frac{2(1-\alpha)\varphi(z)}{z_2}\\ -2z_3\end{bmatrix},\ \nabla^2\psi(z) = \begin{bmatrix}\frac{2\alpha(2\alpha-1)\varphi(z)}{z_1^2} & \frac{4\alpha(2\alpha-1)\varphi(z)}{z_1z_2} & 0\\ \frac{4\alpha(2\alpha-1)\varphi(z)}{z_1z_2} & \frac{2(1-\alpha)(1-2\alpha)\varphi(z)}{z_2^2} & 0\\ 0 & 0 & -2\end{bmatrix},\\ \nabla h(z) &= \begin{bmatrix}-\frac{1-\alpha}{z_1}\\ -\frac{\alpha}{z_2}\\ 0\end{bmatrix},\ \nabla^2 h(z) = \begin{bmatrix}\frac{1-\alpha}{z_1^2} & 0 & 0\\ 0 & \frac{\alpha}{z_2^2} & 0\\ 0 & 0 & 0\end{bmatrix}.\end{aligned} \tag{A.6}$$

A.2 Conjugate gradient for barriers f^* of nonsymmetric cones

A.2.1 Exponential cone

The computation of connjugate gradient $\nabla f(s)$ follows [48]. The barrier function (4.6) is for $z \in \mathcal{K}_{\text{exp}}^*$,

$$f^*(z) = -\log\left(z_2 - z_1 - z_1 \log\left(\frac{z_3}{-z_1}\right)\right) - \log(-z_1) - \log(z_3).$$

If we define $\varphi(z) = \log(-\frac{z_3}{z_1})$ and $\zeta(z) = z_2 - z_1 - z_1\varphi(z)$, the gradient of $f^*(z)$ is

$$\nabla f^*(z) = \left(\frac{\varphi(z)}{\zeta(z)} - \frac{1}{z_1}, -\frac{1}{\zeta(z)}, \frac{z_1}{z_3\zeta(z)} - \frac{1}{z_3}\right)^\top. \tag{A.7}$$

If we abbreviate $\nabla f(s)$ as g, we have

$$\frac{\varphi(-g)}{\zeta(-g)} + \frac{1}{g_1} = -s_1, \tag{A.8a}$$

$$\zeta(-g) = -g_2 + g_1 + g_1\varphi(-g) = \frac{1}{s_2}, \tag{A.8b}$$

$$\frac{g_1}{g_3\zeta(-g)} + \frac{1}{g_3} = -s_3 \tag{A.8c}$$

due to $\nabla f^*(-\nabla f(s)) = -s$ from (2.23). We replace $\varphi(-g)$ and $\zeta(-g)$ in (A.8a) and (A.8c) with (A.8b) and obtain

$$\log\left(-\frac{g_3}{g_1}\right) s_2 + \frac{1}{g_1} = -s_1, \tag{A.9a}$$

$$g_3 = -\frac{1 + g_1 s_2}{s_3}, \tag{A.9b}$$

where we eliminate g_3 and can get

$$\log\left(\frac{1 + g_1 s_2}{g_1 s_3}\right) + \frac{1}{s_2 g_1} = -\frac{s_1}{s_2},$$

and rewrite it as

$$\log\left(1 + \frac{1}{s_2 g_1}\right) + 1 + \frac{1}{s_2 g_1} = -\frac{s_1}{s_2} - \log\left(\frac{s_2}{s_3}\right) + 1. \tag{A.10}$$

We can solve (A.10) via the *Wright-Omega* function $\omega + \log(\omega) = \beta$ where $\beta = 1 - \frac{s_1}{s_2} - \log\left(\frac{s_2}{s_3}\right), \omega = 1 + \frac{1}{s_2 g_1}$. A numerical method to solve the *Wright-Omega*

function can be found in [167]. Finally, we can compute the conjugate gradient $\nabla f(s)$ via ω, (A.8b) and (A.9b)

$$\nabla f(s) = \left(\frac{1}{(\omega-1)s_2}, \frac{1+\log(\omega \cdot \frac{s_2}{s_3})}{(\omega-1)s_2} - \frac{1}{s_2}, \frac{\omega}{s_3(1-\omega)} \right)^\top. \tag{A.11}$$

A.2.2 Generalized power cone

We consider the barrier function of dual generalized power cone $\mathcal{K}^*_{\text{gpow }(\alpha,d_1,d_2)}$

$$f^*_{\text{gpow}}(u,w) = -\ln\left(\prod_{i\in[d_1]} \left(\frac{u_i}{\alpha_i}\right)^{2\alpha_i} - \|w\|^2 \right) - \sum_{i\in[d_1]} (1-\alpha_i) \ln\left(\frac{u_i}{\alpha_i}\right) \tag{A.12}$$

with $\nu = d_1 + 1$. We denote $\varphi(u) = \prod_{i\in[\![d_1]\!]} (u_i/\alpha_i)^{2\alpha_i}$, and $\zeta(u,w) = \varphi(u) - \|w\|^2$. Our goal is to find a conjugate gradient $g_{(p,r)}$ such that $-g^*(-g_{(p,r)}) = (p,r) \in \mathcal{K}_{\text{gpow}(\alpha,d_1,d_2)}$, which is summarized in the next theorem, following the similar idea for the barrier of $\mathcal{K}_{\text{gpow}(\alpha,d_1,d_2)}$ from [105]:

Theorem A.2.1. *Suppose $(p,r) \in \mathcal{K}_{\text{gpow}(\alpha,d_1,d_2)}$. The corresponding primal conjugate gradient $g_{(p,r)} = (g_p, g_r)$ can be obtained via*

1. *If $r = 0$, we have*

$$g_{p_i} = -\frac{1+\alpha_i}{p_i}, \quad \forall i \in [\![d_1]\!],$$
$$g_{r_i} = 0, \quad \forall i \in [d_2].$$

2. *If $r \neq 0$, we have*

$$g_{p_i} = -\frac{(1+\alpha_i) + \alpha_i g_{\|r\|}\|r\|}{p_i}, \ \forall i \in [\![d_1]\!], \qquad g_r = \frac{g_{\|r\|} \cdot r}{\|r\|},$$

where $g_{\|r\|}$ is the root of

$$h(x) = \sum_{i\in[\![d_1]\!]} 2\alpha_i \ln\left(\|r\| \cdot x + \frac{1+\alpha_i}{\alpha_i} \right) - \ln\left(\frac{2x}{\|r\|} + x^2 \right) - \sum_{i\in[\![d_1]\!]} 2\alpha_i \ln(p_i). \tag{A.13}$$

The root of $h(x)$ is computable via the 1-dimensional Newton-Raphson method, which converges quadratically if we choose the initial point x_0 as

$$x_0 = -\frac{1}{\|r\|} + \frac{d_1\|r\| + \sqrt{\chi\left(\frac{\chi}{\|r\|^2} + d_1^2 - 1\right)}}{\chi - \|r\|^2},$$

where $\chi = \prod_{i\in[\![d_1]\!]} p_i^{2\alpha_i}$.

Proof of Theorem A.2.1. Following the preceding discussion, the dual gradient is

$$g^*_{u_i} = \frac{-2\alpha_i\varphi(u)}{u_i\zeta(u,w)} - \frac{1-\alpha_i}{u_i}, \tag{A.14a}$$

$$g^*_{w_i} = \frac{2w_i}{\zeta(u,w)}. \tag{A.14b}$$

If $r = 0$, we have $-\frac{2(-g_{r_i})}{\zeta(p,r)} = r_i = 0, \forall i \in [d_2]$ and then

$$g_{p_i} = -\frac{1+\alpha_i}{p_i}, \quad \forall i \in [\![d_1]\!],$$
$$g_{r_i} = 0, \quad \forall i \in [d_2],$$

from $\varphi(-g_p) = \zeta(-g_p, 0)$.

Suppose instead $r \neq 0$. We will denote $\zeta = \zeta(-g_p, -g_r)$ for simplicity of notation and will assume $r \geq 0$ without loss of generality. Suppose Q is an orthonormal transformation that maps $r \in \mathbb{R}^{d_2}$ to a vector of zeros except one at the first entry that is equal to $\|r\|$, i.e. $[\|r\|, 0, \dots]^\top$. Then the conjugate gradient under the transformation Q satisfies $g_r = Q^\top g_{Qr}$ due to (2.21), where the first entry of g_{Qr} is $g_{\|r\|}$ and all other entries are zeros. From (A.14b) and (2.23), we have

$$g_r = \frac{\zeta r}{2}. \tag{A.15}$$

Due to the invariance of ζ under orthonormal transformations, we then obtain

$$\zeta := \zeta(-g_p, -g_r) = \zeta(-g_p, -Q^\top g_{Qr}) = \zeta(-g_p, [-g_{\|r\|}; 0; \dots]) \overset{(A.15)}{=} \frac{2g_{\|r\|}}{\|r\|},$$

which can be substituted back into (A.15) to produce

$$g_r = \frac{g_{\|r\|} \cdot r}{\|r\|}.$$

From (A.14a) and (2.23) we can write the elements of the subvector p as

$$p_i = -\left(\frac{-2\alpha_i\varphi(-g_p)}{-g_{p_i}\zeta} - \frac{1-\alpha_i}{-g_{p_i}}\right), \ \forall i \in [\![d_1]\!]. \tag{A.16}$$

which can be rearranged to

$$p_i(-g_{p_i})\zeta = 2\alpha_i\varphi(-g_p) + (1-\alpha_i)\zeta. \tag{A.17}$$

Applying $\varphi(\cdot)$ over both sides of (A.17) for all i, we obtain

$$\prod_{i\in[\![d_1]\!]} \alpha_i^{2\alpha_i} \cdot \varphi(p)\varphi(-g_p)\zeta^2 = \varphi(2\alpha_i\varphi(-g_p) + (1-\alpha_i)\zeta).$$

Substituting

$$\varphi(-g_p) = \zeta + \|g_r\|^2 = \zeta + \|Q^\top g_{Qr}\|^2 = \zeta + g_{\|r\|}^2 = \frac{2g_{\|r\|}}{\|r\|} + g_{\|r\|}^2 \tag{A.18}$$

yields

$$\begin{aligned}
&\implies \prod_{i\in[\![d_1]\!]} \alpha_i^{2\alpha_i} \cdot \varphi(p) \left(\frac{2g_{\|r\|}}{\|r\|} + g_{\|r\|}^2\right) \left(\frac{2g_{\|r\|}}{\|r\|}\right)^2 = \varphi\left(2\alpha_i \left(\frac{2g_{\|r\|}}{\|r\|} + g_{\|r\|}^2\right) + (1-\alpha_i)\frac{2g_{\|r\|}}{\|r\|}\right) \\
&\implies \prod_{i\in[\![d_1]\!]} p_i^{2\alpha_i} \left(\frac{2g_{\|r\|}}{\|r\|} + g_{\|r\|}^2\right) \cdot \frac{4g_{\|r\|}^2}{\|r\|^2} = \prod_{i\in[\![d_1]\!]} \left(2g_{\|r\|}^2 + \frac{(1+\alpha_i)}{\alpha_i}\frac{2g_{\|r\|}}{\|r\|}\right)^{2\alpha_i} \\
&\implies \prod_{i\in[\![d_1]\!]} p_i^{2\alpha_i} \left(\frac{2g_{\|r\|}}{\|r\|} + g_{\|r\|}^2\right) = \prod_{i\in[\![d_1]\!]} \left(g_{\|r\|}\cdot\|r\| + \frac{1+\alpha_i}{\alpha_i}\right)^{2\alpha_i},
\end{aligned}$$

where computing $g_{\|r\|}$ reduces to finding the root of the function $h(x)$ defined in (A.13).

We next show that $h(x)$ is convex and monotonically decreasing. The gradient and Hessian of h are

$$\begin{aligned}
h'(x) &= \sum_{i\in[\![d_1]\!]} \frac{2\alpha_i\|r\|}{\|r\|\cdot x + \frac{1+\alpha_i}{\alpha_i}} - \frac{2(\|r\| + \frac{1}{x})}{\|r\|\cdot x + 2} \\
&< \sum_{i\in[\![d_1]\!]} \frac{2\alpha_i\|r\|}{\|r\|\cdot x + 2} - \frac{2(\|r\| + \frac{1}{x})}{\|r\|\cdot x + 2} = -\frac{\frac{2}{x}}{\|r\|\cdot x + 2} < 0 \\
h''(x) &= -\sum_{i\in[\![d_1]\!]} \frac{2\alpha_i\|r\|^2}{(\|r\|\cdot x + \frac{1+\alpha_i}{\alpha_i})^2} - \frac{-\frac{2}{x^2}(\|r\|\cdot x + 2) - 2(\|r\| + \frac{1}{x})\|r\|}{(\|r\|\cdot x + 2)^2} \\
&= -\sum_{i\in[\![d_1]\!]} \frac{2\alpha_i^3}{(\alpha_i x + \frac{1+\alpha_i}{\|r\|})^2} + \frac{\frac{4\|r\|}{x} + \frac{4}{x^2} + 2\|r\|^2}{(\|r\|\cdot x + 2)^2}.
\end{aligned}$$

Since $h'(x) < 0$ holds for $x > 0$ and $h(\hat{x}) > 0$, the root of h is unique over $x > 0$.

Consider next the function $g(\beta) = -\frac{\beta^3}{\left(\beta x + \frac{1+\beta}{\|r\|}\right)^2}$, which is concave over the interval $[0, 1]$ when $x > 0$. We also have $g(0) = 0$, which implies

$$g(t\beta) \geq tg(\beta) + (1-t)g(0) = tg(\beta),\ \forall t, \beta \in [0, 1].$$

Therefore,

$$h''(x) = 2\sum_{i\in[\![d_1]\!]} g(\alpha_i) + \frac{\frac{4\|r\|}{x} + \frac{4}{x^2} + 2\|r\|^2}{(\|r\|\cdot x + 2)^2} \geq 2\sum_{i\in[\![d_1]\!]} \alpha_i g(1) + \frac{\frac{4\|r\|}{x} + \frac{4}{x^2} + 2\|r\|^2}{(\|r\|\cdot x + 2)^2}$$
$$= -\frac{2}{(x + \frac{2}{\|r\|})^2} + \frac{\frac{4\|r\|}{x} + \frac{4}{x^2} + 2\|r\|^2}{(\|r\|\cdot x + 2)^2} = \frac{\frac{4\|r\|}{x} + \frac{4}{x^2}}{(\|r\|\cdot x + 2)^2} > 0$$

shows h is convex over $x > 0$. Since h is convex and decreasing, the root of $h(x)$ is unique and a root finding Newton-Raphson method converges quadratically if it starts with $0 < x_0 \leq \hat{x}$ where $h(\hat{x}) > 0$ [168, Thm. 1.9]. Since $q(t) = \ln\left(\|r\|\cdot x + \frac{1}{t} + 1\right)$ is convex when $t \in [0,1], x \geq 0$, we can apply the Jensen's inequality and obtain

$$h(x) \geq 2\ln(\|r\|\cdot x + \frac{d_1}{\sum_{i\in[\![d_1]\!]}\alpha_i} + 1) - \ln\left(\frac{2x}{\|r\|} + x^2\right) - \sum_{i\in[\![d_1]\!]} 2\alpha_i \ln(p_i),$$

where a positive root of the right part above is

$$\hat{x} = -\frac{1}{\|r\|} + \frac{d_1\|r\| + \sqrt{\chi\left(\frac{\chi}{\|r\|^2} + d_1^2 - 1\right)}}{\chi - \|r\|^2}$$

with $\chi = \prod_{i\in[\![d_1]\!]} p_i^{2\alpha_i}$, which implies $h(\hat{x}) \geq 0$ and we can set it as the starting point of the Newton-Raphson method. After obtaining $g_{\|r\|}$ from (A.13), we get $\zeta = \frac{2g_{\|r\|}}{\|r\|}$ and the primal conjugate gradient

$$g_r = \frac{g_{\|r\|}\cdot r}{\|r\|}, \quad g_{p_i} = -\frac{1}{p_i\zeta}\left((1-\alpha_i)\zeta + 2\alpha_i(\zeta + g_{\|r\|}^2)\right) = -\frac{1}{p_i}(1 + \alpha_i + \alpha_i g_{\|r\|}\|r\|),$$

via (A.16) and (A.18). □

Remark A.2.1. For the conjugate gradient of a dual power cone $\mathcal{K}_{\text{pow}}^*$ in (4.7), it is the special case of the dual generalized power cone $\mathcal{K}_{\text{gpow }(\alpha,d_1,d_2)}^*$ where $d_1 = 2, d_2 = 1, \alpha_1 = \alpha, \alpha_2 = 1-\alpha$ and Theorem A.2.1 applies directly.

A.3 Jordan algebra for symmetric cones

Jordan algebra is needed when the NT scaling is used for an IPM with symmetric cones. Relevant computation includes product $u \circ v$, inverse of $u \circ v$ and idempotents $\mathbf{e}$ for each symmetric cone [41].

A.3.1 Product $u \circ v$

$$u \circ v = \begin{cases} (u_1 v_1, \ldots, u_n v_n) & u, v \in \mathbb{R}^n_+ \\ \left(u^\top v, u_0 v_1 + v_0 u_1\right) & u, v \in \mathcal{K}^n_{\text{soc}} \\ \frac{1}{2}\text{vec}(\text{mat}(u)\text{mat}(v) + \text{mat}(v)\text{mat}(u)) & \text{mat}(u), \text{mat}(v) \in \mathbb{S}^n_+ \end{cases}$$

Some useful properties of the $\circ$ product are the inverse $u^{-1} \circ u = \mathbf{e}$, square $u^2 = u \circ u$, and square root $u^{1/2} \circ u^{1/2} = u$.

A.3.2 Inverse of $u \circ v$

We denote the inverse of $u \circ v$ as $u \backslash v$, which is the solution x of the equation

$$u \circ x = v.$$

It is used in (4.28b) where u is always set to $\lambda := Wz = W^{-\top}s$ defined in Appendix A.4. Hence, the operation $\lambda \backslash v$ for different cones is:

$$\lambda \backslash v = \begin{cases} \text{diag}\,(\lambda)^{-1} v, & \lambda, v \in \mathbb{R}^n_+ \\ \frac{1}{\lambda_0^2 - \lambda_1^T \lambda_1} \begin{bmatrix} \lambda_0 & -\lambda_1^T \\ -\lambda_1 & \lambda_0^{-1}\left(\left(\lambda_0^2 - \lambda_1^T\lambda_1\right) I + \lambda_1 \lambda_1^T\right) \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \end{bmatrix}, & \lambda, v \in \mathcal{K}^n_{\text{soc}} \\ \text{vec}(\text{mat}(v) \odot \Gamma), & \text{mat}(\lambda), \text{mat}(v) \in \mathbb{S}^n_+ \end{cases}$$

where $\Gamma_{ij} = 2/\left(\Lambda_{ii} + \Lambda_{jj}\right), \Lambda = \text{mat}(\lambda)$ and $\odot$ denotes the Hadamard (element-wise) matrix product. Note that for the affine step where $d_s = \lambda \circ \lambda$,

$$W^T\left(\lambda \backslash d_s\right) = W^T \lambda = s,$$

which is consistent with (4.27b).

A.3.3 Idempotents e

$$\mathbf{e} = \begin{cases} (1, 1, \ldots, 1), & \mathcal{K} = \mathbb{R}^n_+ \\ (1, 0, \ldots, 0), & \mathcal{K} = \mathcal{K}^n_{\text{soc}} \\ \text{vec}\left(I_n\right), & \mathcal{K} = \mathbb{S}^n_+ \end{cases}$$

Note that $\mathbf{e}^T(z \circ s) = z^T s$ and $\mathbf{e}^T \mathbf{e} = \nu$.

A.4 Nesterov-Todd scaling for symmetric cones

The scaling points w and matrices W, with the scaled variable $\lambda = W^{-1}s = Wz$, for different symmetric cones at the primal-dual pair (s, z) are listed below, which can be found in [41]:

Nonnegative cone

We have

$$\begin{aligned} w &= s^{1/2} \circ z^{-1/2}, \\ W &= \mathrm{diag}(w) = \mathrm{diag}(s^{1/2} \circ z^{-1/2}), \\ \lambda &= s^{1/2} \circ z^{1/2}. \end{aligned}$$

Second-order cone

We define

$$\bar{z} = \frac{z}{\sqrt{z^\top J z}},\ \bar{s} = \frac{s}{\sqrt{s^\top J s}},\ \gamma = \left(\frac{1+\bar{z}^\top \bar{s}}{2}\right)^{1/2}, \bar{w} = \frac{1}{2\gamma}(\bar{s} + J\bar{z}), \eta = \left(\frac{s^\top J s}{z^\top J z}\right)^{1/4},$$

where $J := \mathrm{diag}([1; -\mathbf{1}^{n-1}])$. we have $\bar{w}^\top J \bar{w} = 1$. Moreover, the scaled scaling matrix $\bar{W}$ corresponds to $\bar{w}$ is

$$\bar{W} = \begin{bmatrix} \bar{w}_0 & \bar{w}_1^T \\ \bar{w}_1 & I + (\bar{w}_0+1)^{-1}\bar{w}_1\bar{w}_1^T \end{bmatrix}, \quad \bar{W}^{-1} = \begin{bmatrix} \bar{w}_0 & -\bar{w}_1^T \\ -\bar{w}_1 & I + (\bar{w}_0+1)^{-1}\bar{w}_1\bar{w}_1^T \end{bmatrix},$$

and we have

$$\bar{\lambda} = \bar{W}\bar{z} = \bar{W}^{-1}\bar{s} = \begin{bmatrix} \gamma \\ \frac{1}{\bar{s}_0+\bar{z}_0+2\gamma}\left((\gamma+\bar{z}_0)\bar{s}_1 + (\gamma+\bar{s}_0)\bar{z}_1\right) \end{bmatrix}.$$

Finally, we obtain w, W, λ

$$w = \eta\bar{w},\ W = \eta\bar{W},\ \lambda = \eta\bar{\lambda}.$$

Positive semidefinite cone

Suppose we define $S = \mathrm{mat}(s), Z = \mathrm{mat}(z)$ and compute the Cholesky factorizations of S, Z

$$S = L_1 L_1^\top, \quad Z = L_2 L_2^\top,$$

and then the SVD of

$$L_2^\top L_1 = U \Lambda V^\top.$$

The scaled variable is

$$\lambda = \mathrm{vec}(\Lambda)$$

and the scaling point w is

$$w = \text{vec}\left(S^{1/2}(S^{1/2}ZS^{1/2})^{-1/2}S^{1/2}\right).$$

Matrix W can be regarded as a congruence transformation such that

$$\text{mat}(\lambda) = Wz = \text{vec}(R^\top \text{mat}(z)R), \quad W^{-\top}s = \text{vec}(R^{-1}\text{mat}(s)R^{-\top}),$$

where R can be obtained from

$$R = L_1 V \Lambda^{1/2} = L_2^{-\top} U \Lambda^{1/2}.$$

Likewise, R^{-1} is given by

$$R^{-1} = \Lambda^{1/2} V^\top L_1^{-1} = \Lambda^{-1/2} U^\top L_2^\top.$$

A.5 Sparse LDL factorization for SOCPs

The scaling matrix of a second-order cone has a special structure as follows:

$$H_{soc} = \eta^2(D + uu^\top - vv^\top),$$

where

$$D = \begin{bmatrix} d & \\ & I_{m-1} \end{bmatrix}, \; u = \begin{bmatrix} u_0 \\ u_1\bar{w}_1 \end{bmatrix}, \; v = \begin{bmatrix} 0 \\ v_1\bar{w}_1 \end{bmatrix},$$
$$d = \frac{1}{2\|\bar{w}\|^2}, u_0 = \sqrt{\|\bar{w}\|^2 - d}, u_1 = \frac{2\bar{w}_0}{u_0}, v_1 = \sqrt{u_1^2 - 2}.$$

It can be proved that $D - vv^\top \succ 0$, and the second line in (4.19) is equivalent to an expanded linear system

$$\begin{bmatrix} A \\ 0 \\ 0 \end{bmatrix} \Delta x - \eta^2 \underbrace{\begin{bmatrix} D & v & u \\ v^\top & 1 & 0 \\ u^\top & 0 & -1 \end{bmatrix}}_{K_{aug}:=} \begin{bmatrix} \Delta z_1 \\ q \\ t \end{bmatrix} = \begin{bmatrix} d_s - d_z \\ 0 \\ 0 \end{bmatrix}.$$

Since K_{aug} is proved quasi-definite [45], the expanded system of (4.19) with K_{aug} is also quasi-definite [107] and the sparse LDL factorization still applies.

A.6 Higher-order corrections for nonsymmetric cones

The analysis of 3rd-order correction for nonsymmetric cones comes from [92]. Since the pair (s_μ, z_μ) on the central path is parametrized by μ, we can take the second derivative of $s_\mu = -\mu\nabla f^*(z_\mu)$ w.r.t. μ, which becomes

$$\dot{s}_\mu + \mu\nabla^2 f^*(z_\mu)\,\dot{z}_\mu = -\nabla f^*(z_\mu), \tag{A.19a}$$

$$\ddot{s}_\mu + \mu\nabla^2 f^*(z_\mu)\,\ddot{z}_\mu = -2\nabla^2 f^*(z_\mu)\,\dot{z}_\mu - \mu\nabla^3 f^*(z_\mu)[\dot{z}_\mu, \dot{z}_\mu] \tag{A.19b}$$

We can rewrite (A.19a) as

$$\mu\dot{z}_\mu = -\left[\nabla^2 f^*(z_\mu)\right]^{-1}\left(\nabla f^*(z_\mu) + \dot{s}_\mu\right) = z_\mu - \left[\nabla^2 f^*(z_\mu)\right]^{-1}\dot{s}_\mu.$$

Substituting the equation above into (A.19b), we have

$$\begin{aligned}\mu\nabla^3 f^*(z_\mu)[\dot{z}_\mu, \dot{z}_\mu] &= \nabla^3 f^*(z_\mu)[\dot{z}_\mu, z_\mu] - \nabla^3 f^*(z_\mu)\left[\dot{z}_\mu, \left(\nabla^2 f^*(z_\mu)\right)^{-1}\dot{s}_\mu\right] \\ &= -2\nabla^2 f^*(z_\mu)\,\dot{z}_\mu - \nabla^3 f^*(z_\mu)\left[\dot{z}_\mu, \left(\nabla^2 f^*(z_\mu)\right)^{-1}\dot{s}_\mu\right]\end{aligned} \tag{A.20}$$

where the last equation follows from the homogeneity property $\nabla^3 f^*(z_\mu)[z_\mu] = -2\nabla^2 f^*(z_\mu)$ in (2.22), and (A.19b) can then be rewritten as,

$$\ddot{s}_\mu + \mu\nabla^2 f^*(z_\mu)\,\ddot{z}_\mu = \nabla^3 f^*(z_\mu)\left[\dot{z}_\mu, \left(\nabla^2 f^*(z_\mu)\right)^{-1}\dot{s}_\mu\right].$$

Setting $\Delta s_a = -\dot{s}_\mu/\mu$ and $\Delta z_a = -\dot{z}_\mu/\mu$, yields the 3rd-order correction (4.23).

We summarize the computation of 3rd-order derivatives and the inverse of Hessians for the use of 3rd-order correction (4.23) in multi-dimensional power cones as follows:

A.6.1 Generalized power cone

Note that $\varphi = \prod_{i\in[\![d_1]\!]} u_i^{2\alpha_i}, \tau_i = \frac{2\alpha_i}{u_i}, \forall i \in [\![d_1]\!]$, and $\zeta = \prod_{i\in[\![d_1]\!]} u_i^{2\alpha_i} - \|w\|^2$. The 3rd-order derivative is

$$\begin{aligned}
\nabla^3_{u_i,u_j,u_k} f_{\text{gpow}} &= \tau_i\tau_j\tau_k \frac{\varphi}{\zeta}\left(1-\frac{\varphi}{\zeta}\right)\left(\frac{2\varphi}{\zeta}-1\right) + \begin{cases} 2\frac{\tau_i\tau_j}{u_i}\frac{\varphi}{\zeta}\left(1-\frac{\varphi}{\zeta}\right), & i=j=k \\ \frac{\tau_i\tau_j}{u_i}\frac{\varphi}{\zeta}\left(1-\frac{\varphi}{\zeta}\right), & i=k\neq j \\ \frac{\tau_i\tau_j}{u_j}\frac{\varphi}{\zeta}\left(1-\frac{\varphi}{\zeta}\right), & i\neq j=k \\ 0, & \text{otherwise} \end{cases} \\
&\quad + \begin{cases} \frac{\tau_i^2\varphi}{u_i\zeta}\left(1-\frac{\varphi}{\zeta}\right) - (1-\alpha_i)\frac{2}{u_i^3} - \frac{2\varphi\tau_i}{\zeta u_i^2}, & i=j=k \\ \frac{\tau_i\tau_k\varphi}{u_i\zeta}\left(1-\frac{\varphi}{\zeta}\right), & i=j\neq k \\ 0, & \text{otherwise} \end{cases}, \\
\nabla^3_{u_i,u_j,w_k} f_{\text{gpow}} &= \frac{2\tau_i\tau_j\varphi w_k}{\zeta^2}\left(\frac{2\varphi}{\zeta}-1\right) + \delta(i,j)\frac{2\tau_i\varphi w_k}{\zeta^2 u_i}, \\
\nabla^3_{u_i,w_j,w_k} f_{\text{gpow}} &= \frac{-8w_jw_k\tau_i\varphi}{\zeta^3} - \delta(j,k)\frac{2}{\zeta^2}\tau_i\varphi, \\
\nabla^3_{w_i,w_j,w_k} f_{\text{gpow}} &= \frac{16w_iw_jw_k}{\zeta^3} + \delta(i,j)\frac{4w_k}{\zeta^2} + \begin{cases} \frac{8w_i}{\zeta^2}, & i=j=k \\ \frac{4w_j}{\zeta^2}, & i=k\neq j \\ \frac{4w_i}{\zeta^2}, & i\neq j=k \\ 0, & \text{otherwise} \end{cases}.
\end{aligned}$$

The inverse of the Hessian is

$$\begin{aligned}
\nabla^{-2}_{u_i,u_j} f_{\text{gpow}} &= -\delta(i,j)\frac{u_i}{\nabla_{u_i} f_{\text{gpow}}} - \frac{4\|w\|^2}{k_2\zeta}\frac{\alpha_i}{\nabla_{u_i} f_{\text{gpow}}}\frac{\alpha_j}{\nabla_{u_j} f_{\text{gpow}}}, \\
\nabla^{-2}_{u_i,w_j} f_{\text{gpow}} &= \frac{2}{k_2}\frac{\alpha_i}{\nabla_{u_i} f_{\text{gpow}}} w_j, \\
\nabla^{-2}_{w_i,w_j} f_{\text{gpow}} &= \delta(i,j)\frac{\zeta}{2} + \frac{(\zeta^{-1}\varphi + 4k_1)}{k_2} w_iw_j,
\end{aligned}$$

where

$$k_1 = \left\langle \frac{\alpha}{\nabla_u f}, \frac{\alpha}{u} \right\rangle, \quad k_2 = -\left(1 + \frac{\|w\|^2}{\varphi}\right) - \frac{4k_1\|w\|^2}{\zeta}.$$

A.6.2 Power mean cone

Note that $\varphi = \prod_{i\in[\![d]\!]} u_i^{\alpha_i}, \zeta = \varphi - w$ and $\tau \in \mathbb{R}^d$ with $\tau_i = \frac{\alpha_i}{u_i\zeta}, \forall i \in [\![d]\!]$. The 3rd-order derivative is

$$\nabla^3_{u_i,u_j,u_k} f_{\text{powm}} = -w\varphi\tau_i\tau_j\tau_k(2w+\zeta) + \begin{cases} -\frac{w\varphi\tau_i\tau_k}{u_i} - \frac{2\varphi\tau_i}{u_i^2} - \frac{2}{u_i^3}, & i=j=k \\ -\frac{w\varphi\tau_i\tau_k}{u_i}, & i=j\neq k \\ 0, & \text{otherwise} \end{cases}$$
$$+ \begin{cases} -\frac{2w\varphi\tau_i\tau_j}{u_i}, & i=j=k \\ -\frac{w\varphi\tau_i\tau_j}{u_i}, & i=k\neq j \\ -\frac{w\varphi\tau_i\tau_j}{u_j}, & i\neq j=k \\ 0, & \text{otherwise} \end{cases},$$
$$\nabla^3_{u_i,u_j,w} f_{\text{powm}} = \varphi\tau_i\tau_j\left(\frac{2\varphi}{\zeta} - 1\right) + \delta(i,j)\frac{\tau_i\varphi}{u_i\zeta},$$
$$\nabla^3_{u_i,w,w} f_{\text{powm}} = -\frac{2\varphi\tau_i}{\zeta^2},$$
$$\nabla^3_{w,w,w} f_{\text{powm}} = \frac{2}{\zeta^3},$$

The inverse of the Hessian is

$$\nabla^{-2}_{u_i,u_j} f_{\text{powm}} = \delta(i,j)\frac{u_i^2}{s_{0,i}} + \frac{\varphi}{\zeta s_2}\frac{\alpha_i u_i}{s_{0,i}}\frac{\alpha_j u_j}{s_{0,j}},$$
$$\nabla^{-2}_{u_i,w} f_{\text{powm}} = \frac{\varphi}{s_2}\frac{\alpha_i u_i}{s_{0,i}},$$
$$\nabla^{-2}_{w,w} f_{\text{powm}} = \zeta^2 + \frac{s_1}{s_2}\varphi^2,$$

where

$$s_{0,i} = 1 + \alpha_i\varphi\zeta^{-1}, \forall i \in [\![d]\!],$$
$$s_1 = \sum_{i\in[\![d]\!]} \frac{\alpha_i^2}{s_{0,i}},$$
$$s_2 = 1 - \varphi\zeta^{-1}s_1.$$

A.7 Detailed benchmark results for Clarabel

Table A.1: Solve times and iteration counts for the Maros-Meszaros problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
TAME	3	2	4	3	5	6	3	1.16e-05	7.36e-06	0.000154	5.81e-05	4.41e-05	0.000461
HS21	5	2	6	2	9	11	11	6.19e-06	6.63e-06	0.000105	5.57e-05	7.29e-05	0.00115
HS35	4	3	6	5	7	9	9	7.64e-06	9.51e-06	0.000105	5.35e-05	8.56e-05	0.000944
HS35MOD	4	3	6	5	12	15	14	5.37e-06	7.67e-06	9.91e-05	6.45e-05	0.000115	0.00139
QPTEST	5	2	7	3	8	9	8	8.01e-06	9.51e-06	0.000158	6.41e-05	8.56e-05	0.00126
HS51	3	5	7	7	0	9	9	0	8.83e-06	0.000104	3.2e-05	7.95e-05	0.000934
HS52	3	5	7	7	0	7	5	0	1.06e-05	0.000121	5.51e-05	7.43e-05	0.000607
ZECEVIC2	6	2	8	1	8	8	7	8.3e-06	7.49e-06	0.000115	6.64e-05	6e-05	0.000806
HS268	5	5	25	15	12	17	-	7.78e-06	1.24e-05	-	9.33e-05	0.000211	-
S268	5	5	25	15	12	17	-	8.49e-06	1.23e-05	-	0.000102	0.00021	-
HS76	7	4	14	6	6	9	8	1.05e-05	1.08e-05	0.000115	6.31e-05	9.69e-05	0.000919
GENHS28	8	10	24	19	0	10	6	0	1.3e-05	0.00014	4.27e-05	0.00013	0.000842
HS53	13	5	17	7	6	15	7	9.67e-06	8.26e-06	0.000133	5.8e-05	0.000124	0.00093
LOTSCHD	19	12	66	6	9	19	19	1.52e-05	1.9e-05	0.000382	0.000137	0.000361	0.00727
HS118	59	15	93	15	11	10	12	2.19e-05	3.16e-05	0.000134	0.000241	0.000316	0.00161
QAFIRO	59	32	115	6	14	14	13	2.68e-05	2.9e-05	0.000185	0.000376	0.000407	0.00241
DPKLO1	77	133	1575	77	0	14	6	0	0.000317	0.00143	0.000602	0.00443	0.00856
DUAL4	151	75	225	2799	12	12	14	0.000287	0.000391	0.000601	0.00345	0.0047	0.00842
QPCBLEND	157	83	574	83	17	18	15	0.0001	0.000137	0.000983	0.00171	0.00246	0.0148
DUALC1	233	9	1953	45	12	24	-	0.000147	0.00014	-	0.00176	0.00336	-
DUALC2	243	7	1617	28	11	-	-	0.000122	-	-	0.00134	-	-
QADLITTL	153	97	480	87	14	27	-	9.36e-05	9.02e-05	-	0.00131	0.00244	-
QSHARE2B	175	79	773	55	16	27	-	0.000125	0.000144	-	0.002	0.00389	-
DUAL1	171	85	255	3558	12	15	11	0.000374	0.00044	0.0007	0.00448	0.0066	0.0077
DUAL2	193	96	288	4508	11	15	13	0.000489	0.000576	0.000796	0.00538	0.00864	0.0104
DUALC5	294	8	2240	36	10	17	15	0.000165	0.000206	0.000185	0.00165	0.00351	0.00278
CVXQP2_S	225	100	274	386	10	-	-	0.000138	-	-	0.00138	-	-
DUAL3	223	111	333	6108	12	19	12	0.000673	0.000836	0.00224	0.00808	0.0159	0.0269
CVXQP1_S	250	100	348	386	9	-	-	0.000175	-	-	0.00157	-	-
CVXQP3_S	275	100	422	386	11	-	-	0.000204	-	-	0.00224	-	-
QSCAGR7	269	140	560	25	16	-	-	0.000121	-	-	0.00194	-	-
PRIMAL1	86	325	5816	324	10	20	10	0.000912	0.000898	0.00297	0.00912	0.018	0.0297
QRECIPE	340	180	912	50	17	21	23	0.00015	0.000151	0.00137	0.00256	0.00318	0.0316
QPCBOEI2	382	143	1480	143	20	-	-	0.000328	-	-	0.00657	-	-
DUALC8	519	8	4040	36	9	-	-	0.000304	-	-	0.00273	-	-
QSHARE1B	342	225	1376	39	32	-	32	0.000203	-	0.00174	0.0065	-	0.0557
PRIMALC5	286	287	2574	286	14	14	14	0.000234	0.000319	0.00268	0.00328	0.00446	0.0376
VALUES	405	202	606	3822	13	-	-	0.000376	-	-	0.00489	-	-
QSC205	408	203	754	21	19	18	18	0.000161	0.000166	0.000662	0.00307	0.00299	0.0119

Table A.1: Solve times and iteration counts for the Maros-Meszaros problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
QBEACONF	435	262	3637	27	28	28	21	0.000496	0.000505	0.00179	0.0139	0.0141	0.0375
QBRANDY	469	249	2397	65	19	32	-	0.000466	0.000412	-	0.00884	0.0132	-
PRIMAL2	97	649	8043	648	8	16	8	0.00146	0.00148	0.00332	0.0117	0.0237	0.0265
QE226	505	282	2860	964	24	26	22	0.000645	0.000874	0.00271	0.0155	0.0227	0.0596
PRIMAL3	112	745	21548	744	9	18	10	0.004	0.00308	0.00622	0.036	0.0554	0.0622
QBORE3D	559	315	1755	78	27	25	21	0.000372	0.000388	0.00249	0.01	0.00969	0.0523
KSIP	1001	20	19898	20	14	20	12	0.00135	0.00121	0.000778	0.0189	0.0242	0.00934
QGROW7	721	301	3193	357	22	22	-	0.000566	0.00063	-	0.0125	0.0139	-
QFORPLAN	604	421	5006	582	21	-	-	0.000817	-	-	0.0172	-	-
PRIMALC8	511	520	4663	519	12	-	-	0.000574	-	-	0.00689	-	-
QCAPRI	741	353	2237	894	32	-	-	0.000672	-	-	0.0215	-	-
QSCORPIO	746	358	1784	40	11	15	16	0.000353	0.00046	0.00305	0.00388	0.0069	0.0487
QSCFXM1	787	457	3046	733	26	-	-	0.00072	-	-	0.0187	-	-
QBANDM	777	472	2966	41	21	37	-	0.000602	0.000618	-	0.0126	0.0229	-
QSCTAP1	780	480	2172	153	19	24	24	0.000405	0.000469	0.00274	0.0077	0.0113	0.0659
QPCSTAIR	823	467	4323	467	22	-	-	0.000988	-	-	0.0217	-	-
QSTAIR	823	467	4323	1018	31	-	-	0.00124	-	-	0.0386	-	-
QPCBOEI1	980	384	4359	384	17	42	-	0.000906	0.00102	-	0.0154	0.0429	-
QSCAGR25	971	500	2054	128	20	-	-	0.000395	-	-	0.0079	-	-
PRIMAL4	76	1489	16032	1488	10	21	13	0.00304	0.00309	0.0063	0.0304	0.065	0.0819
QSCSD1	837	760	3148	745	10	12	12	0.0006	0.000702	0.00244	0.006	0.00843	0.0293
QETAMACR	1223	688	3232	4447	20	33	-	0.00467	0.00458	-	0.0934	0.151	-
QGROW15	1545	645	6865	500	22	22	-	0.00115	0.00121	-	0.0253	0.0266	-
QFFFFF80	1378	854	7081	1916	27	52	-	0.00247	0.00263	-	0.0668	0.137	-
MOSARQP2	1500	900	3830	945	10	22	18	0.00143	0.00189	0.0101	0.0143	0.0416	0.181
GOULDQP2	1747	699	2445	697	14	-	-	0.000596	-	-	0.00835	-	-
GOULDQP3	1747	699	2445	1395	7	-	-	0.00067	-	-	0.00469	-	-
QSCFXM2	1574	914	6097	1131	32	-	-	0.00143	-	-	0.0458	-	-
QSTANDAT	1538	1075	4210	804	18	28	20	0.000859	0.000826	0.00856	0.0155	0.0231	0.171
QSCRS8	1659	1169	4351	121	33	30	31	0.000978	0.00102	0.0061	0.0323	0.0305	0.189
QSCSD6	1497	1350	5666	1404	13	17	19	0.00102	0.00129	0.00465	0.0133	0.0219	0.0884
LASER	2000	1002	6000	3231	11	28	-	0.000896	0.0013	-	0.00985	0.0363	-
QGFRDXPN	1966	1092	3727	162	22	-	-	0.000811	-	-	0.0178	-	-
QSEBA	2057	1028	5902	646	30	-	-	0.00107	-	-	0.032	-	-
QGROW22	2266	946	10078	852	27	30	-	0.00167	0.00168	-	0.0451	0.0503	-
CVXQP2_M	2250	1000	2749	3984	10	-	-	0.00502	-	-	0.0502	-	-
QSHIP04S	1860	1458	5810	56	15	35	-	0.00101	0.000906	-	0.0151	0.0317	-
CVXQP1_M	2500	1000	3498	3984	10	-	-	0.0077	-	-	0.077	-	-
QSCFXM3	2361	1371	9148	1221	34	-	-	0.00202	-	-	0.0688	-	-

Table A.1: Solve times and iteration counts for the Maros-Meszaros problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
CVXQP3_M	2750	1000	4247	3984	12	-	-	0.00892	-	-	0.107	-	-
Q25FV47	2391	1571	11971	59499	26	-	-	0.0135	-	-	0.351	-	-
QSHELL	2428	1775	5448	34790	35	-	-	0.00698	-	-	0.244	-	-
QSHIP04L	2520	2118	8450	56	15	39	-	0.0014	0.00126	-	0.021	0.049	-
QSCTAP2	2970	1880	8594	777	12	20	15	0.00202	0.00247	0.0115	0.0242	0.0495	0.172
AUG3D	1000	3873	6546	2673	0	-	10	0	-	0.0186	0.00577	-	0.186
AUG3DC	1000	3873	6546	3873	0	-	-	0	-	-	0.00582	-	-
QSHIP08S	3165	2387	9501	11677	15	36	-	0.00398	0.00535	-	0.0598	0.193	-
QPILOTNO	3487	2172	15569	485	31	-	-	0.00652	-	-	0.202	-	-
MOSARQP1	3200	2500	5922	2545	10	23	19	0.00192	0.00334	0.0325	0.0192	0.0769	0.618
QSCSD8	3147	2750	11334	2510	11	18	17	0.00213	0.00348	0.0158	0.0235	0.0627	0.269
QSCTAP3	3960	2480	11354	1047	12	22	-	0.00255	0.00355	-	0.0306	0.078	-
QSHIP12S	3914	2763	10941	17403	15	39	-	0.00434	0.0057	-	0.0652	0.222	-
QSIERRA	5279	2036	11354	183	35	51	-	0.00203	0.00224	-	0.0709	0.114	-
STADAT1	5999	2001	13997	2000	15	-	-	0.00176	-	-	0.0265	-	-
STADAT2	5999	2001	13997	2000	45	22	23	0.00193	0.0052	0.0119	0.0867	0.114	0.275
AUG3DCQP	4873	3873	10419	3873	11	29	-	0.00407	0.00668	-	0.0448	0.194	-
AUG3DQP	4873	3873	10419	2673	13	-	-	0.0039	-	-	0.0507	-	-
QSHIP08L	5061	4283	17085	34965	16	36	-	0.0165	0.0229	-	0.264	0.826	-
CONT-050	7595	2597	17199	2597	9	21	20	0.00992	0.0109	0.0425	0.0893	0.229	0.85
EXDATA	7501	3000	12000	1125750	22	21	16	0.548	0.859	0.127	12	18	2.03
QSHIP12L	6578	5427	21597	62228	16	36	-	0.0246	0.0348	-	0.394	1.25	-
STCQP1	10246	4097	21532	26603	7	-	-	0.0127	-	-	0.0892	-	-
STCQP2	10246	4097	21532	26603	9	-	-	0.0248	-	-	0.223	-	-
STADAT3	11999	4001	27997	4000	54	18	24	0.00379	0.00507	0.0237	0.205	0.0912	0.569
HUES-MOD	10002	10000	30000	10000	13	-	-	0.00595	-	-	0.0774	-	-
HUESTIS	10002	10000	30000	10000	9	-	-	0.0111	-	-	0.1	-	-
LISWET2	10000	10002	30000	10002	18	-	-	0.00483	-	-	0.0869	-	-
LISWET3	10000	10002	30000	10002	23	-	-	0.00474	-	-	0.109	-	-
LISWET4	10000	10002	30000	10002	27	-	-	0.00452	-	-	0.122	-	-
LISWET5	10000	10002	30000	10002	11	-	-	0.0055	-	-	0.0605	-	-
LISWET6	10000	10002	30000	10002	18	-	-	0.00485	-	-	0.0873	-	-
DTOC3	10000	14999	34995	14997	0	-	9	0	-	0.067	0.0168	-	0.603
AUG2D	10000	20200	40000	19800	0	-	-	0	-	-	0.0327	-	-
AUG2DC	10000	20200	40000	20200	0	-	-	0	-	-	0.0334	-	-
CVXQP2_L	22500	10000	27499	39984	10	-	-	1.77	-	-	17.7	-	-
CVXQP1_L	25000	10000	34998	39984	11	-	-	3.14	-	-	34.5	-	-
CVXQP3_L	27500	10000	42497	39984	11	-	-	3.52	-	-	38.8	-	-
CONT-100	30195	10197	69399	10197	9	19	14	0.076	0.0866	0.167	0.684	1.64	2.34

Table A.1: Solve times and iteration counts for the Maros-Meszaros problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
CONT-101	30492	10197	69993	2700	12	-	14	0.0761	-	0.143	0.913	-	2.01
UBH1	24018	18009	60018	6003	16	-	35	0.00949	-	0.0979	0.152	-	3.43
AUG2DCQP	30200	20200	60200	20200	12	-	-	0.0236	-	-	0.283	-	-
AUG2DQP	30200	20200	60200	19800	14	-	-	0.0231	-	-	0.324	-	-
CONT-200	120395	40397	278799	40397	13	25	16	0.572	0.744	0.918	7.43	18.6	14.7
CONT-201	120992	40397	279993	10400	12	-	16	0.743	-	0.608	8.92	-	9.73
BOYD1	93279	93261	652246	93261	35	-	-	0.0996	-	-	3.49	-	-
CONT-300	271492	90597	629993	23100	12	-	13	2.86	-	1.6	34.4	-	20.8

Table A.2: Solve times and iteration counts for the Optimal Control problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
pendulum_3	28	23	83	23	5	11	11	2.71e-05	3.17e-05	0.000182	0.000136	0.000349	0.002
fiordosExample_1	32	27	67	18	8	12	12	1.93e-05	3.05e-05	0.000188	0.000155	0.000367	0.00226
fiordosExample_3	32	27	67	18	8	-	-	2.07e-05	-	-	0.000166	-	-
forcesExample_1	38	29	83	28	9	14	11	2.07e-05	3.59e-05	0.000235	0.000186	0.000502	0.00258
forcesExample_2	38	29	83	27	8	13	10	2.38e-05	3.57e-05	0.000244	0.00019	0.000464	0.00244
forcesExample_3	42	32	92	31	9	12	11	2.32e-05	4.21e-05	0.00024	0.000209	0.000506	0.00264
toyExample_1	42	32	102	31	7	-	16	2.57e-05	-	0.000318	0.00018	-	0.00509
toyExample_3	42	32	102	31	9	-	-	2.31e-05	-	-	0.000208	-	-
helicopter_1	48	30	96	39	6	-	21	3.39e-05	-	0.000298	0.000204	-	0.00626
fiordosExample_2	52	27	87	18	7	13	13	2.74e-05	3.3e-05	0.000188	0.000192	0.000428	0.00244
nonlinearCstr_3	64	54	204	30	10	-	12	3.94e-05	-	0.000294	0.000394	-	0.00353
robotArm_1	84	54	174	34	6	-	27	6.23e-05	-	0.000468	0.000374	-	0.0126
pendulum_1	78	63	243	63	6	12	6	5.89e-05	8.41e-05	0.000444	0.000353	0.00101	0.00266
pendulum_2	78	63	243	60	7	14	8	5.38e-05	7.82e-05	0.000406	0.000377	0.0011	0.00325
forcesExample_4	82	62	182	61	7	15	7	4.53e-05	6.95e-05	0.000468	0.000317	0.00104	0.00327
toyExample_2	82	62	202	61	7	18	10	4.54e-05	6.3e-05	0.000476	0.000318	0.00113	0.00476
aircraft_3	104	84	364	20	8	16	12	7.14e-05	0.000113	0.000326	0.000571	0.0018	0.00391
doubleInvertedPendulum_1	104	84	364	50	7	14	9	8.58e-05	0.0001	0.00051	0.000601	0.0014	0.00459
doubleInvertedPendulum_2	104	84	364	50	8	15	10	8.92e-05	8.65e-05	0.000457	0.000714	0.0013	0.00457
helicopter_3	118	86	278	80	7	22	-	8.09e-05	8.37e-05	-	0.000567	0.00184	-
aircraft_1	144	84	404	20	9	10	9	6.6e-05	0.000115	0.000462	0.000594	0.00115	0.00416
aircraft_2	144	84	404	40	9	10	9	6.68e-05	0.000133	0.000525	0.000601	0.00133	0.00472
aircraft_4	144	84	404	20	9	10	9	6.7e-05	0.000114	0.000459	0.000603	0.00114	0.00413
doubleInvertedPendulum_3	144	84	404	50	6	18	8	0.000105	9.56e-05	0.000544	0.000632	0.00172	0.00435
ballOnPlate_1	152	77	257	47	8	15	10	7.75e-05	8.85e-05	0.000745	0.00062	0.00133	0.00745
ballOnPlate_2	152	77	257	47	7	14	7	7.83e-05	8.94e-05	0.000571	0.000548	0.00125	0.004
ballOnPlate_3	152	77	257	47	7	17	9	7.9e-05	0.000106	0.000541	0.000553	0.00179	0.00487
dcMotor_1	144	94	424	32	9	-	-	8.2e-05	-	-	0.000738	-	-
aircraft_10	164	104	444	20	9	10	10	8.14e-05	0.000127	0.00055	0.000733	0.00127	0.0055
aircraft_11	164	104	444	64	9	12	-	8.04e-05	8.75e-05	-	0.000724	0.00105	-
aircraft_12	164	104	444	64	9	12	-	7.58e-05	8.7e-05	-	0.000682	0.00104	-
toyExample_4	202	152	502	151	9	-	-	9.52e-05	-	-	0.000856	-	-
ballOnPlate_4	252	127	427	77	9	19	11	0.00011	0.000155	0.0013	0.000989	0.00294	0.0143
helicopter_2	218	166	538	175	6	21	-	0.000164	0.000165	-	0.000983	0.00347	-
shell_1	249	159	559	60	7	15	8	0.00018	0.000226	0.00147	0.00126	0.00339	0.0118
shell_3	249	159	559	60	11	20	15	0.000151	0.000216	0.00139	0.00166	0.00433	0.0208
nonlinearCstr_1	244	204	804	120	10	-	22	0.000138	-	0.00209	0.00138	-	0.0459
nonlinearCstr_2	244	204	804	120	10	-	22	0.000134	-	0.00208	0.00134	-	0.0459
dcMotor_2	284	184	844	62	10	16	21	0.000166	0.000142	0.00139	0.00166	0.00227	0.0292

Table A.2: Solve times and iteration counts for the Optimal Control problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
dcMotor_5	284	184	844	62	10	16	-	0.000148	0.000146	-	0.00148	0.00234	-
dcMotor_6	284	184	844	62	9	16	-	0.000155	0.00015	-	0.0014	0.0024	-
robotArm_2	324	204	684	124	8	-	-	0.000246	-	-	0.00197	-	-
spacecraft_1	367	187	807	110	10	18	14	0.000165	0.000198	0.00201	0.00165	0.00356	0.0282
spacecraft_2	367	187	807	110	10	18	14	0.000165	0.000197	0.00201	0.00165	0.00354	0.0281
binaryDistillationColumn_1	311	266	2666	90	9	18	13	0.000381	0.000386	0.00221	0.00343	0.00694	0.0287
binaryDistillationColumn_2	311	266	2666	90	9	18	13	0.000379	0.000431	0.0022	0.00341	0.00776	0.0286
quadcopter_1	382	292	1162	110	8	16	10	0.00026	0.000389	0.00245	0.00208	0.00623	0.0245
quadcopter_6	382	292	1162	172	8	16	12	0.000293	0.000366	0.00259	0.00235	0.00585	0.031
toyExample_5	402	302	1002	301	10	27	-	0.000185	0.000259	-	0.00185	0.007	-
shell_2	489	309	1109	120	7	15	9	0.000344	0.000435	0.0035	0.00241	0.00653	0.0315
tripleInvertedPendulum_1	456	366	2166	201	8	-	16	0.000579	-	0.00391	0.00463	-	0.0625
springMass_2	566	286	1646	181	7	18	18	0.000312	0.000353	0.00377	0.00219	0.00635	0.0679
springMass_3	566	286	1646	166	8	21	-	0.000293	0.00032	-	0.00234	0.00672	-
tripleInvertedPendulum_2	636	366	2346	201	8	21	13	0.000658	0.000381	0.00452	0.00527	0.00801	0.0588
tripleInvertedPendulum_3	636	366	2346	201	9	24	-	0.000648	0.000363	-	0.00583	0.00872	-
quadcopter_5	572	572	2212	220	27	-	-	0.000473	-	-	0.0128	-	-
aircraft_13	804	504	2204	304	11	23	-	0.000367	0.00048	-	0.00403	0.011	-
quadcopter_2	752	572	2312	220	8	18	11	0.000484	0.000879	0.00437	0.00387	0.0158	0.0481
quadcopter_4	752	572	2312	220	12	19	16	0.000465	0.000739	0.00396	0.00558	0.0141	0.0634
springMass_4	1126	566	3286	341	8	17	15	0.00062	0.000631	0.0058	0.00496	0.0107	0.087
dcMotor_3	1404	904	4204	302	11	-	-	0.000801	-	-	0.00881	-	-
dcMotor_4	1404	904	4204	302	11	25	-	0.000734	0.000634	-	0.00807	0.0159	-
quadcopter_3	1862	1412	5762	550	8	20	10	0.00121	0.00238	0.00935	0.00968	0.0476	0.0935
nonlinearChain_13	2457	2397	30857	660	9	22	-	0.00537	0.00579	-	0.0483	0.127	-
nonlinearChain_3	2457	2397	30857	660	9	22	-	0.00536	0.0058	-	0.0483	0.128	-
nonlinearChain_14	2637	2397	31037	660	10	23	-	0.00532	0.00663	-	0.0532	0.153	-
nonlinearChain_4	2637	2397	31037	660	10	23	-	0.00535	0.00666	-	0.0535	0.153	-
springMass_1	5606	2806	16406	1621	12	-	-	0.00444	-	-	0.0532	-	-
nonlinearChain_1	9657	9417	123257	2640	8	28	-	0.0225	0.0232	-	0.18	0.649	-
nonlinearChain_11	9657	9417	123257	2640	8	28	-	0.0223	0.0229	-	0.178	0.642	-
nonlinearChain_12	10377	9417	123977	2640	9	-	-	0.0219	-	-	0.197	-	-
nonlinearChain_2	10377	9417	123977	2640	9	-	-	0.022	-	-	0.198	-	-

Table A.3: Solve times and iteration counts for the SuiteSparse least-squares problem set

Problem	vars.	cons.	nnz(A)	nnz(P)	iterations			time per iteration(s)			total time (s)		
					ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
NYPA_Maragal_1_lasso	60	60	322	32	9	15	10	5.02e-05	6.17e-05	0.000325	0.000452	0.000925	0.00325
NYPA_Maragal_1_huber	96	110	394	32	6	10	5	5.97e-05	8.64e-05	0.000603	0.000358	0.000864	0.00301
HB_ash85_lasso	255	255	948	85	8	18	15	0.00018	0.000325	0.00169	0.00144	0.00585	0.0253
HB_ash85_huber	255	340	948	85	6	6	12	0.000297	0.000344	0.00199	0.00178	0.00206	0.0238
HB_ash219_lasso	389	389	997	219	8	18	16	0.000247	0.000354	0.00255	0.00198	0.00637	0.0408
HB_ash331_lasso	539	539	1409	331	7	18	21	0.000503	0.0005	0.00402	0.00352	0.009	0.0845
HB_abb313_lasso	665	665	2574	313	9	22	-	0.000607	0.000686	-	0.00546	0.0151	-
HB_ash219_huber	657	742	1533	219	9	16	16	0.0003	0.000711	0.00417	0.0027	0.0114	0.0667
HB_ash292_lasso	876	876	3668	292	8	24	19	0.000848	0.000992	0.00617	0.00678	0.0238	0.117
HB_ash608_lasso	984	984	2576	608	8	-	-	0.00047	-	-	0.00376	-	-
HB_ash292_huber	876	1168	3668	292	6	11	15	0.000914	0.00138	0.00502	0.00549	0.0152	0.0753
HB_abb313_huber	939	1115	3122	313	9	19	17	0.000498	0.000995	0.0062	0.00448	0.0189	0.105
HB_ash331_huber	993	1097	2317	331	8	16	15	0.000434	0.00104	0.00546	0.00347	0.0166	0.082
NYPA_Maragal_2_lasso	1255	1255	6312	555	7	14	11	0.00122	0.00185	0.011	0.00854	0.0259	0.121
HB_ash958_lasso	1542	1542	4042	958	7	-	21	0.000786	-	0.00901	0.00551	-	0.189
HB_illc1033_lasso	1673	1673	7032	1033	19	-	-	0.000932	-	-	0.0177	-	-
HB_well1033_lasso	1673	1673	7045	1033	13	-	-	0.000894	-	-	0.0116	-	-
NYPA_Maragal_2_huber	1665	2015	7132	555	8	16	12	0.00127	0.0029	0.00951	0.0101	0.0463	0.114
HB_ash608_huber	1824	2012	4256	608	9	18	16	0.000756	0.00127	0.00938	0.00681	0.0228	0.15
HB_ash958_huber	2874	3166	6706	958	9	17	18	0.00131	0.00208	0.0154	0.0118	0.0354	0.277
HB_illc1033_huber	3099	3419	9884	1033	7	11	7	0.002	0.00306	0.0177	0.014	0.0337	0.124
HB_well1033_huber	3099	3419	9897	1033	7	12	7	0.00168	0.00387	0.0175	0.0118	0.0465	0.123
HB_illc1850_lasso	3274	3274	13334	1850	13	-	-	0.00212	-	-	0.0275	-	-
HB_well1850_lasso	3274	3274	13453	1850	12	-	-	0.00201	-	-	0.0241	-	-
NYPA_Maragal_3_lasso	3410	3410	23521	1690	7	15	9	0.00921	0.0102	0.0367	0.0645	0.153	0.33
NYPA_Maragal_4_lasso	4032	4032	32819	1964	6	17	9	0.0228	0.0253	0.0427	0.137	0.429	0.384
NYPA_Maragal_3_huber	5070	5930	26841	1690	8	18	11	0.00971	0.0112	0.0477	0.0777	0.202	0.524
HB_illc1850_huber	5550	6262	17886	1850	7	11	13	0.00343	0.00722	0.0297	0.024	0.0795	0.386
HB_well1850_huber	5550	6262	18005	1850	7	11	13	0.00342	0.00718	0.034	0.024	0.079	0.441
NYPA_Maragal_4_huber	5892	6926	36539	1964	7	18	13	0.0252	0.0245	0.0437	0.177	0.441	0.568
NYPA_Maragal_5_lasso	11294	11294	111025	4654	8	21	9	0.156	0.168	0.168	1.25	3.53	1.51
NYPA_Maragal_5_huber	13962	17282	116361	4654	8	20	-	0.17	0.168	-	1.36	3.36	-
NYPA_Maragal_6_lasso	41559	41559	599557	21255	7	-	8	4.88	-	1.4	34.1	-	11.2
NYPA_Maragal_6_huber	63765	73917	643969	21255	7	-	-	5.41	-	-	37.9	-	-
Pereyra_landmark_lasso	77360	77360	1229616	71952	8	-	-	0.185	-	-	1.48	-	-
NYPA_Maragal_7_lasso	99973	99973	1353638	46845	7	-	-	11	-	-	76.7	-	-
NYPA_Maragal_8_huber	99636	174713	1474475	33212	8	-	16	6.5	-	1.41	52	-	22.5
NYPA_Maragal_7_huber	140535	167099	1434762	46845	10	-	-	10.9	-	-	109	-	-
NYPA_Maragal_8_lasso	183366	183366	1641935	33212	7	-	-	6.59	-	-	46.1	-	-
Pereyra_landmark_huber	215856	218560	1506608	71952	82	-	-	0.176	-	-	14.4	-	-
ANSYS_Delor64K_huber	194157	1979502	975735	64719	7	-	14	0.636	-	1.61	4.45	-	22.5
ANSYS_Delor338K_huber	1029708	1916766	5927779	343236	8	-	4	1.46	-	18.1	11.7	-	72.5
ANSYS_Delor295K_huber	887202	2711130	3879993	295734	8	-	4	1.35	-	20	10.8	-	80.2
ANSYS_Delor338K_lasso	2117352	2117352	8103067	343236	7	-	-	2.14	-	-	15	-	-
ANSYS_Delor64K_lasso	3635409	3635409	7858239	64719	8	-	-	2.61	-	-	20.9	-	-
ANSYS_Delor295K_lasso	3943590	3943590	9992769	295734	8	-	-	3.17	-	-	25.4	-	-

Table A.4: Solve times and iteration counts for the NETLIB Feasible LP problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
AFIRO	78	51	153	0	8	9	11	3.91e-05	3.05e-05	0.000263	0.000312	0.000275	0.00289
KB2	120	68	390	0	16	16	28	6.83e-05	6.32e-05	0.000208	0.00109	0.00101	0.00583
SC50A	128	78	238	0	9	11	11	6e-05	4.84e-05	0.000276	0.00054	0.000532	0.00304
SC50B	128	78	226	0	8	10	9	6.27e-05	4.58e-05	0.000305	0.000502	0.000458	0.00274
BLEND	188	114	636	0	11	12	14	0.000115	0.000102	0.00102	0.00126	0.00122	0.0143
ADLITTLE	194	138	562	0	12	12	16	0.000115	9.69e-05	0.000976	0.00137	0.00116	0.0156
SHARE2B	258	162	939	0	11	13	13	0.000166	0.000143	0.00128	0.00182	0.00186	0.0166
SC105	268	163	503	0	10	12	12	0.00012	0.000103	0.000518	0.0012	0.00123	0.00622
STOCFOR1	282	165	666	0	16	15	26	0.000138	0.000126	0.00124	0.00222	0.00189	0.0323
SCAGR7	314	185	650	0	15	15	15	0.00012	0.00011	0.00125	0.0018	0.00165	0.0187
RECIPE	364	204	960	0	10	11	7	0.000179	0.000149	0.00184	0.00179	0.00164	0.0129
SHARE1B	370	253	1432	0	22	27	45	0.000286	0.000228	0.00177	0.0063	0.00616	0.0798
NUG05	435	225	1275	0	7	7	7	0.000949	0.000839	0.00199	0.00664	0.00587	0.014
BEACONFD	468	295	3703	0	9	10	14	0.000702	0.000561	0.0026	0.00632	0.00561	0.0365
ISRAEL	490	316	2759	0	19	23	33	0.000457	0.000398	0.0024	0.00868	0.00915	0.0793
BRANDY	523	303	2505	0	15	18	24	0.000513	0.000448	0.00245	0.0077	0.00806	0.0587
SC205	522	317	982	0	13	13	13	0.000231	0.000247	0.00145	0.003	0.00321	0.0189
LOTFI	519	366	1502	0	19	20	17	0.000302	0.000247	0.00212	0.00574	0.00495	0.036
BORE3D	578	334	1793	0	19	20	40	0.000401	0.000339	0.00244	0.00762	0.00679	0.0977
VTP_BASE	608	346	1461	0	31	33	19	0.000283	0.000269	0.00267	0.00877	0.00888	0.0507
GROW7	721	301	3193	0	13	12	12	0.000614	0.000547	0.00483	0.00798	0.00657	0.058
E226	695	472	3240	0	22	24	40	0.000588	0.000525	0.00284	0.0129	0.0126	0.114
BANDM	777	472	2966	0	18	21	24	0.000618	0.000656	0.00339	0.0111	0.0138	0.0813
SCORPION	854	466	2000	0	12	12	12	0.000478	0.000416	0.00238	0.00574	0.00499	0.0286
NUG06	858	486	2718	0	7	7	7	0.00302	0.00287	0.00456	0.0211	0.0201	0.032
CAPRI	870	482	2495	0	21	22	16	0.000579	0.000522	0.00397	0.0122	0.0115	0.0635
SCFXM1	930	600	3332	0	19	22	18	0.000767	0.000702	0.00401	0.0146	0.0154	0.0722
STAIR	970	614	4617	0	22	23	15	0.000999	0.000898	0.00704	0.022	0.0206	0.106
SCSD1	837	760	3148	0	10	11	13	0.000506	0.000453	0.0028	0.00506	0.00499	0.0364
TUFF	985	628	5213	0	21	-	22	0.00113	-	0.00488	0.0237	-	0.107
SCTAP1	960	660	2532	0	26	25	16	0.00043	0.000402	0.00365	0.0112	0.0101	0.0584
AGG	1103	615	3477	0	45	43	43	0.000834	0.000554	0.00246	0.0375	0.0238	0.106
SCAGR25	1142	671	2396	0	17	18	19	0.000397	0.000376	0.00411	0.00675	0.00676	0.0781
DEGEN2	1201	757	4958	0	11	12	11	0.00177	0.0016	0.00936	0.0195	0.0192	0.103
AGG2	1274	758	5498	0	26	28	26	0.0019	0.00188	0.00569	0.0494	0.0526	0.148
AGG3	1274	758	5514	0	25	29	27	0.00187	0.00182	0.00609	0.0468	0.0529	0.165
ETAMACRO	1351	816	3488	0	25	33	33	0.00175	0.00142	0.00608	0.0438	0.047	0.201
GROW15	1545	645	6865	0	13	13	14	0.0013	0.00111	0.00946	0.0168	0.0145	0.132
NUG07	1533	931	5145	0	11	11	9	0.00895	0.00919	0.0178	0.0984	0.101	0.16

Table A.4: Solve times and iteration counts for the NETLIB Feasible LP problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
FFFFF800	1552	1028	7429	0	36	49	25	0.0023	0.00202	0.0076	0.0829	0.0992	0.19
FINNIS	1597	1064	3860	0	36	44	26	0.000909	0.000707	0.00631	0.0327	0.0311	0.164
PILOT4	1692	1123	6546	0	39	72	37	0.0016	0.00114	0.00793	0.0625	0.0821	0.294
SCSD6	1497	1350	5666	0	13	13	13	0.000832	0.000806	0.00487	0.0108	0.0105	0.0633
STANDATA	1737	1274	4608	0	12	15	20	0.000954	0.000763	0.00566	0.0114	0.0115	0.113
SCRS8	1765	1275	4563	0	28	29	37	0.00104	0.000916	0.00643	0.0292	0.0266	0.238
SCFXM2	1860	1200	6669	0	22	23	20	0.00149	0.00144	0.00768	0.0328	0.0331	0.154
STANDMPS	1845	1274	5256	0	17	18	29	0.0011	0.000879	0.00609	0.0187	0.0158	0.176
FIT1D	2099	1049	15502	0	28	30	26	0.00173	0.00163	0.00294	0.0484	0.0489	0.0763
GFRD_PNC	2034	1160	3863	0	14	17	24	0.000869	0.000767	0.00664	0.0122	0.013	0.159
GROW22	2266	946	10078	0	14	14	14	0.00184	0.00155	0.0136	0.0257	0.0217	0.19
STANDGUB	1848	1383	4825	0	12	15	21	0.00101	0.000816	0.00573	0.0121	0.0122	0.12
SHIP04S	1908	1506	5906	0	17	16	28	0.00119	0.00111	0.00571	0.0202	0.0177	0.16
BNL1	2229	1586	7118	0	57	58	36	0.00183	0.00182	0.00851	0.104	0.105	0.306
PEROLD	2309	1506	7832	0	51	-	44	0.003	-	0.00978	0.153	-	0.43
MODSZK1	2305	1620	4786	0	27	31	27	0.00125	0.000878	0.00783	0.0337	0.0272	0.211
NUG08	2544	1632	8928	0	9	9	7	0.0294	0.0307	0.0389	0.264	0.277	0.272
QAP8	2544	1632	8928	0	9	9	7	0.031	0.032	0.0333	0.279	0.288	0.233
SHELL	2430	1777	5452	0	17	18	16	0.00123	0.00108	0.00895	0.0209	0.0195	0.143
FIT1P	2703	1677	11944	0	18	23	21	0.00166	0.00137	0.00862	0.0298	0.0315	0.181
25FV47	2697	1876	12581	0	27	28	29	0.00437	0.00442	0.0122	0.118	0.124	0.355
SCFXM3	2790	1800	10006	0	22	24	20	0.00221	0.00214	0.0117	0.0486	0.0514	0.234
SHIP04L	2568	2166	8546	0	17	19	23	0.00156	0.00141	0.00764	0.0265	0.0268	0.176
MAROS	2812	1966	12103	0	28	34	36	0.00405	0.00325	0.0111	0.113	0.11	0.398
GANGES	3412	1706	9040	0	31	37	16	0.00204	0.00181	0.0148	0.0634	0.0668	0.236
WOOD1P	2839	2595	72811	0	16	18	14	0.00935	0.0085	0.0257	0.15	0.153	0.36
SHIP08S	3245	2467	9661	0	16	17	20	0.00198	0.00185	0.0104	0.0317	0.0315	0.207
PILOT_JA	3458	2267	17495	0	59	-	35	0.00599	-	0.0169	0.354	-	0.593
SCSD8	3147	2750	11334	0	11	12	12	0.00183	0.00155	0.0106	0.0201	0.0186	0.128
SCTAP2	3590	2500	9834	0	16	16	28	0.00185	0.00174	0.0115	0.0296	0.0278	0.322
PILOTNOV	3761	2446	16117	0	21	22	22	0.00658	0.0063	0.0188	0.138	0.139	0.413
DEGEN3	4107	2604	28036	0	14	15	11	0.014	0.0139	0.0468	0.196	0.209	0.515
PILOT_WE	3864	2928	12407	0	61	73	59	0.00305	0.00259	0.0133	0.186	0.189	0.785
SHIP12S	4020	2869	11153	0	21	23	25	0.00224	0.00209	0.0117	0.047	0.0482	0.294
CZPROB	4491	3562	14270	0	43	43	25	0.00236	0.00213	0.0135	0.101	0.0916	0.338
SCTAP3	4820	3340	13074	0	16	16	35	0.00251	0.00239	0.0155	0.0401	0.0382	0.542
STOCFOR2	5202	3045	12402	0	28	27	32	0.00275	0.00283	0.0136	0.0771	0.0765	0.436
SIERRA	5978	2735	12752	0	18	21	35	0.00264	0.00283	0.0212	0.0474	0.0595	0.742
CYCLE	5344	3371	24675	0	38	-	47	0.00677	-	0.0241	0.257	-	1.13

Table A.4: Solve times and iteration counts for the NETLIB Feasible LP problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
SHIP08L	5141	4363	17245	0	18	19	21	0.00345	0.00319	0.0157	0.062	0.0607	0.329
BNL2	6810	4486	19482	0	34	38	33	0.00949	0.0094	0.0284	0.323	0.357	0.939
PILOT	7341	4860	50275	0	65	-	59	0.0274	-	0.0368	1.78	-	2.17
SHIP12L	6684	5533	21809	0	29	28	22	0.00398	0.00401	0.0201	0.115	0.112	0.441
D6CUBE	6599	6184	43888	0	18	-	12	0.0125	-	0.0469	0.225	-	0.562
KEN_07	9630	3602	15608	0	17	16	20	0.00336	0.00346	0.0327	0.0572	0.0553	0.654
D2Q06C	8002	5831	38912	0	32	48	48	0.0203	0.0192	0.0338	0.65	0.92	1.62
GREENBEB	8277	5598	36955	0	66	69	47	0.0166	0.0178	0.0331	1.1	1.23	1.56
GREENBEA	8280	5598	36958	0	42	44	55	0.0167	0.0174	0.0329	0.703	0.767	1.81
CRE_C	9479	6411	22388	0	27	32	23	0.00567	0.00595	0.0356	0.153	0.19	0.819
PILOT87	10288	6680	83207	0	100	-	94	0.11	-	0.0648	11	-	6.09
WOODW	9516	8418	45905	0	26	32	19	0.00939	0.00846	0.0351	0.244	0.271	0.667
CRE_A	10764	7248	25416	0	22	26	40	0.00663	0.00682	0.0363	0.146	0.177	1.45
TRUSS	9806	8806	36642	0	20	20	20	0.00879	0.00893	0.036	0.176	0.179	0.72
PDS_02	12803	7716	26421	0	34	32	24	0.00671	0.00721	0.0474	0.228	0.231	1.14
NUG12	12048	8856	47160	0	20	20	16	1.05	1.1	0.192	21	22.1	3.07
QAP12	12048	8856	47160	0	20	20	16	1.03	1.14	0.217	20.5	22.9	3.47
MAROS_R7	12544	9408	154256	0	14	15	13	0.274	0.281	0.116	3.84	4.21	1.51
80BAU3B	17309	12061	38311	0	37	41	25	0.00935	0.011	0.0579	0.346	0.45	1.45
DFL001	18314	12230	47875	0	-	47	35	-	0.871	0.127	-	40.9	4.46
FIT2D	21049	10524	150066	0	24	27	27	0.0138	0.0154	0.0205	0.33	0.417	0.553
FIT2P	24025	13525	71309	0	21	23	21	0.0129	0.0107	0.0624	0.27	0.247	1.31
NUG15	28605	22275	117225	0	24	24	15	10.4	10.4	4.49	249	251	67.4
QAP15	28605	22275	117225	0	24	24	17	10.5	11.3	3.82	251	270	65
OSA_07	26185	25067	169879	0	23	24	27	0.0347	0.0295	0.0988	0.797	0.708	2.67
STOCFOR3	40216	23541	96262	0	42	47	61	0.024	0.027	0.0805	1.01	1.27	4.91
PDS_06	48472	29351	101811	0	46	45	31	0.136	0.144	0.204	6.25	6.46	6.33
KEN_11	57392	21349	91756	0	25	25	21	0.0279	0.03	0.101	0.699	0.75	2.12
PDS_10	82638	49932	173685	0	61	64	38	0.507	0.509	0.39	30.9	32.6	14.8
KEN_13	113950	42659	182564	0	27	27	29	0.0674	0.0691	0.196	1.82	1.86	5.68

Table A.5: Solve times and iteration counts for the NETLIB Infeasible LP problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
LPI_ITEST2	22	13	39	0	6	6	4	1.63e-05	1.2e-05	0.00019	9.75e-05	7.23e-05	0.00076
LPI_GALENET	30	14	44	0	6	5	4	2.2e-05	1.5e-05	0.000187	0.000132	7.5e-05	0.000749
LPI_ITEST6	28	17	46	0	5	8	4	2.42e-05	1.67e-05	0.00024	0.000121	0.000134	0.000961
LPI_BGPRTR	60	40	110	0	8	9	7	3.33e-05	2.84e-05	0.000262	0.000266	0.000256	0.00183
LPI_WOODINFE	138	89	243	0	7	8	6	7.6e-05	5.56e-05	0.000569	0.000532	0.000445	0.00342
LPI_KLEIN1	162	108	858	0	18	18	21	0.000178	0.000164	0.000433	0.00321	0.00296	0.00909
LPI_FOREST6	202	131	382	0	8	9	10	9.06e-05	8.71e-05	0.000913	0.000725	0.000784	0.00913
LPI_EX73A	404	211	668	0	7	6	15	0.000207	0.000179	0.00115	0.00145	0.00107	0.0172
LPI_EX72A	412	215	682	0	7	6	17	0.000223	0.000205	0.00112	0.00156	0.00123	0.0191
LPI_BOX1	492	261	912	0	8	7	4	0.000217	0.000238	0.00294	0.00174	0.00166	0.0118
LPI_QUAL	1032	464	2355	0	61	55	51	0.000537	0.000495	0.00259	0.0327	0.0272	0.132
LPI_REFINERY	1032	464	2335	0	19	20	23	0.000533	0.000469	0.00243	0.0101	0.00938	0.0558
LPI_VOL1	1032	464	2355	0	56	57	43	0.000539	0.000499	0.0024	0.0302	0.0285	0.103
LPI_KLEIN2	1008	531	5593	0	14	16	41	0.000917	0.000958	0.00219	0.0128	0.0153	0.09
LPI_BGDBG1	1022	629	2336	0	9	8	9	0.00062	0.000602	0.00432	0.00558	0.00481	0.0388
LPI_PANG	1117	741	3689	0	24	24	27	0.000852	0.000788	0.00465	0.0205	0.0189	0.126
LPI_CHEMCOM	1176	744	2478	0	7	8	7	0.000791	0.000697	0.00525	0.00554	0.00557	0.0367
LPI_MONDOU2	1393	604	2289	0	1	12	22	0.00196	0.000402	0.00322	0.00196	0.00482	0.0709
LPI_BGETAM	1351	816	3488	0	7	8	8	0.00206	0.00189	0.00958	0.0144	0.0152	0.0766
LPI_REACTOR	1674	808	3947	0	29	29	22	0.000933	0.000956	0.00695	0.0271	0.0277	0.153
LPI_PILOT4I	1692	1123	6546	0	27	30	23	0.00186	0.00138	0.0104	0.0503	0.0413	0.238
LPI_KLEIN3	2076	1082	14183	0	18	17	28	0.00224	0.00208	0.00477	0.0404	0.0354	0.134
LPI_GRAN	5707	2525	23160	0	10	13	46	0.00835	0.00683	0.0128	0.0835	0.0887	0.591
LPI_CERIA3D	7152	4400	24754	0	15	11	10	0.00586	0.00561	0.0557	0.0879	0.0618	0.557
LPI_CPLEX1	8447	5224	16389	0	6	11	22	0.00679	0.00279	0.0152	0.0407	0.0307	0.335
LPI_GREENBEA	8290	5596	36971	0	29	32	17	0.0177	0.0183	0.0371	0.513	0.587	0.63
LPI_BGINDY	13551	10880	77146	0	7	9	9	0.0377	0.0294	0.072	0.264	0.265	0.648
LPI_GOSH	17005	13455	113166	0	41	36	24	0.0498	0.0459	0.0612	2.04	1.65	1.47

Table A.6: Solve times and iteration counts for the LP Optimal Power Flow problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
case3__lmbd	25	9	43	2	8	18	-	1.55e-05	1.27e-05	-	0.000124	0.000229	-
case3__lmbd___api	25	9	43	2	8	16	-	1.53e-05	1.58e-05	-	0.000122	0.000254	-
case3__lmbd___sad	25	9	43	2	8	18	-	1.51e-05	1.59e-05	-	0.000121	0.000286	-
case5__pjm	46	16	82	0	7	8	8	2.61e-05	2.52e-05	0.000194	0.000183	0.000202	0.00155
case5__pjm___api	46	16	82	0	9	9	10	2.49e-05	3.01e-05	0.000186	0.000224	0.000271	0.00186
case14__ieee	125	39	236	0	7	7	7	6.64e-05	5.97e-05	0.000326	0.000465	0.000418	0.00228
case14__ieee___api	125	39	236	0	11	11	11	5.6e-05	6.03e-05	0.000248	0.000616	0.000663	0.00273
case30__as	248	77	470	6	8	16	15	9.86e-05	0.000131	0.000305	0.000789	0.0021	0.00457
case30__as___api	248	77	470	6	9	20	24	0.000103	0.000118	0.000305	0.000924	0.00236	0.00733
case30__ieee	248	77	470	0	8	8	8	0.000121	0.000114	0.00037	0.000966	0.000911	0.00296
case30__ieee___api	248	77	470	0	7	8	9	0.000127	0.000112	0.000346	0.000887	0.000894	0.00311
case24__ieee__rts	273	95	502	22	9	22	-	0.000108	0.00012	-	0.000969	0.00263	-
case24__ieee__rts___api	273	95	502	22	11	23	-	0.000101	0.000108	-	0.00111	0.00247	-
case24__ieee__rts___sad	273	95	502	22	12	27	-	0.000102	0.00012	-	0.00122	0.00325	-
case39__epri	290	95	537	0	9	9	13	0.000126	0.00012	0.000365	0.00113	0.00108	0.00474
case39__epri___api	290	95	537	0	9	10	10	0.000131	0.000118	0.000401	0.00118	0.00118	0.00401
case39__epri___sad	290	95	537	0	18	20	13	0.000119	0.000122	0.000356	0.00215	0.00243	0.00463
case57__ieee	468	144	894	0	8	9	10	0.000217	0.000196	0.000506	0.00174	0.00177	0.00506
case57__ieee___api	468	144	894	0	10	12	11	0.000213	0.000194	0.000507	0.00213	0.00233	0.00558
case60__c	515	171	974	0	8	8	9	0.00023	0.000237	0.000782	0.00184	0.00189	0.00704
case60__c___api	515	171	974	0	14	19	13	0.000201	0.000198	0.000606	0.00282	0.00377	0.00788
case73__ieee__rts	848	292	1570	66	9	25	-	0.000337	0.000391	-	0.00303	0.00977	-
case73__ieee__rts___api	848	292	1570	66	12	25	-	0.000307	0.000349	-	0.00368	0.00871	-
case73__ieee__rts___sad	848	292	1570	66	13	47	-	0.000301	0.000366	-	0.00392	0.0172	-
case89__pegase	1156	311	2331	0	9	8	29	0.000526	0.000513	0.00201	0.00474	0.0041	0.0582
case89__pegase___api	1156	311	2331	0	14	14	35	0.000601	0.000529	0.002	0.00841	0.00741	0.0701
case118__ieee	1143	358	2181	0	12	12	12	0.000492	0.000485	0.00226	0.00591	0.00583	0.0271
case118__ieee___api	1143	358	2181	0	14	14	16	0.000554	0.00059	0.00214	0.00776	0.00826	0.0342
case179__goc	1471	471	2817	0	13	12	18	0.000559	0.000546	0.0026	0.00727	0.00656	0.0468
case179__goc___api	1471	471	2817	0	13	13	23	0.000632	0.000714	0.00255	0.00822	0.00928	0.0586
case200__activ	1502	483	2810	31	10	21	16	0.000567	0.000735	0.00371	0.00567	0.0154	0.0593
case200__activ___api	1502	483	2810	31	11	19	26	0.000571	0.000807	0.00308	0.00628	0.0153	0.0801
case162__ieee__dtc	1599	458	3145	0	14	14	18	0.000739	0.000782	0.00309	0.0104	0.0109	0.0557
case162__ieee__dtc___api	1599	458	3145	0	16	14	16	0.000769	0.000835	0.00312	0.0123	0.0117	0.0499
case162__ieee__dtc___sad	1599	458	3145	0	18	19	16	0.000746	0.000851	0.00316	0.0134	0.0162	0.0506
case197__snem	1572	518	3000	0	9	10	9	0.000645	0.000604	0.00332	0.0058	0.00604	0.0299
case197__snem___api	1572	518	3000	0	12	13	16	0.000684	0.0007	0.00268	0.00821	0.0091	0.0428
case300__ieee	2490	780	4721	0	12	13	18	0.00114	0.0012	0.00416	0.0136	0.0156	0.075
case300__ieee___api	2490	780	4721	0	14	13	16	0.00124	0.00127	0.0044	0.0173	0.0165	0.0704

Table A.6: Solve times and iteration counts for the LP Optimal Power Flow problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
case300__ieee___sad	2490	780	4721	0	14	16	19	0.0013	0.00121	0.00481	0.0182	0.0193	0.0914
case240__pserc	2567	831	4958	0	13	14	23	0.00142	0.00133	0.00496	0.0184	0.0186	0.114
case240__pserc___api	2567	831	4958	0	12	12	14	0.00129	0.00146	0.00527	0.0155	0.0175	0.0738
case588__sdet	4191	1369	7796	0	14	13	14	0.00216	0.00264	0.00722	0.0302	0.0343	0.101
case588__sdet___api	4191	1369	7796	0	13	12	13	0.00232	0.00244	0.00753	0.0302	0.0292	0.0979
case500__goc	4327	1399	8210	60	15	22	-	0.00174	0.00185	-	0.0261	0.0407	-
case500__goc___api	4327	1399	8210	60	18	33	-	0.00167	0.00184	-	0.0301	0.0608	-
case793__goc	5535	1803	10299	48	16	28	-	0.00224	0.00277	-	0.0358	0.0776	-
case793__goc___api	5535	1803	10299	48	15	35	-	0.0022	0.0028	-	0.0331	0.0981	-
case1354__pegase	11268	3605	21558	0	13	13	28	0.00533	0.00671	0.0197	0.0693	0.0872	0.551
case1354__pegase___api	11268	3605	21558	0	14	14	27	0.00603	0.00734	0.0198	0.0844	0.103	0.534
case1888__rte	14678	4709	27820	0	12	13	36	0.00737	0.00827	0.023	0.0884	0.108	0.83
case1888__rte___api	14678	4709	27820	0	16	16	32	0.00755	0.00804	0.0242	0.121	0.129	0.773
case1888__rte___sad	14678	4709	27820	0	13	13	25	0.00708	0.00901	0.0242	0.0921	0.117	0.606
case1803__snem	15041	4828	29036	0	17	18	32	0.00734	0.00927	0.0241	0.125	0.167	0.771
case1803__snem___api	15041	4828	29036	0	20	30	39	0.00818	0.00713	0.024	0.164	0.214	0.935
case1951__rte	15222	4913	28771	0	14	14	33	0.00724	0.0097	0.0246	0.101	0.136	0.811
case1951__rte___api	15222	4913	28771	0	17	17	36	0.00882	0.00788	0.0257	0.15	0.134	0.926
case2312__goc	17464	5551	33090	42	22	27	-	0.0076	0.00911	-	0.167	0.246	-
case2312__goc___api	17464	5551	33090	42	18	62	-	0.00764	0.00818	-	0.138	0.507	-
case2383wp__k	17498	5606	32798	0	21	23	25	0.0104	0.0102	0.0294	0.219	0.235	0.734
case2383wp__k___api	17498	5606	32798	0	12	10	13	0.0088	0.0119	0.0401	0.106	0.119	0.522
case2000__goc	18988	5871	37370	122	14	36	-	0.00782	0.00847	-	0.109	0.305	-
case2000__goc___api	18988	5871	37370	122	18	37	-	0.00797	0.00844	-	0.143	0.312	-
case2737sop__k	19509	6225	36593	0	16	18	20	0.0105	0.0137	0.036	0.168	0.247	0.719
case2737sop__k___api	19509	6225	36593	0	15	17	25	0.0107	0.0139	0.0332	0.161	0.236	0.83
case2736sp__k	19610	6275	36746	0	13	13	23	0.01	0.0125	0.0351	0.13	0.163	0.807
case2736sp__k___api	19610	6275	36746	0	19	23	23	0.0117	0.0124	0.0345	0.222	0.285	0.793
case2746wp__k	20042	6481	37414	0	15	17	27	0.0103	0.013	0.0351	0.154	0.221	0.947
case2746wp__k___api	20042	6481	37414	0	25	20	26	0.00833	0.0108	0.0358	0.208	0.215	0.93
case2746wop__k	20128	6484	37639	0	14	15	23	0.0106	0.0138	0.0362	0.149	0.207	0.833
case2746wop__k___api	20128	6484	37639	0	18	20	22	0.0122	0.0111	0.0361	0.22	0.222	0.794
case3012wp__k	21631	6969	40424	0	16	20	25	0.0135	0.0131	0.0378	0.215	0.262	0.945
case3012wp__k___api	21631	6969	40424	0	24	37	27	0.013	0.0101	0.0381	0.312	0.375	1.03
case2848__rte	22083	7135	41734	0	13	13	32	0.0109	0.0123	0.0362	0.142	0.16	1.16
case2848__rte___api	22083	7135	41734	0	15	16	26	0.0112	0.0127	0.0384	0.168	0.204	0.999
case3120sp__k	22164	7111	41482	0	17	18	27	0.0132	0.0128	0.0376	0.225	0.231	1.02
case3120sp__k___api	22164	7111	41482	0	17	26	26	0.0138	0.0117	0.0391	0.235	0.304	1.02
case2868__rte	22357	7237	42224	0	14	15	29	0.0112	0.0157	0.0373	0.156	0.235	1.08

Table A.6: Solve times and iteration counts for the LP Optimal Power Flow problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
case2868_rte___api	22357	7237	42224	0	17	16	34	0.012	0.0128	0.0373	0.204	0.204	1.27
case2853_sdet	23525	7593	44445	0	19	20	24	0.0117	0.0127	0.0368	0.223	0.255	0.884
case2853_sdet___api	23525	7593	44445	0	41	-	49	0.00887	-	0.0359	0.364	-	1.76
case3022_goc	23816	7484	45395	110	22	40	-	0.0114	0.013	-	0.251	0.521	-
case3022_goc___api	23816	7484	45395	110	23	73	55	0.00994	0.0111	0.0486	0.229	0.809	2.67
case3022_goc___sad	23816	7484	45395	110	22	43	-	0.0115	0.0126	-	0.253	0.542	-
case2742_goc	25136	7597	49278	48	15	56	-	0.0114	0.0119	-	0.171	0.665	-
case2742_goc___api	25136	7597	49278	48	19	74	-	0.0117	0.0129	-	0.223	0.952	-
case2742_goc___sad	25136	7597	49278	48	15	62	-	0.0117	0.0107	-	0.176	0.666	-
case3375wp_k	24952	8014	46837	0	16	15	27	0.0132	0.0184	0.0432	0.212	0.276	1.17
case3375wp_k___api	24952	8014	46837	0	18	19	38	0.0142	0.0133	0.0426	0.256	0.252	1.62
case3375wp_k___sad	24952	8014	46837	0	17	16	30	0.0139	0.0164	0.0416	0.236	0.263	1.25
case2869_pegase	25572	7961	49477	0	16	15	38	0.0119	0.0173	0.0394	0.19	0.26	1.5
case2869_pegase___api	25572	7961	49477	0	16	16	37	0.0134	0.02	0.0404	0.215	0.32	1.5
case4661_sdet	35603	11382	67156	0	17	20	27	0.0269	0.0245	0.0566	0.457	0.49	1.53
case4661_sdet___api	35603	11382	67156	0	17	31	35	0.0246	0.0199	0.06	0.419	0.618	2.1
case3970_goc	36084	10994	70485	65	15	67	-	0.0163	0.0244	-	0.244	1.64	-
case3970_goc___api	36084	10994	70485	65	18	-	-	0.0169	-	-	0.304	-	-
case4020_goc	37867	11360	74329	23	26	36	40	0.0179	0.0244	0.0669	0.466	0.879	2.67
case4020_goc___api	37867	11360	74329	23	20	-	-	0.0195	-	-	0.389	-	-
case4917_goc	38604	12210	73532	193	21	69	-	0.0215	0.0174	-	0.451	1.2	-
case4917_goc___api	38604	12210	73532	193	22	-	-	0.017	-	-	0.375	-	-
case4917_goc___sad	38604	12210	73532	193	22	68	-	0.018	0.0178	-	0.396	1.21	-
case4601_goc	39625	12208	76838	62	16	75	-	0.0176	0.0231	-	0.282	1.73	-
case4601_goc___api	39625	12208	76838	62	19	-	-	0.0186	-	-	0.354	-	-
case4601_goc___sad	39625	12208	76838	62	21	-	-	0.0204	-	-	0.429	-	-
case4837_goc	42041	12934	81840	50	17	51	-	0.0199	0.0236	-	0.339	1.2	-
case4837_goc___api	42041	12934	81840	50	21	67	-	0.0188	0.0211	-	0.395	1.42	-
case4619_goc	44438	13116	87440	29	18	53	-	0.0209	0.0222	-	0.375	1.18	-
case4619_goc___api	44438	13116	87440	29	20	-	-	0.0243	-	-	0.486	-	-
case5658_epigrids	49549	15204	96379	0	17	16	39	0.0265	0.0376	0.0732	0.451	0.602	2.85
case5658_epigrids___api	49549	15204	96379	0	14	16	40	0.0297	0.0384	0.074	0.416	0.615	2.96
case6468_rte	50397	15867	96458	0	15	16	61	0.0275	0.0347	0.0686	0.412	0.556	4.18
case6468_rte___api	50397	15867	96458	0	18	18	55	0.0274	0.0359	0.0659	0.494	0.646	3.62
case6468_rte___sad	50397	15867	96458	0	15	17	61	0.0272	0.0368	0.0673	0.408	0.626	4.1
case6495_rte	51081	16194	97510	0	18	19	70	0.0271	0.0474	0.0731	0.487	0.9	5.11
case6495_rte___api	51081	16194	97510	0	17	18	58	0.03	0.0311	0.0734	0.509	0.56	4.26
case6495_rte___sad	51081	16194	97510	0	18	19	65	0.0264	0.0393	0.0745	0.475	0.747	4.84
case6470_rte	51140	16236	97583	0	16	16	48	0.0272	0.0479	0.0702	0.435	0.767	3.37

Table A.6: Solve times and iteration counts for the LP Optimal Power Flow problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
case6470_rte___api	51140	16236	97583	0	16	16	40	0.0298	0.0309	0.0736	0.477	0.495	2.94
case6470_rte___sad	51140	16236	97583	0	16	16	45	0.0271	0.0495	0.0704	0.434	0.792	3.17
case6515_rte	51203	16236	97728	0	18	20	58	0.0305	0.0362	0.0673	0.549	0.723	3.9
case6515_rte___api	51203	16236	97728	0	18	18	49	0.0314	0.0263	0.0696	0.565	0.474	3.41
case6515_rte___sad	51203	16236	97728	0	18	19	58	0.0294	0.0359	0.0697	0.53	0.683	4.04
case7336_epigrids	63100	19539	122362	0	16	18	37	0.0365	0.0507	0.0962	0.585	0.913	3.56
case7336_epigrids___api	63100	19539	122362	0	17	19	40	0.0465	0.0457	0.0996	0.791	0.868	3.99
case10000_goc	79096	25209	149368	511	25	44	-	0.0369	0.0513	-	0.923	2.26	-
case10000_goc___api	79096	25209	149368	511	25	-	-	0.0385	-	-	0.963	-	-
case8387_pegase	81791	24813	159503	0	23	27	31	0.0561	0.0751	0.159	1.29	2.03	4.93
case8387_pegase___api	81791	24813	159503	0	23	26	35	0.0579	0.0575	0.14	1.33	1.49	4.9
case8387_pegase___sad	81791	24813	159503	0	23	31	32	0.0575	0.0675	0.145	1.32	2.09	4.64
case9591_goc	86151	25871	168669	55	20	-	-	0.0486	-	-	0.972	-	-
case9591_goc___api	86151	25871	168669	55	24	-	-	0.0519	-	-	1.25	-	-
case9241_pegase	88693	26735	173507	0	108	-	71	0.0347	-	0.152	3.75	-	10.8
case9241_pegase___api	88693	26735	173507	0	19	19	57	0.0557	0.0742	0.15	1.06	1.41	8.54
case10192_epigrids	92051	27914	180020	697	18	63	-	0.052	0.0656	-	0.935	4.14	-
case10192_epigrids___api	92051	27914	180020	697	18	80	-	0.0502	0.057	-	0.904	4.56	-
case10480_goc	99926	29816	196673	276	21	82	-	0.0591	0.0755	-	1.24	6.19	-
case10480_goc___api	99926	29816	196673	276	19	-	-	0.0605	-	-	1.15	-	-
case13659_pegase	120495	38218	230046	0	126	-	43	0.0473	-	0.216	5.96	-	9.3
case13659_pegase___api	120495	38218	230046	0	45	54	37	0.0526	0.0754	0.214	2.37	4.07	7.91
case20758_epigrids	182928	56275	355508	1881	19	53	-	0.123	0.16	-	2.34	8.46	-
case20758_epigrids___api	182928	56275	355508	1881	19	70	-	0.118	0.149	-	2.24	10.4	-
case19402_goc___sad	184959	55077	364846	249	23	-	-	0.127	-	-	2.93	-	-
case30000_goc	213698	68919	399262	372	39	51	-	0.118	0.154	-	4.59	7.87	-
case30000_goc___api	213698	68919	399262	372	31	95	-	0.13	0.152	-	4.02	14.5	-
case30000_goc___sad	213698	68919	399262	372	26	-	-	0.119	-	-	3.1	-	-
case78484_epigrids	685984	211266	1334253	0	30	-	-	0.906	-	-	27.2	-	-
case78484_epigrids___api	685984	211266	1334253	0	27	-	122	0.919	-	2.29	24.8	-	280
case78484_epigrids___sad	685984	211266	1334253	0	31	-	-	0.9	-	-	27.9	-	-

Table A.7: Solve times and iteration counts for the SOCP Optimal Power Flow problem set

Problem	vars.	cons.	nnz(A)	nnz(P)	iterations			time per iteration(s)			total time (s)		
					ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
case3_lmbd	124	29	190	0	13	15	15	5.42e-05	5.95e-05	0.000252	0.000705	0.000893	0.00378
case3_lmbd_api	124	29	190	0	13	14	14	5.68e-05	6.21e-05	0.000255	0.000739	0.00087	0.00357
case3_lmbd_sad	124	29	190	0	13	16	14	5.3e-05	5.95e-05	0.00026	0.000688	0.000952	0.00364
case5_pjm	220	51	352	0	16	19	27	9.52e-05	0.000116	0.000342	0.00152	0.0022	0.00924
case5_pjm_api	220	51	352	0	17	20	29	0.000103	0.000123	0.000343	0.00175	0.00247	0.00996
case5_pjm_sad	220	51	352	0	14	18	22	0.0001	0.000135	0.00036	0.0014	0.00243	0.00793
case14_ieee	676	144	1069	0	14	18	28	0.000283	0.000359	0.00183	0.00396	0.00647	0.0512
case14_ieee_api	676	144	1069	0	16	26	31	0.000285	0.000395	0.00187	0.00456	0.0103	0.0578
case14_ieee_sad	676	144	1069	0	17	22	31	0.000283	0.000363	0.00183	0.00482	0.00798	0.0566
case30_ieee	1374	288	2188	0	25	28	35	0.000629	0.000628	0.00335	0.0157	0.0176	0.117
case30_ieee_api	1374	288	2188	0	22	29	37	0.000653	0.000604	0.0034	0.0144	0.0175	0.126
case30_ieee_sad	1374	288	2188	0	-	-	45	-	-	0.00345	-	-	0.155
case30_as	1404	294	2218	0	21	25	32	0.000567	0.000843	0.00355	0.0119	0.0211	0.114
case30_as_api	1404	294	2218	0	25	31	40	0.000602	0.0007	0.00349	0.0151	0.0217	0.14
case30_as_sad	1404	294	2218	0	19	28	46	0.000579	0.000954	0.00346	0.011	0.0267	0.159
case24_ieee_rts	1430	332	2253	0	18	23	40	0.000661	0.000699	0.00367	0.0119	0.0161	0.147
case24_ieee_rts_api	1430	332	2253	0	19	23	40	0.00067	0.000723	0.00361	0.0127	0.0166	0.145
case24_ieee_rts_sad	1430	332	2253	0	17	22	43	0.000648	0.000802	0.00363	0.011	0.0176	0.156
case39_epri	1576	335	2506	0	24	36	53	0.000712	0.00112	0.00365	0.0171	0.0405	0.193
case39_epri_api	1576	335	2506	0	25	40	53	0.000718	0.000899	0.00359	0.0179	0.0359	0.19
case39_epri_sad	1576	335	2506	0	28	42	61	0.000706	0.000771	0.00365	0.0198	0.0324	0.222
case57_ieee	2632	547	4171	0	23	30	43	0.00126	0.00129	0.00625	0.0289	0.0387	0.269
case57_ieee_api	2632	547	4171	0	22	29	41	0.00124	0.00133	0.00641	0.0273	0.0385	0.263
case57_ieee_sad	2632	547	4171	0	26	37	58	0.00127	0.00115	0.00618	0.033	0.0426	0.358
case60_c	2780	602	4310	0	21	27	38	0.00115	0.00143	0.00668	0.0242	0.0385	0.254
case60_c_api	2780	602	4310	0	26	40	-	0.00123	0.00158	-	0.032	0.063	-
case60_c_sad	2780	602	4310	0	24	31	-	0.00128	0.00153	-	0.0308	0.0474	-
case73_ieee_rts	4474	1033	7067	0	22	26	45	0.00203	0.00264	0.0113	0.0448	0.0685	0.51
case73_ieee_rts_api	4474	1033	7067	0	20	-	43	0.00203	-	0.0114	0.0407	-	0.491
case73_ieee_rts_sad	4474	1033	7067	0	19	25	47	0.00219	0.00229	0.011	0.0416	0.0573	0.516
case118_ieee	6184	1328	10052	0	24	29	63	0.00284	0.00384	0.0149	0.0683	0.111	0.939
case118_ieee_api	6184	1328	10052	0	23	28	62	0.00298	0.00303	0.0148	0.0686	0.0848	0.919
case118_ieee_sad	6184	1328	10052	0	21	27	66	0.00312	0.00325	0.015	0.0655	0.0876	0.987
case89_pegase	6656	1365	11100	0	38	-	-	0.0034	-	-	0.129	-	-
case89_pegase_api	6656	1365	11100	0	35	40	-	0.00336	0.00275	-	0.118	0.11	-
case89_pegase_sad	6656	1365	11100	0	50	-	-	0.00296	-	-	0.148	-	-
case179_goc	8230	1733	12996	0	25	26	-	0.00413	0.00553	-	0.103	0.144	-
case179_goc_api	8230	1733	12996	0	54	-	-	0.00426	-	-	0.23	-	-
case179_goc_sad	8230	1733	12996	0	25	27	77	0.00393	0.00583	0.0193	0.0983	0.158	1.49

Table A.7: Solve times and iteration counts for the SOCP Optimal Power Flow problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
case200_activ	8457	1777	13527	0	38	34	42	0.00381	0.00404	0.0194	0.145	0.137	0.815
case200_activ___api	8457	1777	13527	0	48	-	67	0.00394	-	0.0191	0.189	-	1.28
case200_activ___sad	8457	1777	13527	0	40	42	-	0.00379	0.00395	-	0.152	0.166	-
case197_snem	8752	1857	14224	0	-	-	25	-	-	0.0221	-	-	0.551
case197_snem___api	8752	1857	14224	0	32	-	49	0.00403	-	0.0204	0.129	-	0.998
case197_snem___sad	8752	1857	14224	0	-	-	24	-	-	0.02	-	-	0.48
case162_ieee_dtc	9168	1882	14920	0	48	54	-	0.00449	0.00481	-	0.215	0.26	-
case162_ieee_dtc___api	9168	1882	14920	0	47	68	90	0.00464	0.00441	0.023	0.218	0.3	2.07
case162_ieee_dtc___sad	9168	1882	14920	0	46	59	95	0.00453	0.00472	0.0229	0.208	0.278	2.18
case300_ieee___api	13782	2900	21993	0	51	-	-	0.00576	-	-	0.294	-	-
case300_ieee___sad	13782	2900	21993	0	85	-	-	0.00528	-	-	0.449	-	-
case240_pserc	13772	3014	22076	0	28	35	-	0.00717	0.00575	-	0.201	0.201	-
case240_pserc___api	13772	3014	22076	0	26	39	-	0.00745	0.00706	-	0.194	0.275	-
case240_pserc___sad	13772	3014	22076	0	29	39	-	0.00735	0.00583	-	0.213	0.227	-
case588_sdet	23204	4876	37012	0	77	-	-	0.0093	-	-	0.716	-	-
case588_sdet___api	23204	4876	37012	0	68	-	-	0.0103	-	-	0.699	-	-
case588_sdet___sad	23204	4876	37012	0	78	-	-	0.00965	-	-	0.753	-	-
case500_goc	23888	5114	38653	0	45	-	-	0.0137	-	-	0.616	-	-
case500_goc___api	23888	5114	38653	0	37	-	-	0.012	-	-	0.445	-	-
case500_goc___sad	23888	5114	38653	0	43	-	-	0.0136	-	-	0.584	-	-
case793_goc	31082	6495	49764	0	48	-	-	0.0136	-	-	0.654	-	-
case793_goc___api	31082	6495	49764	0	56	-	-	0.014	-	-	0.782	-	-
case1354_pegase	62814	13258	103260	0	99	-	-	0.0281	-	-	2.78	-	-
case1354_pegase___api	62814	13258	103260	0	66	-	-	0.0327	-	-	2.16	-	-
case1354_pegase___sad	62814	13258	103260	0	85	-	-	0.0289	-	-	2.45	-	-
case1888_rte___api	81966	17208	131225	0	90	-	-	0.0384	-	-	3.46	-	-
case1888_rte___sad	81966	17208	131225	0	65	-	-	0.049	-	-	3.19	-	-
case1951_rte	84496	17817	134660	0	79	-	-	0.0486	-	-	3.84	-	-
case1951_rte___api	84496	17817	134660	0	72	-	-	0.053	-	-	3.82	-	-
case1951_rte___sad	84496	17817	134660	0	80	-	-	0.0478	-	-	3.83	-	-
case1803_snem	84794	17835	138430	0	85	-	-	0.0389	-	-	3.3	-	-
case2383wp_k___api	97600	20393	155950	0	22	32	-	0.0692	0.0529	-	1.52	1.69	-
case2383wp_k___sad	97600	20393	155950	0	104	-	-	0.0479	-	-	4.98	-	-
case2000_goc	108628	22742	178538	0	53	-	-	0.0774	-	-	4.1	-	-
case2000_goc___api	108628	22742	178538	0	51	-	-	0.0699	-	-	3.57	-	-
case2737sop_k	109822	22777	176602	0	71	-	-	0.053	-	-	3.76	-	-
case2737sop_k___api	109822	22777	176602	0	85	-	-	0.0491	-	-	4.18	-	-
case2736sp_k	110022	22878	176901	0	71	-	-	0.0502	-	-	3.57	-	-
case2736sp_k___sad	110022	22878	176901	0	77	-	-	0.0521	-	-	4.01	-	-

Table A.7: Solve times and iteration counts for the SOCP Optimal Power Flow problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
case2746wp_k___api	111106	23320	178556	0	-	-	36	-	-	0.258	-	-	9.29
case2746wp_k___sad	111106	23320	178556	0	78	-	-	0.0505	-	-	3.94	-	-
case2746wop_k	111822	23434	179829	0	69	-	-	0.0563	-	-	3.89	-	-
case2746wop_k___sad	111822	23434	179829	0	66	-	-	0.0524	-	-	3.46	-	-
case3012wp_k	120676	25202	193907	0	111	-	-	0.0565	-	-	6.27	-	-
case3012wp_k___sad	120676	25202	193907	0	117	-	-	0.0563	-	-	6.58	-	-
case2848_rte	122708	25858	197228	0	81	-	-	0.0742	-	-	6.01	-	-
case2848_rte___api	122708	25858	197228	0	87	-	-	0.084	-	-	7.31	-	-
case2848_rte___sad	122708	25858	197228	0	70	-	-	0.0722	-	-	5.05	-	-
case2868_rte	123912	26164	198869	0	85	-	-	0.0796	-	-	6.77	-	-
case2868_rte___api	123912	26164	198869	0	77	-	-	0.0779	-	-	6	-	-
case2868_rte___sad	123912	26164	198869	0	83	-	-	0.0757	-	-	6.28	-	-
case2853_sdet___sad	128886	27445	206896	0	105	-	-	0.0532	-	-	5.58	-	-
case3375wp_k	139126	29112	223999	0	93	-	-	0.0595	-	-	5.54	-	-
case3375wp_k___sad	139126	29112	223999	0	110	-	-	0.0583	-	-	6.42	-	-
case2869_pegase___api	143608	30153	236217	0	55	-	-	0.0994	-	-	5.47	-	-
case2869_pegase___sad	143608	30153	236217	0	52	-	-	0.102	-	-	5.32	-	-
case2742_goc	144110	29856	236467	0	113	-	-	0.103	-	-	11.6	-	-
case2742_goc___sad	144110	29856	236467	0	109	-	-	0.102	-	-	11.1	-	-
case4661_sdet___api	198498	41599	320728	0	131	-	-	0.107	-	-	14	-	-
case3970_goc	205819	42789	337328	0	142	-	-	0.143	-	-	20.4	-	-
case4020_goc	216455	44877	355706	0	197	-	-	0.154	-	-	30.3	-	-
case4917_goc___sad	218213	45522	352833	0	-	-	177	-	-	0.458	-	-	81
case4601_goc	225588	46885	368445	0	178	-	-	0.153	-	-	27.2	-	-
case4601_goc___sad	225588	46885	368445	0	182	-	-	0.148	-	-	26.9	-	-
case4837_goc___sad	240160	49855	392884	0	132	-	-	0.198	-	-	26.1	-	-
case4619_goc	254753	52616	419210	0	154	-	-	0.215	-	-	33.1	-	-
case4619_goc___sad	254753	52616	419210	0	150	-	-	0.214	-	-	32.1	-	-
case5658_epigrids	282180	58620	461724	0	162	-	-	0.222	-	-	35.9	-	-
case5658_epigrids___api	282180	58620	461724	0	165	-	-	0.261	-	-	43.1	-	-
case5658_epigrids___sad	282180	58620	461724	0	160	-	-	0.216	-	-	34.5	-	-
case6468_rte___api	286248	59396	463599	0	91	-	-	0.22	-	-	20	-	-
case6468_rte___sad	286248	59396	463599	0	101	-	-	0.208	-	-	21	-	-
case6470_rte	287806	60144	465757	0	115	-	-	0.201	-	-	23.1	-	-
case6470_rte___api	287806	60144	465757	0	89	-	-	0.222	-	-	19.8	-	-
case6495_rte	288050	60099	465939	0	124	-	-	0.195	-	-	24.2	-	-
case6495_rte___api	288050	60099	465939	0	111	-	-	0.197	-	-	21.9	-	-
case6495_rte___sad	288050	60099	465939	0	112	-	-	0.193	-	-	21.6	-	-
case6515_rte___sad	288710	60239	466922	0	93	-	-	0.207	-	-	19.3	-	-

Table A.7: Solve times and iteration counts for the SOCP Optimal Power Flow problem set

					iterations			time per iteration(s)			total time (s)		
Problem	vars.	cons.	nnz(A)	nnz(P)	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
case7336_epigrids___api	358450	74618	585766	0	151	-	-	0.333	-	-	50.3	-	-
case10000_goc___api	440997	92799	712177	0	67	-	-	0.392	-	-	26.2	-	-
case10480_goc___sad	573754	118760	943278	0	166	-	-	0.548	-	-	91	-	-

Table A.8: Solve times and iteration counts for the CBLIB Exponential Cone problem set

Problem	vars.	cons.	nnz(A)	nnz(P)	iterations			time per iteration(s)			total time (s)		
					ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek	ClarabelRs	ECOS	Mosek
BSS1	14	11	23	0	8	14	9	1.28e-05	8.13e-06	0.000153	0.000102	0.000114	0.00138
DEMB782	14	11	23	0	6	14	8	1.52e-05	7.99e-06	0.000154	9.1e-05	0.000112	0.00123
BSS2	20	15	33	0	8	14	9	1.59e-05	1.07e-05	0.000158	0.000127	0.000149	0.00142
DEMB781	26	19	40	0	10	16	7	1.72e-05	1.45e-05	0.000173	0.000172	0.000232	0.00121
GPTEST	32	24	48	0	8	16	9	2.12e-05	1.5e-05	0.000168	0.00017	0.00024	0.00152
RIJC781	32	24	48	0	8	16	9	2.15e-05	1.5e-05	0.000176	0.000172	0.00024	0.00158
RIJC784	40	29	66	0	10	18	10	2.46e-05	2.25e-05	0.000186	0.000246	0.000404	0.00186
RIJC785	44	33	83	0	10	17	9	2.77e-05	2.45e-05	0.000201	0.000277	0.000416	0.00181
RIJC786	44	33	83	0	8	18	8	3.01e-05	2.38e-05	0.00021	0.000241	0.000428	0.00168
RIJC782	56	40	87	0	12	17	11	3.13e-05	2.98e-05	0.00021	0.000376	0.000507	0.00231
RIJC783	74	53	124	0	9	17	11	4.34e-05	3.84e-05	0.000232	0.00039	0.000653	0.00256
BECK751	113	80	236	0	11	21	11	6.17e-05	5.58e-05	0.000271	0.000678	0.00117	0.00298
BECK752	113	80	236	0	13	23	12	6e-05	5.8e-05	0.000294	0.00078	0.00133	0.00352
BECK753	113	80	236	0	11	22	11	6.16e-05	5.41e-05	0.00027	0.000677	0.00119	0.00297
FIAC81B	122	87	201	0	13	25	13	6.23e-05	6.22e-05	0.000305	0.00081	0.00156	0.00396
FANG88	165	119	252	0	14	24	14	8.06e-05	8.44e-05	0.00033	0.00113	0.00203	0.00462
DEMB761	183	131	284	0	15	20	15	9.36e-05	9.3e-05	0.000352	0.0014	0.00186	0.00529
DEMB762	183	131	284	0	14	22	14	9.1e-05	9.1e-05	0.000361	0.00127	0.002	0.00505
DEMB763	183	131	284	0	15	20	14	8.68e-05	9.12e-05	0.000356	0.0013	0.00182	0.00499
FIAC81A	289	191	496	0	13	22	10	0.000138	0.000138	0.000544	0.0018	0.00303	0.00544
RIJC787	296	200	510	0	12	23	10	0.000143	0.000133	0.000567	0.00171	0.00306	0.00567
CAR	865	601	1584	0	19	26	10	0.00041	0.000424	0.00193	0.00779	0.011	0.0193
GP_DAVE_1	1441	705	3686	0	26	32	20	0.000683	0.000732	0.00438	0.0178	0.0234	0.0876
JHA88	1691	1131	3005	0	14	24	13	0.000813	0.000736	0.00509	0.0114	0.0177	0.0662
VARUN	2013	1346	6788	0	24	47	24	0.00131	0.00106	0.00562	0.0315	0.05	0.135
GP_DAVE_2	2489	1219	6752	0	28	37	22	0.00129	0.00133	0.00731	0.0362	0.0493	0.161
LOGEXPCR_N20_M400	3223	2022	13238	0	30	25	18	0.0017	0.00148	0.0061	0.0511	0.0369	0.11
GP_DAVE_3	3537	1733	9818	0	-	40	24	-	0.00198	0.0117	-	0.0793	0.281
LOGEXPCR_N100_M400	3303	2102	45314	0	31	27	19	0.00569	0.00508	0.00782	0.176	0.137	0.148
MRA01	5513	3681	10680	0	25	-	14	0.00271	-	0.0173	0.0678	-	0.242
LOGEXPCR_N20_M800	6423	4022	26411	0	27	28	21	0.00352	0.00295	0.013	0.0951	0.0827	0.272
LOGEXPCR_N100_M800	6503	4102	90322	0	34	30	23	0.0118	0.0107	0.0168	0.401	0.322	0.387
LOGEXPCR_N20_M1200	9623	6022	39574	0	27	27	21	0.00561	0.00492	0.0193	0.151	0.133	0.406
LOGEXPCR_N20_M1600	12823	8022	52729	0	28	29	20	0.00775	0.0105	0.0254	0.217	0.306	0.507
LOGEXPCR_N20_M2000	16023	10022	65872	0	31	33	23	0.00994	0.0133	0.0314	0.308	0.44	0.721
MRA02	21965	14606	50050	0	28	-	17	0.0121	-	0.0549	0.339	-	0.933
CX02_100	31087	20693	56430	0	19	28	13	0.0159	0.017	0.0822	0.301	0.477	1.07
CX02_200	122187	81393	222880	0	22	35	13	0.0762	0.0887	0.36	1.68	3.11	4.68

References

[1] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[2] Douglas A. Allan and James B. Rawlings. "Moving Horizon Estimation". In: *Handbook of Model Predictive Control*. Cham: Springer International Publishing, 2019, pp. 99–124. URL: https://doi.org/10.1007/978-3-319-77489-3_5.

[3] A. Makrodimopoulos and C. M. Martin. "Lower bound limit analysis of cohesive-frictional materials using second-order cone programming". In: *International Journal for Numerical Methods in Engineering* 66.4 (2006), pp. 604–634. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1567. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1567.

[4] A. Makrodimopoulos and C. M. Martin. "Upper bound limit analysis using simplex strain elements and second-order cone programming". In: *International Journal for Numerical and Analytical Methods in Geomechanics* 31.6 (2007), pp. 835–865. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nag.567. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nag.567.

[5] Paul J. Goulart and Sergei Chernyshenko. "Global stability analysis of fluid flows using sum-of-squares". In: *Physica D: Nonlinear Phenomena* 241.6 (2012), pp. 692–704. URL: https://www.sciencedirect.com/science/article/pii/S0167278911003575.

[6] P.L. Combettes. "The Convex Feasibility Problem in Image Recovery". In: Advances in Imaging and Electron Physics 95 (1996). Ed. by Peter W. Hawkes, pp. 155–270. URL: https://www.sciencedirect.com/science/article/pii/S1076567008701575.

[7] C. Cortes and V. Vapnik. "Support-Vector Networks". In: *Machine Learning* 20.3 (1995), pp. 273–297.

[8] R. Tibshirani. "Regression shrinkage and selection via the Lasso". In: *Journal of the Royal Statistical Society: Series B* 58.1 (1996), pp. 267–288.

[9] E. J. Candés, M. B. Wakin, and S. Boyd. "Enhancing Sparsity by Reweighted ℓ_1 Minimization". In: *Journal of Fourier Analysis and Applications* 14.5 (2008), pp. 877–905.

[10] Stephen Boyd et al. "A tutorial on geometric programming". In: *Optimization and Engineering* 8.1 (2007), pp. 67–127. URL: https://doi.org/10.1007/s11081-007-9001-7.

[11] G. Cornuejols and R. Tütüncü. *Optimization Methods in Finance*. Mathematics, Finance and Risk. Cambridge University Press, 2006.

[12] H. Markowitz. "PORTFOLIO SELECTION". In: *The Journal of Finance* 7.1 (1952), pp. 77–91.

[13] Y. E. Nesterov and M. J. Todd. "Self-Scaled Barriers and Interior-Point Methods for Convex Programming". In: *Mathematics of Operations Research* 22.1 (1997), pp. 1–42. (Visited on 11/08/2022).

[14] Y. E. Nesterov and M. J. Todd. "Primal-Dual Interior-Point Methods for Self-Scaled Cones". In: *SIAM Journal on Optimization* 8.2 (1998), pp. 324–364.

[15] Yurii Nesterov. "Towards non-symmetric conic optimization". In: *Optimization Methods and Software* 27.4-5 (2012), pp. 893–917.

[16] Martin S. Andersen, Joachim Dahl, and Lieven Vandenberghe. "Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones". In: *Mathematical Programming Computation* 2.3 (2010), pp. 167–201.

[17] Dávid Papp and Sercan Yildiz. "Sum-of-Squares Optimization without Semidefinite Programming". In: *SIAM Journal on Optimization* 29.1 (2019), pp. 822–851.

[18] Brendan O'Donoghue et al. "Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding". In: *Journal of Optimization Theory and Applications* 169.3 (2016), pp. 1042–1068.

[19] Michael Garstka, Mark Cannon, and Paul Goulart. "COSMO: A Conic Operator Splitting Method for Convex Conic Problems". In: *Journal of Optimization Theory and Applications* 190.3 (2021), pp. 779–810.

[20] Stephen Boyd et al. "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Foundations and Trends in Machine Learning* 3.1 (2011), pp. 1–122.

[21] Miles Lubin. "Mixed-integer convex optimization : outer approximation algorithms and modeling power". PhD thesis. Massachusetts Institute of Technology, 2017.

[22] Alberto Bemporad and Manfred Morari. "Control of systems integrating logic, dynamics, and constraints". In: *Automatica* 35.3 (1999), pp. 407–427.

[23] Hande Y. Benson and Ümit Sağlam. "Mixed-Integer Second-Order Cone Programming: A Survey". In: *Theory Driven by Influential Applications*. INFORMS, 2014, pp. 13–36.

[24] Bartolomeo Stellato, Tobias Geyer, and Paul J. Goulart. "High-Speed Finite Control Set Model Predictive Control for Power Electronics". In: *IEEE Transactions on Power Electronics* 32.5 (2017), pp. 4007–4020.

[25] Kazuo Yonekura and Yoshihiro Kanno. "Global optimization of robust truss topology via mixed integer semidefinite programming". In: *Optimization and Engineering* 11.3 (2010), pp. 355–379.

[26] Tobias Achterberg et al. "Presolve Reductions in Mixed Integer Programming". In: *INFORMS Journal on Computing* 32.2 (2020), pp. 473–506.

[27] Timo Berthold, Andrea Lodi, and Domenico Salvagnin. "Ten years of feasibility pump, and counting". In: *EURO Journal on Computational Optimization* 7.1 (2019), pp. 1–14.

[28] D. Axehill and A. Hansson. "A Mixed Integer Dual Quadratic Programming Algorithm Tailored for MPC". In: *Proceedings of the 45th IEEE Conference on Decision and Control*. 2006, pp. 5693–5698.

[29] J. Liang, S. D. Cairano, and R. Quirynen. "Early Termination of Convex QP Solvers in Mixed-Integer Programming for Real-Time Decision Making". In: *IEEE Control Systems Letters* 5.4 (2021), pp. 1417–1422.

[30] Steven J. Benson, Yinyu Ye, and Xiong Zhang. "Solving Large-Scale Sparse Semidefinite Programs for Combinatorial Optimization". In: *SIAM Journal on Optimization* 10.2 (2000), pp. 443–461.

[31] Yongwei Huang and Daniel P. Palomar. "Rank-Constrained Separable Semidefinite Programming With Applications to Optimal Beamforming". In: *IEEE Transactions on Signal Processing* 58.2 (2010), pp. 664–678.

[32] Stephen Boyd et al. *Linear Matrix Inequalities in System and Control Theory*. Philadelphia: Society for Industrial and Applied Mathematics, 1994.

[33] Murat A. Erdogdu et al. "Convergence Rate of Block-Coordinate Maximization Burer-Monteiro Method for Solving Large SDPs". In: *Mathematical Programming* (2021).

[34] Mehdi Karimi and Levent Tunçel. "Domain-Driven Solver (DDS) Version 2.1: a MATLAB-based software package for convex optimization problems in domain-driven form". In: *Mathematical Programming Computation* (Oct. 2023). URL: https://doi.org/10.1007/s12532-023-00248-2.

[35] Chris Coey, Lea Kapelevich, and Juan Pablo Vielma. "Solving Natural Conic Formulations with Hypatia.jl". In: *INFORMS Journal on Computing* 34.5 (2022), pp. 2686–2699.

[36] Vihangkumar V. Naik and Alberto Bemporad. "Embedded Mixed-Integer Quadratic Optimization using Accelerated Dual Gradient Projection". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 10723–10728.

[37] Stephen J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, 1997. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611971453. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611971453.

[38] Yinyu Ye, Michael J. Todd, and Shinji Mizuno. "An $O(\sqrt{nL})$-Iteration Homogeneous and Self-Dual Linear Programming Algorithm". In: *Mathematics of Operations Research* 19.1 (1994), pp. 53–67.

[39] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[40] MOSEK ApS. *MOSEK Optimization Suite 10.0*. 2023. URL: https://docs.mosek.com/latest/intro/index.html.

[41] Lieven Vandenberghe. *The CVXOPT linear and quadratic cone program solvers*. 2010.

[42] Heinz H. Bauschke and Patrick L. Combettes. *Convex analysis and monotone operator theory in hilbert spaces*. eng. 2nd ed. CMS books in mathematics. Cham: Springer, 2017.

[43] Michael Garstka. *Operator splitting methods for large convex conic programs*. 2021.

[44] R. Tyrrell Rockafellar and Roger J.-B. Wets. *Variational Analysis*. Springer Verlag, 1998.

[45] Alexander Domahidi, Eric Chu, and Stephen Boyd. "ECOS: An SOCP solver for embedded systems". In: *2013 European Control Conference (ECC)*. 2013, pp. 3071–3076.

[46] Paul J. Goulart and Yuwen Chen. *Clarabel: An interior-point solver for conic programs with quadratic objectives*. 2024. arXiv: 2405.12762 [math.OC].

[47] Y. Nesterov, M. J. Todd, and Y. Ye. "Infeasible-start primal-dual methods and infeasibility detectors for nonlinear programming problems". In: *Mathematical*

Programming 84.2 (1999), pp. 227–267. URL: https://doi.org/10.1007/s10107980009a.

[48] Anders Skajaa and Yinyu Ye. "A homogeneous interior-point algorithm for nonsymmetric convex conic optimization". In: *Mathematical Programming* 150.2 (2015), pp. 391–422.

[49] Lieven Vandenberghe and Stephen Boyd. "Semidefinite Programming". In: *SIAM Review* 38.1 (1996), pp. 49–95.

[50] Stephen Boyd and Lieven Vandenberghe. "Semidefinite Programming Relaxations of Non-Convex Problems in Control and Combinatorial Optimization". In: *Communications, Computation, Control, and Signal Processing: a tribute to Thomas Kailath*. Boston, MA: Springer US, 1997, pp. 279–287.

[51] Adrian Gepp, Geoff Harris, and Bruce Vanstone. "Financial applications of semidefinite programming: a review and call for interdisciplinary research". In: *Accounting and Finance* 60.4 (Dec. 2020), pp. 3527–3555.

[52] Christoph Helmberg et al. "An Interior-Point Method for Semidefinite Programming". In: *SIAM Journal on Optimization* 6.2 (1996), pp. 342–361.

[53] Anirudha Majumdar, Georgina Hall, and Amir Ali Ahmadi. "Recent Scalability Improvements for Semidefinite Programming with Applications in Machine Learning, Control, and Robotics". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.1 (2020), pp. 331–360.

[54] Mituhiro Fukuda et al. "Exploiting Sparsity in Semidefinite Programming via Matrix Completion I: General Framework". In: *SIAM Journal on Optimization* 11.3 (Mar. 2000), pp. 647–674.

[55] Lieven Vandenberghe and Martin S. Andersen. "Chordal Graphs and Semidefinite Optimization". In: *Foundations and Trends® in Optimization* 1.4 (May 2015), pp. 241–433.

[56] Yang Zheng et al. "Chordal decomposition in operator-splitting methods for sparse semidefinite programs". In: *Mathematical Programming* 180.1 (2020), pp. 489–532.

[57] Amir Ali Ahmadi and Anirudha Majumdar. "DSOS and SDSOS Optimization: More Tractable Alternatives to Sum of Squares and Semidefinite Optimization". In: *SIAM Journal on Applied Algebra and Geometry* 3.2 (2019), pp. 193–230.

[58] Alex Lemon, Anthony Man-Cho So, and Yinyu Ye. "Low-Rank Semidefinite Programming: Theory and Applications". In: *Foundations and Trends® in Optimization* 2.1-2 (2016), pp. 1–156.

[59] Renato D.C. Monteiro Samuel Burer. "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization". In: *Mathematical Programming* 95.2 (2003), pp. 329–357.

[60] Robert M. Freund, Paul Grigas, and Rahul Mazumder. "An Extended Frank-Wolfe Method with "In-Face" Directions, and its Application to Low-Rank Matrix Completion". In: *SIAM Journal on Optimization* 27.1 (2017), pp. 319–346.

[61] Alp Yurtsever et al. "Scalable Semidefinite Programming". In: *SIAM Journal on Mathematics of Data Science* 3.1 (2021), pp. 171–200.

[62] Martin Jaggi. "Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization". In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28. Proceedings of Machine Learning Research 1. 2013, pp. 427–435.

[63] David P. Woodruff. "Sketching as a Tool for Numerical Linear Algebra". In: *Foundations and Trends® in Theoretical Computer Science* 10.1–2 (2014), pp. 1–157.

[64] John C Duchi et al. "Conic Descent and its Application to Memory-efficient Optimization over Positive Semidefinite Matrices". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. 2020, pp. 8308–8317.

[65] Lijun Ding et al. "An Optimal-Storage Approach to Semidefinite Programming Using Approximate Complementarity". In: *SIAM Journal on Optimization* 31.4 (2021), pp. 2695–2725.

[66] Po-Wei Wang, Wei-Cheng Chang, and J. Zico Kolter. "The Mixing method: low-rank coordinate descent for semidefinite programming with diagonal constraints". In: *arXiv preprint arXiv:1706.00476* (2017).

[67] Farid Alizadeh, Jean-Pierre A. Haeberly, and Michael L. Overton. "Complementarity and nondegeneracy in semidefinite programming". In: *Mathematical Programming* 77.1 (1997), pp. 111–128.

[68] Gábor Pataki. "On the Rank of Extreme Matrices in Semidefinite Programs and the Multiplicity of Optimal Eigenvalues". In: *Mathematics of Operations Research* 23.2 (1998), pp. 339–358.

[69] Nicolas Boumal, Vlad Voroninski, and Afonso Bandeira. "The non-convex Burer-Monteiro approach works on smooth semidefinite programs". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. 2016, pp. 2765–2773.

[70] Irène Waldspurger and Alden Waters. "Rank Optimality for the Burer–Monteiro Factorization". In: *SIAM Journal on Optimization* 30.3 (2020), pp. 2577–2602.

[71] Bo Jiang et al. "Structured nonconvex and nonsmooth optimization: algorithms and iteration complexity analysis". In: *Computational Optimization and Applications* 72.1 (2019), pp. 115–157.

[72] Jonathan Eckstein and Dimitri P. Bertsekas. "On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators". In: *Mathematical Programming* 55.1 (1992), pp. 293–318.

[73] G. Banjac et al. "Infeasibility detection in the alternating direction method of multipliers for convex optimization". In: *Journal of Optimization Theory and Applications* 183.2 (2019), pp. 490–519.

[74] Rongjie Lai and Stanley Osher. "A Splitting Method for Orthogonality Constrained Problems". In: *Journal of Scientific Computing* 58.2 (2014), pp. 431–449.

[75] Artiom Kovnatsky, Klaus Glashoff, and Michael M. Bronstein. "MADMM: A Generic Algorithm for Non-smooth Optimization on Manifolds". In: vol. 9909. Lecture Notes in Computer Science. 2016, pp. 680–696.

[76] Songtao Lu, Mingyi Hong, and Zhengdao Wang. "A Nonconvex Splitting Method for Symmetric Nonnegative Matrix Factorization: Convergence Analysis and Optimality". In: *IEEE Transactions on Signal Processing* 65.12 (2017), pp. 3120–3135.

[77] Seungil You and Qiuyu Peng. "A non-convex alternating direction method of multipliers heuristic for optimal power flow". In: *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. 2014, pp. 788–793.

[78] Rina Foygel Barber and Emil Y. Sidky. "Convergence for nonconvex ADMM, with applications to CT imaging". In: *arXiv preprint arXiv:2006.07278* (2021).

[79] Mingyi Hong, Zhi-Quan Luo, and Meisam Razaviyayn. "Convergence Analysis of Alternating Direction Method of Multipliers for a Family of Nonconvex Problems". In: *SIAM Journal on Optimization* 26.1 (2016), pp. 337–364.

[80] Yu Wang, Wotao Yin, and Jinshan Zeng. "Global Convergence of ADMM in Nonconvex Nonsmooth Optimization". In: *Journal of Scientific Computing* 78.1 (2019), pp. 29–63.

[81] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. "Proximal alternating linearized minimization for nonconvex and nonsmooth problems". In: *Mathematical Programming* 146.1 (2014), pp. 459–494.

[82] Michel X. Goemans and David P. Williamson. "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming". In: *Journal of the ACM* 42.6 (1995), pp. 1115–1145.

[83] Murat A Erdogdu, Yash Deshpande, and Andrea Montanari. "Inference in Graphical Models via Semidefinite Programming Hierarchies". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[84] Afonso S. Bandeira, Nicolas Boumal, and Vladislav Voroninski. "On the low-rank approach for semidefinite programs arising in synchronization and community detection". In: *Proceedings of the 29th Conference on Learning Theory, COLT*. Vol. 49. JMLR Workshop and Conference Proceedings. 2016, pp. 361–382.

[85] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods". In: *Mathematical Programming* 137.1 (2013), pp. 91–129.

[86] Hédy Attouch et al. "Proximal Alternating Minimization and Projection Methods for Nonconvex Problems: An Approach Based on the Kurdyka-Łojasiewicz Inequality". In: *Mathematics of Operations Research* 35.2 (2010), pp. 438–457.

[87] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. USA: Princeton University Press, 2007.

[88] Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge: Cambridge University Press, June 2023. URL: `https://www.nicolasboumal.net/book`.

[89] Song Mei et al. "Solving SDPs for synchronization and MaxCut problems via the Grothendieck inequality". In: *Proceedings of the 2017 Conference on Learning Theory*. Vol. 65. Proceedings of Machine Learning Research. 2017, pp. 1476–1515.

[90] Ronny Bergmann. "Manopt.jl: Optimization on Manifolds in Julia". In: *Journal of Open Source Software* 7.70 (2022), p. 3866.

[91] Peter Robert Chares. "Cones and interior-point Algorithms for Structured Convex Optimization involving Powers and Exponentials". PhD thesis. Universit´e catholique de Louvain, 2009.

[92] Joachim Dahl and Erling D. Andersen. "A primal-dual interior-point algorithm for nonsymmetric exponential-cone optimization". In: *Mathematical Programming* (2021).

[93] Santiago Akle Serrano. "Algorithms for unsymmetric cone optimization and an implementation for problems with the exponential cone". PhD thesis. Stanford University, 2015. URL: `https://search.proquest.com/docview/1709297118`.

[94] Yinyu Ye. "On homogeneous and self-dual algorithms for LCP". en. In: *Mathematical Programming* 76.1 (Jan. 1997), pp. 211–221. URL: https://doi.org/10.1007/BF02614384 (visited on 11/24/2023).

[95] Erling D. Andersen and Yinyu Ye. "On a homogeneous algorithm for the monotone complementarity problem". In: *Mathematical Programming* 84.2 (1999), pp. 375–399. URL: https://doi.org/10.1007/s101070050027.

[96] Akiko Yoshise. "Interior Point Trajectories and a Homogeneous Model for Nonlinear Complementarity Problems over Symmetric Cones". In: *SIAM Journal on Optimization* 17.4 (2007), pp. 1129–1153. eprint: https://doi.org/10.1137/04061427X. URL: https://doi.org/10.1137/04061427X.

[97] Akiko Yoshise. "Homogeneous Algorithms for Monotone Complementarity Problems over Symmetric Cones". In: *Pacific Journal of Optimization* 5 (Jan. 2007).

[98] Csaba Meszaros. "The practical behavior of the homogeneous self-dual formulations in interior point methods". In: *Central European Journal of Operations Research* 23.4 (Dec. 2015), pp. 913–924. URL: https://doi.org/10.1007/s10100-013-0336-1.

[99] Brendan O'Donoghue. "Operator Splitting for a Homogeneous Embedding of the Linear Complementarity Problem". In: *SIAM Journal on Optimization* 31.3 (2021), pp. 1999–2023. eprint: https://doi.org/10.1137/20M1366307. URL: https://doi.org/10.1137/20M1366307.

[100] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Berlin: Springer, 2006.

[101] Paul T. Boggs and Jon W. Tolle. "Sequential Quadratic Programming". In: *Acta Numerica* 4 (1995), pp. 1–51.

[102] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers*. Athena Scientific, 2005.

[103] David R. Morrison et al. "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning". In: *Discrete Optimization* 19 (2016), pp. 79–102. URL: https://www.sciencedirect.com/science/article/pii/S1572528616000062.

[104] Goran Banjac et al. "Infeasibility Detection in the Alternating Direction Method of Multipliers for Convex Optimization". In: *Journal of Optimization Theory and Applications* 183.2 (2019), pp. 490–519. URL: https://doi.org/10.1007/s10957-019-01575-y.

[105] Lea Kapelevich, Erling D. Andersen, and Juan Pablo Vielma. "Computing Conjugate Barrier Information for Nonsymmetric Cones". In: *Journal of Optimization Theory and Applications* (2022).

[106] Levent Tunçel. "Generalization of Primal-Dual Interior-Point Methods to Convex Optimization Problems in Conic Form". In: *Foundations of Computational Mathematics* 1.3 (2001), pp. 229–254.

[107] Robert J. Vanderbei. "Symmetric Quasidefinite Matrices". In: *SIAM Journal on Optimization* 5.1 (1995), pp. 100–113. eprint: https://doi.org/10.1137/0805005. URL: https://doi.org/10.1137/0805005.

[108] Philip E. Gill, Michael A. Saunders, and Joseph R. Shinnerl. "On the Stability of Cholesky Factorization for Symmetric Quasidefinite Systems". In: *SIAM Journal*

on Matrix Analysis and Applications 17.1 (1996), pp. 35–46. eprint: https://doi.org/10.1137/S0895479893252623. URL: https://doi.org/10.1137/S0895479893252623.

[109] Timothy A. Davis. "Algorithm 849: A Concise Sparse Cholesky Factorization Package". In: *ACM Trans. Math. Softw.* 31.4 (2005), pp. 587–591. URL: https://doi.org/10.1145/1114268.1114277.

[110] Behçet Açıkmeşe, John M. Carson, and Lars Blackmore. "Lossless Convexification of Nonconvex Control Bound and Pointing Constraints of the Soft Landing Optimal Control Problem". In: *IEEE Transactions on Control Systems Technology* 21.6 (2013), pp. 2104–2113.

[111] Sogol Babaeinejadsarookolaee et al. "The power grid library for benchmarking AC optimal power flow algorithms". In: *arXiv:1908.02788* (2019).

[112] Sanjay Mehrotra. "On the Implementation of a Primal-Dual Interior Point Method". In: *SIAM Journal on Optimization* 2.4 (1992), pp. 575–601.

[113] Steve Klabnik and Carol Nichols. *The Rust programming language*. No Starch Press, 2023.

[114] Jeff Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1 (2017), pp. 65–98. URL: https://doi.org/10.1137/141000671.

[115] Steven Diamond and Stephen Boyd. "CVXPY: A Python-embedded modeling language for convex optimization". In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.

[116] Akshay Agrawal et al. "A rewriting system for convex optimization problems". In: *Journal of Control and Decision* 5.1 (2018), pp. 42–60.

[117] *CVXPY*. https://www.cvxpy.org/. Accessed: 2023-11-24.

[118] B. Stellato et al. "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672. URL: https://doi.org/10.1007/s12532-020-00179-2.

[119] L Susan Blackford et al. "An updated set of basic linear algebra subprograms (BLAS)". In: *ACM Transactions on Mathematical Software* 28.2 (2002), pp. 135–151.

[120] Yanqing Chen et al. "Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate". In: *ACM Trans. Math. Softw.* 35.3 (Oct. 2008). URL: https://doi.org/10.1145/1391989.1391995.

[121] Olaf Schenk and Klaus Gärtner. "PARDISO". In: *Encyclopedia of Parallel Computing*. Ed. by David Padua. Boston, MA: Springer US, 2011, pp. 1458–1464. URL: https://doi.org/10.1007/978-0-387-09766-4_90.

[122] Iain S. Duff. "MA57—a Code for the Solution of Sparse Symmetric Definite and Indefinite Systems". In: *ACM Trans. Math. Softw.* 30.2 (June 2004), pp. 118–144. URL: https://doi.org/10.1145/992200.992202.

[123] William Kahan. "IEEE standard 754 for binary floating-point arithmetic". In: *Lecture Notes on the Status of IEEE* 754.94720-1776 (1996), p. 11.

[124] E Michael Gertz and Stephen J Wright. "Object-oriented software for quadratic programming". In: *ACM Transactions on Mathematical Software (TOMS)* 29.1 (2003), pp. 58–81.

[125] Nevena Milojkovic, Mohammad Ghafari, and Oscar Nierstrasz. "It's Duck (Typing) Season!" In: *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*. 2017, pp. 312–315.

[126] Q. Huangfu and J. A. J. Hall. "Parallelizing the dual revised simplex method". In: *Mathematical Programming Computation* 10.1 (2018), pp. 119–142. URL: https://doi.org/10.1007/s12532-017-0130-5.

[127] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2023. URL: https://www.gurobi.com.

[128] Andrew Richards. *University of Oxford Advanced Research Computing*. Aug. 2015. URL: https://doi.org/10.5281/zenodo.22558.

[129] Miles Lubin et al. "JuMP 1.0: Recent improvements to a modeling language for mathematical optimization". In: *Mathematical Programming Computation* (2023).

[130] Paul Goulart and Yuwen Chen. *Clarabel solver Benchmark suite*. July 2023. URL: https://github.com/oxfordcontrol/ClarabelBenchmarks.

[131] H. Mittelmann. *Benchmarks for optimization software*. http://plato.asu.edu/bench.html. Accessed: 2019-09-08.

[132] Elizabeth D. Dolan and Jorge J. Moré. "Benchmarking optimization software with performance profiles". In: *Mathematical Programming* 91.2 (Jan. 2002), pp. 201–213.

[133] I. Maros and C. Mészáros. "A repository of convex quadratic programming problems". In: *Optimization Methods and Software* 11.1-4 (1999), pp. 671–681.

[134] T. A. Davis and Y. Hu. "The University of Florida Sparse Matrix Collection". In: *ACM Trans. Math. Softw.* 38.1 (Dec. 2011), 1:1–1:25.

[135] P. J. Huber. "Robust estimation of a location parameter". In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101.

[136] P. J. Huber. *Robust Statistics*. John Wiley & Sons, 1981.

[137] O. L. Mangasarian and D. R. Musicant. "Robust linear and support vector regression". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.9 (2000), pp. 950–955.

[138] D. Kouzoupis et al. "Towards proper assessment of QP algorithms for embedded model predictive control". In: *2015 European Control Conference (ECC)*. 2015, pp. 2609–2616.

[139] Carleton Coffrin et al. "PowerModels.jl: An Open-Source Framework for Exploring Power Flow Formulations". In: *2018 Power Systems Computation Conference (PSCC)*. June 2018, pp. 1–8.

[140] Daniel K Molzahn and Ian A Hiskens. "A survey of relaxations and approximations of the power flow equations". In: *Foundations and Trends in Electric Energy Systems* 4.1-2 (2019), pp. 1–221.

[141] Carleton James Coffrin and Line Alnaes Roald. *Convex Relaxations in Power System Optimization, A Brief Introduction*. July 2018. URL: https://www.osti.gov/biblio/1461380.

[142] B. Stott, J. Jardim, and O. Alsac. "DC Power Flow Revisited". In: *IEEE Transactions on Power Systems* 24.3 (Aug. 2009), pp. 1290–1300.

[143] R. A. Jabr. "Radial distribution load flow using conic programming". In: *IEEE Transactions on Power Systems* 21.3 (Aug. 2006), pp. 1458–1459.

[144] Henrik A. Friberg. "CBLIB 2014: a benchmark library for conic mixed-integer and continuous optimization". In: *Mathematical Programming Computation* 8.2 (2016), pp. 191–214. URL: https://cblib.zib.de/.

[145] Dávid Papp and Sercan Yıldız. *On "A Homogeneous Interior-Point Algorithm for Non-Symmetric Convex Conic Optimization"*. June 2018.

[146] Dávid Papp and Sercan Yıldız. "Alfonso: Matlab Package for Nonsymmetric Conic Optimization". In: *INFORMS Journal on Computing* 34.1 (2022), pp. 11–19.

[147] Scott Roy and Lin Xiao. "On self-concordant barriers for generalized power cones". In: *Optimization Letters* 16.2 (2022), pp. 681–694.

[148] Robert J. Vanderbei. "Symmetric Quasidefinite Matrices". In: *SIAM Journal on Optimization* 5.1 (1995), pp. 100–113.

[149] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. New York, NY: Springer New York, 2006. URL: `https://books.google.co.uk/books?id=VbHYoSyelFcC`.

[150] T. Terlaky and J. Ph. Vial. "Computing Maximum Likelihood Estimators of Convex Density Functions". In: *SIAM J. Sci. Comput.* 19.2 (Jan. 1998). Publisher: Society for Industrial and Applied Mathematics, pp. 675–694. URL: `https://epubs.siam.org/doi/10.1137/S1064827595286578` (visited on 01/09/2024).

[151] Chris Coey, Lea Kapelevich, and Juan Pablo Vielma. "Performance enhancements for a generic conic interior point algorithm". In: *Mathematical Programming Computation* 15.1 (2023), pp. 53–101.

[152] Timo Berthold. "Primal Heuristics for Mixed Integer Programs". PhD thesis. Technische Universität Berlin, 2006.

[153] Emilie Danna, Edward Rothberg, and Claude Le Pape. "Exploring relaxation induced neighborhoods to improve MIP solutions". In: *Mathematical Programming* 102.1 (2005), pp. 71–90.

[154] Edward Rothberg. "An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions". In: *INFORMS Journal on Computing* 19.4 (2007), pp. 534–541.

[155] Roger Fletcher and Sven Leyffer. "Numerical Experience with Lower Bounds for MIQP Branch-And-Bound". In: *SIAM Journal on Optimization* 8.2 (1998), pp. 604–616.

[156] Christoph Buchheim et al. "A Feasible Active Set Method with Reoptimization for Convex Quadratic Mixed-Integer Programming". In: *SIAM Journal on Optimization* 26.3 (2016), pp. 1695–1714.

[157] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[158] Yuwen Chen and Paul Goulart. "An Early Termination Technique for ADMM in Mixed Integer Conic Programming". In: *European Control Conference (ECC)*. 2022, pp. 60–65.

[159] Yuwen Chen, Catherine Ning, and Paul Goulart. "A Unified Early Termination Technique for Primal-Dual Algorithms in Mixed Integer Conic Programming". In: *IEEE Control Systems Letters* 7 (2023), pp. 2803–2808.

[160] B. Stellato et al. "Embedded Mixed-Integer Quadratic optimization Using the OSQP Solver". In: *European Control Conference (ECC)*. 2018, pp. 1536–1541.

[161] David Applegate et al. "Practical large-scale linear programming using primal-dual hybrid gradient". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20243–20257.

[162] Neal Parikh and Stephen Boyd. "Proximal Algorithms". In: *Found. Trends Optim.* 1.3 (2014), pp. 127–239.

[163] C. V. Rao, S. J. Wright, and J. B. Rawlings. "Application of Interior-Point Methods to Model Predictive Control". In: *Journal of Optimization Theory and Applications* 99.3 (1998), pp. 723–757.

[164] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, 2020.

[165] Michel Schubiger, Goran Banjac, and John Lygeros. "GPU Acceleration of ADMM for Large-Scale Quadratic Programming". In: *Journal of Parallel and Distributed Computing* 144 (2020), pp. 55–67.

[166] Haihao Lu and Jinwen Yang. *cuPDLP.jl: A GPU Implementation of Restarted Primal-Dual Hybrid Gradient for Linear Programming in Julia*. 2023.

[167] Piers W. Lawrence, Robert M. Corless, and David J. Jeffrey. "Algorithm 917: Complex Double-Precision Evaluation of the Wright ω Function". In: *ACM Trans. Math. Software* 38.3 (2012).

[168] Endre Süli and David F. Mayers. *An Introduction to Numerical Analysis*. Cambridge: Cambridge University Press, 2003.