Interior point and outer approximation methods for conic optimization

by
Christopher Daniel Lang Coey
A.B. (summa cum laude), Harvard College (2013)

submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Signature of author .....
Sloan School of Management
May 4, 2022

Certified by .....
Juan Pablo Vielma Centeno
Research Scientist, Google
Thesis supervisor

Certified by .....
Georgia Perakis
William F. Pounds Professor of Management Science
Co-director, Operations Research Center
Thesis supervisor

Accepted by .....
Patrick Jaillet
Dugald C. Jackson Professor
Department of Electrical Engineering and Computer Science
Co-director, Operations Research Center

Interior point and outer approximation methods for conic optimization

by

Chris Coey

submitted to the MIT Sloan School of Management on May 4, 2022
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

Any convex optimization problem may be represented as a conic problem that minimizes a linear function over the intersection of an affine subspace with a convex cone. An advantage of representing convex problems in conic form is that, under certain regularity conditions, a conic problem has a simple and easily checkable certificate of optimality, primal infeasibility, or dual infeasibility. As a natural generalization of linear programming duality, conic duality allows us to design powerful algorithms for continuous and mixed-integer convex optimization.

The main goal of this thesis is to improve the generality and practical performance of (i) interior point methods for continuous conic problems and (ii) outer approximation methods for mixed-integer conic problems. We implement our algorithms in extensible open source solvers accessible through the convenient modeling language JuMP. From around 50 applied examples, we formulate continuous and mixed-integer problems over two dozen different convex cone types, many of which are new. Our extensive computational experiments with these examples explore which algorithmic features and what types of equivalent conic formulations lead to the best performance.

Thesis Supervisor: Georgia Perakis
Title: William F. Pounds Professor of Management Science; Co-director, Operations Research Center

Thesis Supervisor: Juan Pablo Vielma Centeno
Title: Research Scientist, Google

Acknowledgements

Thank you to my parents John and Heather, my family and friends, my doctoral advisor Juan Pablo Vielma, my thesis supervisor Georgia Perakis, my collaborators Miles Lubin and Lea Kapelevich, my ORC colleagues and the ORC staff, my undergraduate advisors Özlem Ergun and David C. Parkes, and my teachers in Australia and at Harvard and MIT.

Contents

Chapter 0

Introduction

0.1 Overview of chapters

In Chapter 1, we introduce our conic interior point method (IPM) solver, Hypatia. Hypatia's generic cone interface allows defining new cones by implementing logarithmically homogeneous self-concordant barrier (LHSCB) oracles. We describe Hypatia's algorithm, which generalizes the nonsymmetric conic IPM by Skajaa and Ye (2015), and we develop advanced IPM stepping techniques to enhance its practical performance. On a diverse benchmark set of conic instances, our IPM stepping enhancements reduce iteration counts and solve times by over 80% and 70% respectively.

In Chapter 2, we argue that although many convex problems are representable with conic extended formulations (EFs) using only the small number of standard cones currently recognized by advanced conic solvers (such as MOSEK 9), standard conic EFs can be much larger and more complex than natural formulations (NFs) over cones supported by Hypatia. We focus on three broad classes of cones in Hypatia: the positive semidefinite slice cones, the infinity/spectral norm cones, and the spectral function cones. We derive efficient and numerically stable LHSCB oracles for the first two classes here and for the third class in Chapter 3. For seven applied examples over these cones, we demonstrate significant computational advantages from solving the NFs with Hypatia compared to solving the EFs with either Hypatia or MOSEK 9.

In Chapter 3, we consider the class of spectral function cones, which we define from epigraphs and perspectives of spectral functions on Euclidean Jordan algebras. For two common and useful subclasses - the root-determinant cones and the matrix monotone derivative cones - we propose LHSCBs with nearly optimal barrier parameters. We derive efficient, numerically stable procedures for LHSCB oracles, including closed form inverse Hessian operators. For four applied examples over these cones, Hypatia solves the NFs more efficiently than Hypatia, MOSEK 9, or ECOS solves the equivalent EFs.

In Chapter 4, we present the first conic-duality-based branch-and-bound outer approximation method (OAM) for mixed-integer conic (MI-conic) problems. This finite-time algorithm refines polyhedral relaxations of the conic constraints using $\mathcal{K}^*$ cuts derived from conic certificates for continuous subproblems. We describe our practical implementations in Pajarito solver, which

combine the power of external mixed-integer linear solvers and continuous primal-dual conic solvers. On a library of mixed-integer second order cone problems, Pajarito greatly outperforms Bonmin and is competitive with CPLEX. On applied examples over standard cones, we demonstrate the value of our $\mathcal{K}^*$ cut techniques that strengthen the polyhedral relaxations.

In Chapter 5, we introduce our new open source MI-conic OAM solver, MOIPajarito (the successor of Pajarito). Like Hypatia, MOIPajarito has a generic cone interface. We describe $\mathcal{K}^*$ cut oracles for the three broad classes of cones from Chapter 2, enabling us to solve NFs for a dozen new applied examples. We develop MI-conic formulations for disjunctions and relaxations of common types of nonconvex constraints, by homogenizing advanced piecewise linear formulations. One of our formulations is the first logarithmic-sized mixed-integer convex formulation for a finite union of convex sets with different recession cones. Using particular conic EFs, we accelerate MOIPajarito's OAM and extend the MI-conic relaxation techniques to the high-dimensional nonconvex setting.

0.2 Main sources and collaborations

Chapter 1 follows Coey, Kapelevich, and Vielma (2021d), though material on PSD slice oracles has been moved to Chapter 2, and some material from Coey, Kapelevich, and Vielma (2021e) on Hypatia solver has been added. Chapter 2 loosely follows Coey, Kapelevich, and Vielma (2021e), and we have added new cone and EF descriptions and new spectral norm cone oracles. Chapter 3 closely follows Coey, Kapelevich, and Vielma (2021a). This thesis does not include relevant material on new sum-of-squares cones in Kapelevich, Coey, and Vielma (2021). Chapter 4 follows Coey, Lubin, and Vielma (2020), though we have updated the notation and removed much of the software architecture discussion. Chapter 5 is joint work with Juan Pablo Vielma that is not yet submitted.

Hypatia solver is introduced in Section 1.2 and is available at `https://github.com/chriscoey/Hypatia.jl`. Pajarito solver is introduced in Section 4.5 and is available at `https://github.com/JuliaOpt/Pajarito.jl`. However as we discuss in Section 5.2.1, Pajarito is now obsolete as it is based on MathProgBase, which is superseded by MathOptInterface (Benoit Legat et al., 2020). MOIPajarito is the MathOptInterface-based successor to Pajarito. MOIPajarito is introduced in Section 5.2.1 and is currently available at `https://github.com/chriscoey/MOIPajarito.jl`, but this URL and package name may change when the solver is officially released.

0.3 General notation

For a natural number d, we define the index set $[\![d]\!] := \{1, 2, \ldots, d\}$. Often we construct vectors with round parentheses, e.g. (a, b, c), and matrices with square brackets, e.g. $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$. For a set $\mathcal{C}$, $\text{cl}(\mathcal{C})$ and $\text{int}(\mathcal{C})$ denote the closure and interior of $\mathcal{C}$, respectively.

$\mathbb{R}$ denotes the space of reals, and $\mathbb{R}_{\geq}$, $\mathbb{R}_{>}$, $\mathbb{R}_{\leq}$, $\mathbb{R}_{<}$ denote the nonnegative, positive, nonpositive, and negative reals. $\mathbb{R}^d$ is the space of d-dimensional real vectors, and $\mathbb{R}^{d_1 \times d_2}$ is the d_1-by-d_2-dimensional real matrices. The vectorization operator $\text{vec} : \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}^{d_1 d_2}$ maps matrices to vectors

by stacking columns, and its inverse operator is $\mathrm{mat}_{d_1,d_2} : \mathbb{R}^{d_1 d_2} \to \mathbb{R}^{d_1 \times d_2}$.

$\mathbb{S}^d$ is the space of symmetric matrices with side dimension d, and $\mathbb{S}^d_\succeq$ and $\mathbb{S}^d_\succ$ denote the positive semidefinite (PSD) and positive definite symmetric matrices. The inequality $S \succeq Z$ is equivalent to $S - Z \in \mathbb{S}^d_\succeq$ (and similarly for the strict inequality $\succ$ and $\mathbb{S}^d_\succ$). We let $\mathrm{sd}(d) \coloneqq d(d+1)/2$ be the dimension of the vectorized upper triangle of $\mathbb{S}^d$. We overload the vectorization operator $\mathrm{vec} : \mathbb{S}^d \to \mathbb{R}^{\mathrm{sd}(d)}$ to perform an *svec* transformation, which rescales off-diagonal elements by $\sqrt{2}$ and stacks columns of the upper triangle (or equivalently, rows of the lower triangle). For example, for the *smat* space point $S \in \mathbb{S}^3$ we have $\mathrm{sd}(3) = 6$ and $\mathrm{vec}(S) = (S_{1,1}, \sqrt{2}S_{1,2}, S_{2,2}, \sqrt{2}S_{1,3}, \sqrt{2}S_{2,3}, S_{3,3}) \in \mathbb{R}^{\mathrm{sd}(3)}$ in svec space. The inverse mapping $\mathrm{mat} : \mathbb{R}^{\mathrm{sd}(d)} \to \mathbb{S}^d$ is well-defined.

For a vector or matrix A, the transpose is A' and the trace is $\mathrm{tr}(A)$. $I(d)$ is the identity matrix in $\mathbb{R}^{d \times d}$. $\mathrm{Diag} : \mathbb{R}^d \to \mathbb{S}^d$ is the diagonal matrix of a given vector, and $\mathrm{diag} : \mathbb{S}^d \to \mathbb{R}^d$ is the vector of the diagonal of a given matrix. For dimensions implied by context, e is a vector of ones, e_i is the ith unit vector, and 0 may be a vector or matrix of zeros.

We use the standard inner product on $\mathbb{R}^d$, i.e. $\langle s, z \rangle = s'z = \sum_{i \in [\![d]\!]} s_i z_i$ for $s, z \in \mathbb{R}^d$ This equips $\mathbb{R}^d$ with the standard norm $\|s\| = (s's)^{1/2}$. The linear operators vec and mat preserve inner products, e.g. $\langle S, Z \rangle = \mathrm{vec}(S)' \mathrm{vec}(Z) = \mathrm{tr}(S'Z)$ for $S, Z \in \mathbb{R}^{d_1 \times d_2}$ or $S, Z \in \mathbb{S}^d$.

$|x|$ is the absolute value of $x \in \mathbb{R}$ and $\log(x)$ is the natural logarithm of $x > 0$. For a vector $x \in \mathbb{R}^d$, $\|x\|_\infty = \max_{i \in [\![d]\!]} |x_i|$ is the ℓ_∞ norm and $\|x\|_1 = \sum_{i \in [\![d]\!]} |x_i|$ is the ℓ_1 norm. $\det(X)$ is the determinant of $X \in \mathbb{S}^d$, and $\mathrm{logdet}(X)$ is the log-determinant of $X \succ 0$. $\lambda_i(X)$ is the ith largest eigenvalue of $X \in \mathbb{S}^d$, and $\sigma_i(X)$ is the ith largest singular value for $X \in \mathbb{R}^{d_1 \times d_2}$.

Often we also work with vectors or matrices over the complex numbers $\mathbb{C}$. For $x \in \mathbb{C}$, the real part is $\Re(x)$ and the imaginary part is $\Im(x)$. We vectorize complex arrays to real vectors by storing each complex element as two consecutive real elements (the real part followed by the imaginary part). $\mathbb{H}^d$ is the complex Hermitian matrices and $\mathbb{H}^d_\succeq$ are the Hermitian PSD matrices. For $\mathbb{H}^d$, we use a modified svec transformation to the d^2-dimensional real vectors. For example, the smat space point $S \in \mathbb{H}^2$ maps to the svec space point $\mathrm{vec}(S) = (S_{1,1}, \sqrt{2}\Re(S_{1,2}), \sqrt{2}\Im(S_{1,2}), S_{2,2}) \in \mathbb{R}^4$. The complex-to-real vectorization operators preserve inner products and their inverse mappings are well-defined. A' denotes the conjugate transpose of a complex matrix A, and most of our other notation generalizes easily to the complex case.

Given a set $\mathcal{C} \subset \mathbb{R}^d$, suppose the function $f : \mathrm{int}(\mathcal{C}) \to \mathbb{R}$ is strictly convex and three times continuously differentiable on the interior of $\mathcal{C}$. For a point $p \in \mathrm{int}(\mathcal{C})$, we denote the gradient and Hessian of f at p as $\nabla f(p) \in \mathbb{R}^d$ and $\nabla^2 f(p) \in \mathbb{S}^d_\succ$. Given an $h \in \mathbb{R}^d$, the first, second, and third order directional derivatives of f at p in direction h are $\nabla f(p)[h] \in \mathbb{R}$, $\nabla^2 f(p)[h, h] \in \mathbb{R}_\geq$, and $\nabla^3 f(p)[h, h, h] \in \mathbb{R}$. Sometimes we interpret (directional) derivatives as operators, e.g. $\nabla^2 f(p)[h, h] = \langle \nabla^2 f(p)[h], h \rangle$. Often we omit the point at which the derivative is evaluated if this is clear from context, e.g. $\nabla^2 f$. We use subscripts for partial derivatives, e.g. $\nabla_w f(u, w)$.

Chapter 1

Performance enhancements for a generic conic interior point algorithm

Abstract

We define an *exotic cone* as a proper cone for which we can implement a small set of tractable (i.e. fast, numerically stable, analytic) oracles for a logarithmically homogeneous self-concordant barrier function for the cone or for its dual cone. Our extensible, open source conic interior point solver *Hypatia* allows modeling and solving any conic problem over a Cartesian product of exotic cones. In this chapter, we introduce Hypatia's interior point algorithm, which generalizes that of Skajaa and Ye (2015) by handling exotic cones without tractable primal oracles. To improve iteration count and solve time in practice, we propose four enhancements to the interior point stepping procedure of Skajaa and Ye (2015): (1) loosening the central path proximity conditions, (2) adjusting the directions using a third order directional derivative barrier oracle, (3) performing a backtracking search on a curve, and (4) combining the prediction and centering directions. We implement two dozen useful exotic cones in Hypatia. We summarize the complexity of computing oracles for these cones and show that our new third order oracle is not a bottleneck. From 37 applied examples, we generate a diverse benchmark set of 379 problems. Overall, our stepping enhancements reduce the geometric means of iteration count and solve time by over 80% and 70% respectively.

1.1 Introduction

Any convex optimization problem may be represented as a conic problem that minimizes a linear function over the intersection of an affine subspace with a Cartesian product of primitive proper cones (i.e. irreducible, closed, convex, pointed, and full-dimensional conic sets). Under certain conditions, a conic problem has a simple and easily checkable certificate of optimality, primal infeasibility, or dual infeasibility (Permenter, Friberg, and E. D. Andersen, 2017). Most commercial and open source conic solvers (such as CSDP (Borchers, 1999), CVXOPT (M. Andersen et al., 2011), ECOS (Domahidi, Chu, and Boyd, 2013; Serrano, 2015), MOSEK (MOSEK ApS, 2020a), SDPA (Yamashita, Fujisawa, and Kojima, 2003), Alfonso (Papp and Yıldız, 2017)) implement primal-dual

interior point methods (IPMs) based on the theory of logarithmically homogeneous self-concordant barrier (LHSCB) functions. Compared to first order conic methods (see O'Donoghue et al. (2016) on SCS solver), idealized IPMs typically exhibit higher per-iteration cost, but have a lower iteration complexity of $\mathcal{O}(\sqrt{\nu}\log(1/\varepsilon))$ iterations to converge to ε tolerance, where ν is the barrier parameter of the LHSCB. We limit the scope of this chapter to conic IPMs, but note that there are other notions of duality and IPMs for convex problems outside of the conic realm (see e.g. Karimi and Tunçel (2020b)).

1.1.1 The Skajaa-Ye algorithm

Most conic IPM solvers use efficient algorithms specialized for symmetric cones, in particular, the nonnegative, (rotated) second order, and positive semidefinite (PSD) cones. However, many useful nonsymmetric conic constraints (such as $u \leq \log(w)$, representable with an exponential cone) are not exactly representable with symmetric cones. Although nonsymmetric conic IPMs proposed by Nesterov, Todd, and Ye (1996) and Nesterov (2012) can handle a broader class of cones, they have several disadvantages compared to specialized symmetric methods (e.g. requiring larger linear systems, strict feasibility of the initial iterate, or conjugate LHSCB oracles).

The algorithm by Skajaa and Ye (2015), henceforth referred to as *SY*, addresses these issues by approximately tracing the central path of the homogeneous self-dual embedding (HSDE) (E. D. Andersen, Roos, and Terlaky, 2003; Xu, Hung, and Ye, 1996) to an approximate solution for the HSDE. This final iterate provides an approximate conic certificate for the conic problem, if a conic certificate exists. The SY algorithm relies on an idea by Nesterov (2012) that a high quality prediction direction (enabling a long step and rapid progress towards a solution) can be computed if the current iterate is in close proximity to the central path (i.e. it is an approximate scaling point). To restore centrality after each prediction step, SY performs a series of centering steps.

By using a different definition of central path proximity to Nesterov (2012), SY avoids needing conjugate LHSCB oracles. Some proposed techniques such as the Hessian scaling updates and central path proximity definitions of Myklebust and Tunçel (2014) and Dahl and E. D. Andersen (2021) require conjugate LHSCB oracles. Indeed, a major advantage of SY is that it only requires access to a few tractable oracles for the primal cone: an initial interior point, feasibility check, and gradient and Hessian evaluations for the LHSCB. In our experience, for a large class of proper cones, these oracles can be evaluated analytically, i.e. without requiring the implementation of iterative numerical procedures (such as optimization) that can be expensive and may need numerical tuning. Conjugate LHSCB oracles in general require optimization, and compared to the analytic oracles, they are often significantly less efficient and more prone to numerical instability.

1.1.2 Practical algorithmic developments

We describe an IPM that generalizes and enhances the performance of SY. We implement this algorithm in our new IPM solver, Hypatia. A key feature of Hypatia is a generic cone interface that allows users to define new *exotic cones*. We define an exotic cone as a proper cone $\mathcal{K}$ for which we

can implement tractable (i.e. fast, numerically stable, analytic) oracles (those required by SY) for either $\mathcal{K}$ or $\mathcal{K}^*$ (not both). We have already predefined two dozen useful exotic cone types (some with multiple variants) in Hypatia.

Defining a new cone through Hypatia's cone interface makes both the cone and its dual available for use in conic formulations. For many proper cones of interest, including most of Hypatia's nonsymmetric cones, we are not aware of tractable LHSCB oracles for both $\mathcal{K}$ and $\mathcal{K}^*$. Suppose a conic problem involves a Cartesian product of exotic cones, some with primal oracles implemented and some with dual oracles implemented (as in several example formulations described in Section 2.3). In this case, SY can solve neither the primal conic problem nor its conic dual, as SY requires primal oracles for all cones. Our algorithm generalizes SY to allow a conic formulation over any Cartesian product of exotic cones.

The focus of Skajaa and Ye (2015) is demonstrating that SY has the best known iteration complexity for conic IPMs. This complexity analysis was corrected by Papp and Yıldız (2017), who implemented SY in their recent MATLAB solver Alfonso (Papp and Yıldız, 2020; Papp and Yıldız, 2021). It is well known that performant IPM implementations tend to violate assumptions used in iteration complexity analysis, so in this chapter we are not concerned with iteration complexity. Our goal is to reduce iteration counts and solve times in practice, by enhancing the performance of the interior point stepping procedure proposed by SY and implemented by Alfonso.

The basic SY-like stepping procedure computes a prediction or centering direction by solving a structured linear system, performs a backtracking line search in the direction, and steps as far as possible given a restrictive central path proximity condition. We propose a sequence of four practical performance enhancements.

Less restrictive proximity. We use a relaxed central path proximity condition, allowing longer prediction steps and fewer centering steps.

Third order adjustments. After computing the prediction or centering direction, we compute a *third order adjustment* (TOA) direction using a new *third order oracle* (TOO) for exotic cones. We use a line search in the unadjusted direction to determine how to combine it with the TOA direction, before performing a second line search and stepping in the new adjusted direction.

Curve search. Due to the central path proximity checks, each backtracking line search can be quite expensive. Instead of performing two line searches, we use a single backtracking search along a particular quadratic curve of combinations of the unadjusted and TOA directions.

Combined directions. Unlike SY, most conic IPMs do not use separate prediction and centering phases. We compute the prediction and centering directions and their associated TOA directions, then perform a backtracking search along a quadratic curve of combinations of all four directions.

Our TOA approach is distinct from the techniques by Mehrotra (1992) and Dahl and E. D.

Andersen (2021) that also use higher order LHSCB information.[1] Unlike these techniques, we derive adjustments (using the TOO) for both the prediction and centering directions. Our TOO has a simpler and more symmetric structure than the third order term used by Dahl and E. D. Andersen (2021), and we leverage this for fast and numerically stable evaluations. Whereas the method by Mehrotra (1992) only applies to symmetric cones, and Dahl and E. D. Andersen (2021) test their technique only for the standard exponential cone, we implement and test our TOO for all of Hypatia's two dozen predefined cones. In our experience, requiring a tractable TOO is only as restrictive as requiring tractable gradient and Hessian oracles. We show that the time complexity of the TOO is no higher than that of the other required oracles for each of our cones.

Although this chapter is mainly concerned with the stepping procedures, we also outline our implementations of other key algorithmic components. These include preprocessing of problem data, finding an initial iterate, the solution of structured linear systems for search directions, and efficient backtracking searches with central path proximity checks. We note that Hypatia has a variety of algorithmic options for these components; these different options can have a dramatic impact on overall solve time and memory usage, but in most cases they have minimal effect on the iteration count. For the purposes of this chapter, we only describe and test one set of (default) options for these components.

1.1.3 Benchmark instances and computational testing

We implement and briefly describe 37 applied examples (available in Hypatia's examples folder), each of which has options for creating formulations of different types and sizes. From these examples, we generate 379 problem instances (primal-dual feasible, primal infeasible, or dual infeasible) of a wide range of sizes. Since there is currently no conic benchmark storage format that recognizes more than a handful of cone types, we generate all instances on the fly using JuMP or Hypatia's native interface. All of Hypatia's predefined cones are represented in these instances, so we believe this is the most diverse conic benchmark set available.

On this benchmark set, we run five different stepping procedures: the basic SY-like procedure (similar to Alfonso) and the sequence of four cumulative enhancements to this procedure. Our results show that each enhancement tends to improve Hypatia's iteration count and solve time, with minimal impact on the number of instances solved. We do not enforce time or iteration limits, but we note that under strict limits the enhancements would greatly improve the number of instances solved. The TOA enhancement alone leads to a particularly consistent improvement of around 45% for iteration counts. Overall, the enhancements together reduce the iterations and solve time by more than 80% and 70% respectively. For instances that take more iterations or solve time, the enhancements tend to yield greater relative improvements in these measures.

[1] To avoid confusion, we do not use the term 'corrector' in this chapter. In the terminology of Mehrotra (1992) and Dahl and E. D. Andersen (2021) our TOA approach is a type of 'higher order corrector' technique, but also our unadjusted centering direction is referred to by Skajaa and Ye (2015) and Papp and Yıldız (2017) as the 'corrector' direction.

1.1.4 Overview

In Section 1.2, we introduce Hypatia solver. In Section 1.3, we define exotic cones, LHSCBs, and our required cone oracles (including the TOO). In Section 1.4, we describe Hypatia's general primal-dual conic form, associated conic certificates, and the HSDE. In Section 1.5, we define the central path of the HSDE and central path proximity measures, and we outline Hypatia's high level algorithm. We also derive the prediction and centering directions and our new TOA directions, and we describe the SY-like stepping procedure and our series of four enhancements to this procedure. In Sections 1.6 and 1.7, we discuss advanced procedures for preprocessing and initial point finding, solving structured linear systems for directions, and performing efficient backtracking searches and proximity checks. In Section 1.8, we briefly introduce Hypatia's predefined exotic cones and show that our TOO is relatively cheap to compute. In Section 1.9, we summarize our applied examples and exotic conic benchmark instances, and finally we present our computational results demonstrating the practical efficacy of our stepping enhancements.

1.2 Hypatia solver

We introduce our new conic IPM solver, Hypatia, available under the open source MIT license at `https://github.com/chriscoey/Hypatia.jl` Hypatia is written in the Julia language (Bezanson et al., 2017) and is accessible through either a flexible, low-level native interface or the open source modeling tools JuMP (Dunning, Huchette, and Lubin, 2017) and Convex.jl (Udell et al., 2014). In Coey, Kapelevich, and Vielma (2021c), we provide basic documentation, examples, and instructions for using Hypatia.

A key feature of Hypatia is the generic cone interface, which allows defining new exotic cones. The interface requires only the implementation of the few cone oracles needed by SY, and allows optional specification of additional cone oracles that can improve performance. Unlike SY (and its implementation in Alfonso solver (Papp and Yıldız, 2020; Papp and Yıldız, 2021)), defining a new cone in Hypatia makes both the cone and its dual cone simultaneously available for use in conic formulations. For many cones of interest, easily computable oracles are only known for either the primal cone or the dual cone but not both. This means Hypatia is able to handle a broader class of conic formulations than SY/Alfonso, which require oracles specifically for all cones in the primal conic formulation. For example, in our portfolio rebalancing example in Section 2.3.1, we have both ℓ_1 norm cone and ℓ_∞ norm cone constraints; we are aware of an LHSCB with simple closed form oracles for the ℓ_∞ norm cone, but not for its dual cone - the ℓ_1 norm cone (see Section 2.2.2.1).

We have already implemented two dozen predefined exotic cone types in Hypatia (not counting their dual cones). Many of these cones have multiple variants, for example both real and complex flavors. We describe these cones, their dual cones, and associated LHSCBs in Coey, Kapelevich, and Vielma (2021b) and Coey, Kapelevich, and Vielma (2021c). Several cones are new and required the development of LHSCBs and efficient procedures for oracle evaluations (see Chapters 2 and 3). We note that for many of these cones, computing conjugate barrier oracles requires optimization, which

can be slow and numerically fraught. Fortunately, like SY, Hypatia does not need conjugate barrier oracles.

Hypatia uses a primal-dual conic form that (unlike SY/Alfonso) does not force the user to introduce slack variables, and allows linear operators to be represented with Julia's sparse, dense, or structured abstract matrix types. This conic form matches CVXOPT's *cone LP* form (M. Andersen et al., 2011), and we describe the associated conic certificates in Section 1.4. Hypatia allows representing and solving conic problems in any real floating point type in Julia. For example, since Julia wraps the GNU MPFR Library, using Julia's *BigFloat* type allows solving conic problems to arbitrary precision (though this tends to be less efficient than using the default *Float64* type).

Hypatia's solver interface is highly extensible. We provide several optional interior point search and stepping procedures, including those we develop in Section 1.5. Since the per-iteration bottleneck of IPMs such as Hypatia's algorithm tends to be solving the large structured linear system for search directions, Hypatia allows the user to choose from several predefined methods (including options for sparse or dense factorization-based solves or linear-operator-based iterative/indirect solves) or to implement their own formulation-specific procedure to leverage problem structure.

In Hypatia's examples folder, we have included over three dozen applied example problems modeled using JuMP or Hypatia's native interface. Most examples have multiple formulation type and size options. We make use of various modeling utilities Hypatia provides for manipulating real and complex arrays and for generating well-conditioned polynomial interpolations (useful in polynomial sum of squares formulations). We briefly summarize the examples in Section 1.9.1.

1.3 Exotic cones and oracles

Let $\mathcal{K}$ be a proper cone in $\mathbb{R}^q$, i.e. a conic subset of $\mathbb{R}^q$ that is closed, convex, pointed, and full-dimensional (see Skajaa and Ye (2015)). Note that requiring $\mathcal{K}$ to be a subset of $\mathbb{R}^q$ simplifies our notation but is not restrictive, e.g. for the PSD cone, we use the standard svec vectorization (see Section 0.3). The dual cone of $\mathcal{K}$ is $\mathcal{K}^*$, which is also a proper cone in $\mathbb{R}^q$:

$$\mathcal{K}^* := \{z \in \mathbb{R}^q : s'z \geq 0, \forall s \in \mathcal{K}\}. \tag{1.1}$$

Following Nesterov and Nemirovski (1994, Sections 2.3.1 and 2.3.3), $f : \text{int}(\mathcal{K}) \to \mathbb{R}$ is a ν-LHSCB for $\mathcal{K}$, where $\nu \geq 1$ is the *LHSCB parameter*, if it is three times continuously differentiable, strictly convex, satisfies $f(s_i) \to \infty$ along every sequence $s_i \in \text{int}(\mathcal{K})$ converging to the boundary of $\mathcal{K}$, and:

$$\left|\nabla^3 f(s)[h,h,h]\right| \leq 2\left(\nabla^2 f(s)[h,h]\right)^{3/2} \qquad \forall s \in \text{int}(\mathcal{K}), h \in \mathbb{R}^q, \tag{1.2a}$$

$$f(\theta s) = f(s) - \nu \log(\theta) \qquad \forall s \in \text{int}(\mathcal{K}), \theta \in \mathbb{R}_>. \tag{1.2b}$$

Following Renegar (2001, Section 3.3), we define the *conjugate* of f, $f^* : \text{int}(\mathcal{K}^*) \to \mathbb{R}$, as:

$$f^*(z) := -\inf_{s \in \text{int}(\mathcal{K})}\{s'z + f(s)\}, \tag{1.3}$$

which is a ν-LHSCB for $\mathcal{K}^*$.

A Cartesian product $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K$ of K proper cones is a proper cone, and its dual cone is $\mathcal{K}^* = \mathcal{K}_1^* \times \cdots \times \mathcal{K}_K^*$. In this case, if f_k is a ν_k-LHSCB for $\mathcal{K}_k$, then $\sum_{k\in[\![K]\!]} f_k$ is an LHSCB for $\mathcal{K}$ with parameter $\sum_{k\in[\![K]\!]}\nu_k$ (Nesterov and Nemirovski, 1994, Proposition 2.3.3). We call $\mathcal{K}$ a primitive cone if it cannot be written as a Cartesian product of two or more lower-dimensional cones (i.e. K must equal one). Note $\mathcal{K}^*$ is primitive if and only if $\mathcal{K}$ is primitive. Primitive proper cones are the fundamental building blocks of conic formulations.

We call a proper cone $\mathcal{K}$ an exotic cone if we can implement a particular set of tractable oracles for either $\mathcal{K}$ or $\mathcal{K}^*$. Suppose we have tractable oracles for $\mathcal{K} \subset \mathbb{R}^q$ and let $f : \operatorname{int}(\mathcal{K}) \to \mathbb{R}$ denote the ν-LHSCB for $\mathcal{K}$. The oracles for $\mathcal{K}$ that we require in this chapter are as follows.

Feasibility check. The strict feasibility oracle checks whether a given point $s \in \mathbb{R}^q$ satisfies $s \in \operatorname{int}(\mathcal{K})$.

Gradient and Hessian evaluations. Given a point $s \in \operatorname{int}(\mathcal{K})$, the gradient oracle g and Hessian oracle H evaluated at s are:

$$g(s) \coloneqq \nabla f(s) \in \mathbb{R}^q, \tag{1.4a}$$

$$H(s) \coloneqq \nabla^2 f(s) \in \mathbb{S}^q_{\succ}. \tag{1.4b}$$

Third order directional derivative. Given a point $s \in \operatorname{int}(\mathcal{K})$ and a direction $\delta_s \in \mathbb{R}^q$, our new *third order oracle* (TOO), denoted T, is a rescaled third order directional derivative vector:

$$\mathrm{T}(s, \delta_s) \coloneqq -\tfrac{1}{2}\nabla^3 f(s)[\delta_s, \delta_s] \in \mathbb{R}^q. \tag{1.5}$$

Initial interior point. The initial interior point $t \in \operatorname{int}(\mathcal{K})$ is an arbitrary point in the interior of $\mathcal{K}$ (which is nonempty since $\mathcal{K}$ is proper).

In Section 1.8, we introduce Hypatia's predefined cones and discuss the time complexity of computing the feasibility check, gradient, Hessian, and TOO oracles. In Section 2.5, we describe efficient and numerically stable techniques for computing these oracles for a handful of our cones. Although Hypatia's generic cone interface allows specifying additional oracles that can improve speed and numerical performance (e.g. a dual cone feasibility check, Hessian product, and inverse Hessian product), these optional oracles are outside the scope of this chapter.

For the initial interior point (which Hypatia only calls once, when finding an initial iterate), we prefer to use the *central point* of $\mathcal{K}$. This is the unique point satisfying $t \in \operatorname{int}(\mathcal{K}) \cap \operatorname{int}(\mathcal{K}^*)$ and $t = -g(t)$ (Dahl and E. D. Andersen, 2021). It can also be characterized as the solution to the following strictly convex problem:

$$\arg\min_{s\in\operatorname{int}(\mathcal{K})} \left(f(s) + \tfrac{1}{2}\|s\|^2\right). \tag{1.6}$$

For the nonnegative cone $\mathcal{K} = \mathbb{R}^n_{\geq}$, $f(s) = -\log(s)$ is an LHSCB with $\nu = 1$, and we have $g(s) = -s^{-1}$

and the central point $t = 1 = -g(1)$. For some of Hypatia's cones, we are not aware of a simple analytic expression for the central point, in which case we typically use a non-central interior point.

1.4 Conic form and certificates

In Sections 1.4.1 and 1.4.2, we describe our general conic primal-dual form and the associated conic certificates. In Section 1.4.3, we introduce the homogeneous self-dual embedding (HSDE) conic feasibility problem, a solution of which may provide a conic certificate.

1.4.1 General conic form

Hypatia uses the following primal conic form over variable $x \in \mathbb{R}^n$:

$$\inf_x \quad c'x : \tag{1.7a}$$
$$b - Ax = 0, \tag{1.7b}$$
$$h - Gx \in \mathcal{K}, \tag{1.7c}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^p$, and $h \in \mathbb{R}^q$ are vectors, $A : \mathbb{R}^n \to \mathbb{R}^p$ and $G : \mathbb{R}^n \to \mathbb{R}^q$ are linear maps, and $\mathcal{K} \subset \mathbb{R}^q$ is a Cartesian product $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K$ of exotic cones. For $k \in [\![K]\!]$, we let $q_k = \dim(\mathcal{K}_k)$, so $\sum_{k \in [\![K]\!]} q_k = q = \dim(\mathcal{K})$. Henceforth we use n, p, q to denote respectively the variable, equality, and conic constraint dimensions of a conic problem.

Once a proper cone $\mathcal{K}_k$ is defined through Hypatia's generic cone interface, both $\mathcal{K}_k$ and $\mathcal{K}_k^*$ may be used in any combination with other cones recognized by Hypatia to construct the Cartesian product cone $\mathcal{K}$ in (1.7c). The primal form (1.7) matches CVXOPT's form, however CVXOPT only recognizes symmetric cones (Vandenberghe, 2010). Unlike the conic form used by Skajaa and Ye (2015) and Papp and Yıldız (2021), which recognizes conic constraints of the form $x \in \mathcal{K}$, our form does not require introducing slack variables to represent a more general constraint $h - Gx \in \mathcal{K}$.

The conic dual problem of (1.7), over variables $y \in \mathbb{R}^p$ and $z \in \mathbb{R}^q$ associated with (1.7b) and (1.7c), is:

$$\sup_{y,z} \quad -b'y - h'z : \tag{1.8a}$$
$$c + A'y + G'z = 0, \tag{1.8b}$$
$$z \in \mathcal{K}^*, \tag{1.8c}$$

where (1.8b) is associated with the primal variable $x \in \mathbb{R}^n$.

1.4.2 Conic certificates

Under certain conditions, there exists a simple conic certificate providing an easily verifiable proof of infeasibility of the primal (1.7) or dual (1.8) problem (via the conic generalization of Farkas' lemma) or optimality of a given primal-dual solution.

A primal improving ray x is a feasible direction for the primal along which the objective improves:

$$c'x < 0, \tag{1.9a}$$
$$-Ax = 0, \tag{1.9b}$$
$$-Gx \in \mathcal{K}, \tag{1.9c}$$

and hence it certifies dual infeasibility.

A dual improving ray (y, z) is a feasible direction for the dual along which the objective improves:

$$-b'y - h'z > 0, \tag{1.10a}$$
$$A'y + G'z = 0, \tag{1.10b}$$
$$z \in \mathcal{K}^*, \tag{1.10c}$$

and hence it certifies primal infeasibility.

A complementary solution (x, y, z) satisfies the primal-dual feasibility conditions (1.7b), (1.7c), (1.8b) and (1.8c), and has equal and attained primal and dual objective values:

$$c'x = -b'y - h'z, \tag{1.11}$$

and hence certifies optimality of (x, y, z) via conic weak duality.

One of these certificates exists if neither the primal nor the dual is *ill-posed*. Intuitively, according to MOSEK ApS (2020a, Section 7.2), a conic problem is ill-posed if a small perturbation of the problem data can change the feasibility status of the problem or cause arbitrarily large perturbations to the optimal solution (see Permenter, Friberg, and E. D. Andersen (2017) for more details).

1.4.3 Homogeneous self-dual embedding

The HSDE is a self-dual conic feasibility problem in variables $x \in \mathbb{R}^n, y \in \mathbb{R}^p, z \in \mathbb{R}^q, \tau \in \mathbb{R}, s \in \mathbb{R}^q, \kappa \in \mathbb{R}$ (see Vandenberghe (2010, Section 6)), derived from a homogenization of the primal-dual optimality conditions (1.7b), (1.7c), (1.8b), (1.8c) and (1.11):

$$\begin{bmatrix} 0 \\ 0 \\ s \\ \kappa \end{bmatrix} = \begin{bmatrix} 0 & A' & G' & c \\ -A & 0 & 0 & b \\ -G & 0 & 0 & h \\ -c' & -b' & -h' & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \tau \end{bmatrix}, \tag{1.12a}$$
$$(z, \tau, s, \kappa) \in \left(\mathcal{K}^* \times \mathbb{R}_{\geq} \times \mathcal{K} \times \mathbb{R}_{\geq}\right). \tag{1.12b}$$

For convenience we let $\omega := (x, y, z, \tau, s, \kappa) \in \mathbb{R}^{n+p+2q+2}$ represent a point. We define the structured 4×6 block matrix $E \in \mathbb{R}^{(n+p+q+1)\times \dim(\omega)}$ such that (1.12a) is equivalent to:

$$E\omega = 0. \tag{1.13}$$

Here we assume E has full row rank; in Section 1.6 we discuss preprocessing techniques that handle linearly dependent rows. Note that $\omega = 0$ satisfies (1.12), so the HSDE is always feasible. A point ω is an interior point if it is strictly feasible for the conic constraints (1.12b), i.e. ω satisfies $(z, \tau, s, \kappa) \in \operatorname{int}\big(\mathcal{K}^* \times \mathbb{R}_{\geq} \times \mathcal{K} \times \mathbb{R}_{\geq}\big)$.

Suppose a point ω is feasible for the HSDE (1.12). From skew symmetry of the square 4×4 block matrix in (1.12a), we have $s'z + \kappa\tau = 0$. From the conic constraints (1.12b) and the dual cone inequality (1.1) we have $s'z \geq 0$ and $\kappa\tau \geq 0$. Hence $s'z = \kappa\tau = 0$. We consider an exhaustive list of cases below.

Optimality. If $\tau > 0, \kappa = 0$, then $(x, y, z)/\tau$ is a complementary solution satisfying the primal-dual optimality conditions (1.7b), (1.7c), (1.8b), (1.8c) and (1.11).

Infeasibility. If $\tau = 0, \kappa > 0$, then $c'x + b'y + h'z < 0$ and we consider two sub-cases.

Of primal. If $b'y + h'z < 0$, then (y, z) is a primal infeasibility certificate satisfying (1.10).

Of dual. If $c'x < 0$, then x is a dual infeasibility certificate satisfying (1.9).

No information. If $\tau = \kappa = 0$, then ω provides no information about the feasibility or optimal values of the primal or dual.

Thus an HSDE solution ω satisfying $\kappa + \tau > 0$ provides an optimality or infeasibility certificate (see Skajaa and Ye (2015, Lemma 1) and Vandenberghe (2010, Section 6.1)).

According to Skajaa and Ye (2015, Section 2), if the primal and dual problems are both feasible and have zero duality gap, SY (their algorithm) finds an HSDE solution with $\tau > 0$ (yielding a complementary solution), and if the primal or dual (possibly both) is infeasible, SY finds an HSDE solution with $\kappa > 0$ (yielding an infeasibility certificate). This implies that if SY finds a solution with $\kappa = \tau = 0$, then $\kappa = \tau = 0$ for all solutions to the HSDE; in this case, no complementary solution or improving ray exists, and the primal or dual (possibly both) is ill-posed (Permenter, Friberg, and E. D. Andersen, 2017). The algorithm we describe in Section 1.5 is an extension of SY that inherits these properties.

1.5 Central path following algorithm

In Section 1.5.1, we describe the central path of the HSDE, and in Section 1.5.2 we define central path proximity measures. In Section 1.5.3, we outline a high level IPM that maintains iterates close to the central path, and we give numerical convergence criteria for detecting approximate conic certificates. In Section 1.5.4, we derive prediction and centering directions and our corresponding

TOA directions using the TOO. Finally in Section 1.5.5, we summarize an SY-like stepping procedure and describe our sequence of four enhancements to this procedure.

1.5.1 Central path of the homogeneous self-dual embedding

We define the HSDE in (1.12). Recall that $\mathcal{K}$ in our primal conic form (1.7) is a Cartesian product $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K$ of K exotic cones. We partition the exotic cone indices $[\![K]\!]$ into two sets: K_{pr} for cones with primal oracles (i.e. for $\mathcal{K}_k$) and K_{du} for cones with dual oracles (i.e. for $\mathcal{K}_k^*$). For convenience, we append the τ and κ variables onto the s and z variables. Letting $\bar{K} = K + 1$, we define for $k \in [\![\bar{K}]\!]$:

$$\bar{\mathcal{K}}_k \coloneqq \begin{cases} \mathcal{K}_k & k \in K_{\text{pr}}, \\ \mathcal{K}_k^* & k \in K_{\text{du}}, \\ \mathbb{R}_\geq & k = \bar{K}, \end{cases} \tag{1.14a}$$

$$(\bar{z}_k, \bar{s}_k) \coloneqq \begin{cases} (z_k, s_k) & k \in K_{\text{pr}}, \\ (s_k, z_k) & k \in K_{\text{du}}, \\ (\kappa, \tau) & k = \bar{K}. \end{cases} \tag{1.14b}$$

For a given initial interior point $\omega^0 = (x^0, y^0, z^0, \tau^0, s^0, \kappa^0)$, the central path of the HSDE is the trajectory of solutions $\omega_\mu = (x_\mu, y_\mu, z_\mu, \tau_\mu, s_\mu, \kappa_\mu)$, parameterized by $\mu > 0$, satisfying:

$$E\omega_\mu = \mu E\omega^0, \tag{1.15a}$$

$$\bar{z}_{\mu,k} + \mu g_k(\bar{s}_{\mu,k}) = 0 \qquad \forall k \in [\![\bar{K}]\!], \tag{1.15b}$$

$$(\bar{z}_\mu, \bar{s}_\mu) \in \text{int}(\bar{\mathcal{K}}^* \times \bar{\mathcal{K}}). \tag{1.15c}$$

When all exotic cones have primal oracles (i.e. K_{du} is empty), our definition (1.15) exactly matches the central path defined in Vandenberghe (2010, Equation 32), and only differs from the definition in Skajaa and Ye (2015, Equations 7-8) in the affine form (i.e. the variable names and affine constraint structure). Unlike SY, our central path condition (1.15b) allows cones with dual oracles (K_{du} may be nonempty).

To obtain an initial point ω^0, we first let:

$$\left(\bar{z}_k^0, \bar{s}_k^0\right) = (-g_k(t_k), t_k) \qquad \forall k \in [\![\bar{K}]\!], \tag{1.16}$$

where $t_k \in \text{int}\left(\bar{\mathcal{K}}_k\right)$ is the initial interior point oracle (note that $\tau^0 = \kappa^0 = 1$). Although x^0 and y^0 can be chosen arbitrarily, we let x^0 be the solution of:

$$\min_{x \in \mathbb{R}^n} \quad \|x\| : \tag{1.17a}$$

$$-Ax + b\tau^0 = 0, \tag{1.17b}$$

$$-Gx + h\tau^0 - s^0 = 0, \tag{1.17c}$$

and we let y^0 be the solution of:

$$\min_{y \in \mathbb{R}^p} \quad \|y\| : \tag{1.18a}$$

$$A'y + G'z^0 + c\tau^0 = 0. \tag{1.18b}$$

In Section 1.6, we outline a QR-factorization-based procedure for preprocessing the affine data of the conic model and solving for ω^0.

Like Skajaa and Ye (2015, Section 4.1), we define the *complementarity gap* function:

$$\mu(\omega) := \bar{s}'\bar{z} / \textstyle\sum_{k \in [\![\bar{K}]\!]} \nu_k, \tag{1.19}$$

where ν_k is the LHSCB parameter of the LHSCB f_k for $\bar{\mathcal{K}}_k$ (see (1.2b)). Note that $\mu(\omega) > 0$ if $(\bar{z}, \bar{s}) \in \operatorname{int}(\bar{\mathcal{K}}^*) \times \operatorname{int}(\bar{\mathcal{K}})$, by a strict version of the dual cone inequality (1.1). From (1.16), $\mu(\omega^0) = 1$, since in (1.19) we have $(\bar{s}^0)'\bar{z}^0 = \sum_{k \in [\![\bar{K}]\!]} t_k'(-g_k(t_k))$, and $t_k'(-g_k(t_k)) = \nu_k$ by logarithmic homogeneity of f_k (Nesterov and Nemirovski, 1994, Proposition 2.3.4). Hence ω^0 satisfies the central path conditions (1.15) for parameter value $\mu = 1$. The central path is therefore a trajectory that starts at ω^0 with complementarity gap $\mu = 1$ and approaches a solution for the HSDE as μ decreases to zero.

1.5.2 Central path proximity

Given a point ω, we define the central path *proximity* π_k for exotic cone $k \in [\![\bar{K}]\!]$ as:

$$\pi_k(\omega) := \begin{cases} \left\| (H_k(\bar{s}_k))^{-1/2} (\bar{z}_k/\mu(\omega) + g_k(\bar{s}_k)) \right\| & \text{if } \mu(\omega) > 0, \bar{s}_k \in \operatorname{int}\big(\bar{\mathcal{K}}_k\big), \\ \infty & \text{otherwise.} \end{cases} \tag{1.20}$$

Hence π_k is a measure of the distance from $\bar{s}_k$ and $\bar{z}_k$ to the surface defined by the central path condition (1.15b) (compare to Skajaa and Ye (2015, Equation 9) and Nesterov and Todd (1998, Section 4)).

In Lemma 1.5.1, we show that for exotic cone $k \in [\![\bar{K}]\!]$, if $\pi_k(\omega) < 1$, then $\bar{s}_k \in \operatorname{int}\big(\bar{\mathcal{K}}_k\big)$ and $\bar{z}_k \in \operatorname{int}\big(\bar{\mathcal{K}}_k^*\big)$. This condition is sufficient but not necessary for strict cone feasibility. If it holds for all $k \in [\![\bar{K}]\!]$, then ω is an interior point (by definition) and (1.15c) is satisfied. From (1.20), $\pi_k(\omega)$ can be computed by evaluating the feasibility check, gradient, and Hessian oracles for $\bar{\mathcal{K}}_k$ at $\bar{s}_k$.

Lemma 1.5.1. Given ω, for each $k \in [\![\bar{K}]\!]$, $\pi_k(\omega) < 1$ implies $\bar{s}_k \in \operatorname{int}\big(\bar{\mathcal{K}}_k\big)$ and $\bar{z}_k \in \operatorname{int}\big(\bar{\mathcal{K}}_k^*\big)$.

Proof. We adapt Papp and Yıldız (2017, Lemma 15). Fix $\mu = \mu(\omega)$ for convenience, and suppose $\pi_k(\omega) < 1$ for exotic cone $k \in [\![\bar{K}]\!]$. Then by (1.20), $\mu > 0$ and $\bar{s}_k \in \operatorname{int}\big(\bar{\mathcal{K}}_k\big)$. By Papp and Yıldız (2017, Theorem 8), $\bar{s}_k \in \operatorname{int}\big(\bar{\mathcal{K}}_k\big)$ implies $-g_k(\bar{s}_k) \in \operatorname{int}\big(\bar{\mathcal{K}}_k^*\big)$. Let f_k be the LHSCB for $\bar{\mathcal{K}}_k$, and let $H_k^* := \nabla^2 f_k^*$ denote the Hessian operator for the conjugate f_k^* (see (1.3)) of f_k. By Papp and Yıldız (2017, Equation 13), $H_k^*(-g_k(\bar{s}_k)) = (H_k(\bar{s}_k))^{-1}$, so:

$$\left\| (H_k^*(-g_k(\bar{s}_k)))^{1/2} (\bar{z}_k/\mu + g_k(\bar{s}_k)) \right\| \tag{1.21a}$$

$$= \left\|(H_k(\bar{s}_k))^{-1/2}(\bar{z}_k/\mu + g_k(\bar{s}_k))\right\| \tag{1.21b}$$
$$= \pi_k(\omega) < 1. \tag{1.21c}$$

So by Papp and Yıldız (2017, Definition 1), $\bar{z}_k/\mu \in \operatorname{int}\left(\bar{\mathcal{K}}_k^*\right)$, hence $\bar{z}_k \in \operatorname{int}\left(\bar{\mathcal{K}}_k^*\right)$. □

We now define a proximity function that aggregates the exotic cone central path proximity values $\pi_k(\omega) \geq 0, \forall k \in [\![\bar{K}]\!]$. SY aggregates by taking the ℓ_2 norm:

$$\pi_{\ell_2}(\omega) \coloneqq \left\|(\pi_k(\omega))_{k\in[\![\bar{K}]\!]}\right\|. \tag{1.22}$$

An alternative aggregated proximity uses the ℓ_∞ norm (maximum):

$$\pi_{\ell_\infty}(\omega) \coloneqq \left\|(\pi_k(\omega))_{k\in[\![\bar{K}]\!]}\right\|_\infty. \tag{1.23}$$

Clearly, $0 \leq \pi_k(\omega) \leq \pi_{\ell_\infty}(\omega) \leq \pi_{\ell_2}(\omega), \forall k \in [\![\bar{K}]\!]$. Both conditions $\pi_{\ell_2}(\omega) < 1$ and $\pi_{\ell_\infty}(\omega) < 1$ guarantee by Lemma 1.5.1 that ω is an interior point, however using π_{ℓ_2} leads to a more restrictive condition on ω.

1.5.3 High level algorithm

We describe a high level algorithm for approximately solving the HSDE. The method starts at the initial interior point ω^0 with complementarity gap $\mu(\omega^0) = 1$ and approximately tracks the central path trajectory (1.15) through a series of iterations. It maintains feasibility for the linear equality conditions (1.15a) and strict cone feasibility conditions (1.15c), but allows violation of the nonlinear equality conditions (1.15b). On the ith iteration, the current interior point is ω^{i-1} satisfying $\pi_k(\omega^{i-1}) < 1, \forall k \in [\![\bar{K}]\!]$, and the complementarity gap is $\mu(\omega^{i-1})$. The method searches for a new point ω^i that maintains the proximity condition $\pi_k(\omega^i) < 1, \forall k \in [\![\bar{K}]\!]$ (and hence is an interior point) and either has a smaller complementarity gap $\mu(\omega^i) < \mu(\omega^{i-1})$ or a smaller aggregate proximity value $\pi(\omega^i) < \pi(\omega^{i-1})$ (where π is π_{ℓ_2} or π_{ℓ_∞}), or both. As the complementarity gap decreases towards zero, the RHS of (1.15a) approaches the origin, so the iterates approach a solution of the HSDE (1.12).

To detect an approximate conic certificate and terminate the iterations, we check whether the current iterate ω satisfies any of the following numerical convergence criteria. These conditions use positive tolerance values for feasibility ε_f, infeasibility ε_i, absolute gap ε_a, relative gap ε_r, and ill-posedness ε_p. The criteria and default tolerance values are similar to those described by MOSEK in MOSEK ApS (2022, Section 13.3.2) and CVXOPT in M.S. Andersen, J. Dahl, L. Vandenberghe (2021), and implemented in Alfonso (Papp and Yıldız, 2020). In Section 1.9.2, we describe the tolerance values we use for computational testing in this chapter.

Optimality. We terminate with a complementary solution $(x, y, z)/\tau$ approximately satisfying the primal-dual optimality conditions (1.7b), (1.7c), (1.8b), (1.8c) and (1.11) if:

$$\max\left(\frac{\|A'y + G'z + c\tau\|_\infty}{1 + \|c\|_\infty}, \frac{\|-Ax + b\tau\|_\infty}{1 + \|b\|_\infty}, \frac{\|-Gx + h\tau - s\|_\infty}{1 + \|h\|_\infty}\right) \leq \varepsilon_f\tau, \tag{1.24a}$$

and at least one of the following two conditions holds:

$$s'z \leq \varepsilon_a, \tag{1.24b}$$

$$\min(s'z/\tau, |c'x + b'y + h'z|) \leq \varepsilon_r \max(\tau, \min(|c'x|, |b'y + h'z|)). \tag{1.24c}$$

Note that (1.24b) and (1.24c) are absolute and relative optimality gap conditions respectively.

Primal infeasibility. We terminate with a dual improving ray (y, z) approximately satisfying (1.10) if:

$$b'y + h'z < 0, \qquad \|A'y + G'z\|_\infty \leq -\varepsilon_i(b'y + h'z). \tag{1.25}$$

Dual infeasibility. We terminate with a primal improving ray x approximately satisfying (1.9) if:

$$c'x < 0, \qquad \max(\|Ax\|_\infty, \|Gx + s\|_\infty) \leq -\varepsilon_i c'x. \tag{1.26}$$

Ill-posed primal or dual. If τ and κ are approximately 0, the primal and dual problem statuses cannot be determined (see Section 1.4.3). We terminate with an ill-posed status if:

$$\mu(\omega) \leq \varepsilon_p, \qquad \tau \leq \varepsilon_p \min(1, \kappa). \tag{1.27}$$

The high level path following algorithm below computes an approximate solution to the HSDE. In Section 1.5.5, we describe specific stepping procedures for Line 5.

```
1 Procedure SolveHSDE():
2     compute initial interior point ω^0
3     i ← 1
4     while ω^{i−1} does not satisfy any of the convergence conditions (1.24) to (1.27) do
5         ω^i ← Step(ω^{i−1})
6         i ← i + 1
7     return ω^i
```

1.5.4 Search directions

At a given iteration of the path following method, let ω be the current interior point and fix $\mu = \mu(\omega)$ for convenience. The stepping procedures we describe in Section 1.5.5 first compute one or more search directions, which depend on ω. We derive the *centering* direction in Section 1.5.4.1 and the *prediction* direction in Section 1.5.4.2. The goal of centering is to step to a point with a smaller aggregate central path proximity than the current point, i.e. to step towards the central path. The goal of prediction is to step to a point with a smaller complementarity gap, i.e. to step closer to a solution of the HSDE. The centering and prediction directions match those used by SY. We associate with each of these directions a new *third order adjustment* (TOA) direction, which depends on the

TOO and helps to correct the corresponding unadjusted direction (which must be computed before the TOA direction). Hence we derive four types of directions here.

Each direction is computed as the solution to a linear system with a structured square block matrix left hand side (LHS) and a particular right hand side (RHS) vector. The LHS, which depends only on ω and the problem data, is the same for all four directions at a given iteration. We let $r \coloneqq (r_E, r_1, \ldots, r_{\bar{K}}) \in \mathbb{R}^{\dim(\omega)}$ represent an RHS, where $r_E \in \mathbb{R}^{n+p+q+1}$ corresponds to the linear equalities (1.15a) and $r_k \in \mathbb{R}^{q_k}, \forall k \in [\![\bar{K}]\!]$ corresponds to the nonlinear equalities (1.15b). The direction $\delta \coloneqq (\delta_x, \delta_y, \delta_z, \delta_\tau, \delta_s, \delta_\kappa) \in \mathbb{R}^{\dim(\omega)}$ corresponding to r is the solution to:

$$E\delta = r_E, \tag{1.28a}$$

$$\delta_{\bar{z},k} + \mu H_k(\bar{s}_k)\delta_{\bar{s},k} = r_k \qquad \forall k \in [\![\bar{K}]\!]. \tag{1.28b}$$

Since E is assumed to have full row rank and each H_k is positive definite, this square system is nonsingular and hence has a unique solution. In Section 1.6, we describe a particular method for solving (1.28).

1.5.4.1 Centering

The centering direction δ^c is analogous to the definition of Skajaa and Ye (2015, Section 3.2). It reduces the violation on the central path nonlinear equality condition (1.15b) (and can be interpreted as a Newton step), while keeping the complementarity gap μ (approximately) constant. We denote the centering TOA direction δ^{ct}. To maintain feasibility for the linear equality condition (1.15a), we ensure $E\delta^c = E\delta^{ct} = 0$ in (1.28a).

Dropping the index $k \in [\![\bar{K}]\!]$ for conciseness, recall that (1.15b) expresses $\bar{z} + \mu g(\bar{s}) = 0$. A first order approximation of this condition gives:

$$\bar{z} + \delta_{\bar{z}} + \mu(g(\bar{s}) + H(\bar{s})\delta_{\bar{s}}) = 0 \tag{1.29a}$$

$$\Rightarrow \quad \delta_{\bar{z}} + \mu H(\bar{s})\delta_{\bar{s}} = -\bar{z} - \mu g(\bar{s}), \tag{1.29b}$$

which matches the form of (1.28b). Hence we let the centering direction δ^c be the solution to:

$$E\delta = 0, \tag{1.30a}$$

$$\delta_{\bar{z},k} + \mu H_k(\bar{s}_k)\delta_{\bar{s},k} = -\bar{z}_k - \mu g_k(\bar{s}_k) \qquad \forall k \in [\![\bar{K}]\!]. \tag{1.30b}$$

Similarly, a second order approximation of $\bar{z} + \mu g(\bar{s}) = 0$ gives:

$$\bar{z} + \delta_{\bar{z}} + \mu\big(g(\bar{s}) + H(\bar{s})\delta_{\bar{s}} + \tfrac{1}{2}\nabla^3 f(\bar{s})[\delta_{\bar{s}}, \delta_{\bar{s}}]\big) = 0 \tag{1.31a}$$

$$\Rightarrow \quad \delta_{\bar{z}} + \mu H(\bar{s})\delta_{\bar{s}} = -\bar{z} - \mu g(\bar{s}) + \mu \mathrm{T}(\bar{s}, \delta_{\bar{s}}), \tag{1.31b}$$

where (1.31b) uses the definition of the TOO in (1.5). Note that the RHSs of (1.29b) and (1.31b) differ only by $\mu\mathrm{T}(\bar{s}, \delta_{\bar{s}})$, which depends on $\delta_{\bar{s}}$. To remove this dependency, we substitute the centering direction δ^c, which we assume is already computed, into the RHS of (1.31b). Hence we let the

centering TOA direction δ^{ct}, which adjusts the centering direction, be the solution to:

$$E\delta = 0, \tag{1.32a}$$

$$\delta_{\bar{z},k} + \mu H_k(\bar{s}_k)\delta_{\bar{s},k} = \mu \mathrm{T}_k\big(\bar{s}_k, \delta^c_{\bar{s},k}\big) \qquad \forall k \in [\![\bar{K}]\!]. \tag{1.32b}$$

We note that for a rescaling factor $\alpha \in (0,1)$, the TOA direction corresponding to $\alpha\delta^c$ (a rescaling of the centering direction) is $\alpha^2\delta^{ct}$ (a rescaling of the centering TOA direction).

1.5.4.2 Prediction

The prediction direction δ^p reduces the complementarity gap and is analogous to the definition of Skajaa and Ye (2015, Section 3.1). We derive δ^p and its corresponding TOA direction δ^{pt} by considering the central path conditions (1.15) as a dynamical system parametrized by $\mu > 0$, and differentiating the linear and nonlinear equalities (1.15a) and (1.15b).

Differentiating (1.15a) once gives:

$$E\dot{\omega}_\mu = E\omega^0. \tag{1.33}$$

Rescaling (1.33) by $-\mu$ and substituting (1.15a) gives:

$$E(-\mu\dot{\omega}_\mu) = -\mu E\omega^0 = -E\omega_\mu. \tag{1.34}$$

Dropping the index $k \in [\![\bar{K}]\!]$ for conciseness, we differentiate $\bar{z}_\mu + \mu g(\bar{s}_\mu) = 0$ from (1.15b) once to get:

$$\dot{\bar{z}}_\mu + g(\bar{s}_\mu) + \mu H(\bar{s}_\mu)\dot{\bar{s}}_\mu = 0. \tag{1.35}$$

Rescaling (1.35) by $-\mu$ and substituting $\bar{z}_\mu = -\mu g(\bar{s}_\mu)$ from (1.15b) gives:

$$-\mu\dot{\bar{z}}_\mu + \mu H(\bar{s}_\mu)(-\mu\dot{\bar{s}}_\mu) = -\bar{z}_\mu. \tag{1.36}$$

The direction $\dot{\omega}_\mu$ is tangent to the central path. Like SY, we interpret the prediction direction as $\delta^p = -\mu\dot{\omega}_\mu$, so (1.34) and (1.36) become:

$$E\delta^p = -E\omega_\mu, \tag{1.37a}$$

$$\delta^p_{\bar{z}} + \mu H(\bar{s}_\mu)\delta^p_{\bar{s}} = -\bar{z}_\mu, \tag{1.37b}$$

which matches the form (1.28). So we let δ^p be the solution to:

$$E\delta = -E\omega, \tag{1.38a}$$

$$\delta_{\bar{z},k} + \mu H_k(\bar{s}_k)\delta_{\bar{s},k} = -\bar{z}_k \qquad \forall k \in [\![\bar{K}]\!]. \tag{1.38b}$$

Differentiating (1.15a) twice and rescaling by $\frac{1}{2}\mu^2$ gives:

$$E\big(\tfrac{1}{2}\mu^2\ddot{\omega}_\mu\big) = 0. \tag{1.39}$$

Differentiating $\bar{z}_\mu + \mu g(\bar{s}_\mu) = 0$ twice gives:

$$\ddot{\bar{z}}_\mu + 2H(\bar{s}_\mu)\dot{\bar{s}}_\mu + \mu\nabla^3 f(\bar{s}_\mu)[\dot{\bar{s}}_\mu, \dot{\bar{s}}_\mu] + \mu H(\bar{s}_\mu)\ddot{\bar{s}}_\mu = 0. \tag{1.40}$$

Rescaling (1.40) by $\frac{1}{2}\mu^2$ and substituting the TOO definition (1.5), we have:

$$\begin{aligned} \tfrac{1}{2}\mu^2\ddot{\bar{z}}_\mu + \mu H(\bar{s}_\mu)\big(\tfrac{1}{2}\mu^2\ddot{\bar{s}}_\mu\big) &= \mu H(\bar{s}_\mu)(-\mu\dot{\bar{s}}_\mu) - \tfrac{1}{2}\mu\nabla^3 f(\bar{s}_\mu)[-\mu\dot{\bar{s}}_\mu, -\mu\dot{\bar{s}}_\mu] && \text{(1.41a)} \\ &= \mu H(\bar{s}_\mu)(-\mu\dot{\bar{s}}_\mu) + \mu\mathrm{T}(\bar{s}_\mu, -\mu\dot{\bar{s}}_\mu). && \text{(1.41b)} \end{aligned}$$

We interpret the prediction TOA direction, which adjusts the prediction direction, as $\delta^{pt} = \frac{1}{2}\mu^2\ddot{\omega}$. The RHS of (1.41b) depends on $\dot{\bar{s}}_\mu$, so we remove this dependency by substituting the prediction direction $\delta^p = -\mu\dot{\omega}_\mu$, which we assume is already computed. Hence using (1.39) and (1.41b), we let δ^{pt} be the solution to:

$$\begin{aligned} E\delta &= 0, && \text{(1.42a)} \\ \delta_{\bar{z},k} + \mu H_k(\bar{s}_k)\delta_{\bar{s},k} &= \mu H_k(\bar{s}_k)\delta^p_{\bar{s},k} + \mu\mathrm{T}_k\big(\bar{s}_k, \delta^p_{\bar{s},k}\big) \qquad \forall k \in [\![\bar{K}]\!]. && \text{(1.42b)} \end{aligned}$$

We note that the RHS in (1.42b) differs from the 'higher order corrector' RHS proposed by Dahl and E. D. Andersen (2021, Equation 16), which has the form $\frac{1}{2}\nabla^3 f_k\big[\delta^p_{\bar{s},k}, (H_k(\bar{s}_k))^{-1}\delta^p_{\bar{z},k}\big]$. For example, our form does not satisfy all of the properties in Dahl and E. D. Andersen (2021, Lemmas 3 and 4).

1.5.5 Stepping procedures

A stepping procedure computes one or more directions from Section 1.5.4 and uses the directions to search for a new interior point. Recall from Line 5 of the high level IPM in Section 1.5.3 that on iteration i with current iterate ω^{i-1}, the *Step* procedure computes ω^i satisfying $\pi(\omega^i) < 1$ and either $\mu(\omega^i) < \mu(\omega^{i-1})$ (prediction) or $\pi(\omega^i) < \pi(\omega^{i-1})$ (centering) or both. In Section 1.5.5.1, we describe a baseline stepping procedure mirroring that of Alfonso, which is an implementation of the SY algorithm with worst-case polynomial time iteration complexity. This procedure, which we call *basic*, alternates between prediction and centering steps and does not use the TOA directions. In Sections 1.5.5.2 to 1.5.5.5, we describe a sequence of four cumulative enhancements to the *basic* procedure. The goal is to improve iteration counts or per-iteration computational efficiency in practice, without regard for theoretical iteration complexity guarantees. Our computational testing in Section 1.9 assesses the value of these enhancements on a diverse set of benchmark instances.

1.5.5.1 Basic stepping procedure

First, we decide whether to perform a centering step or a prediction step. If the current iterate ω^{i-1} (at the ith iteration) is very close to the central path, i.e. if the sum proximity (1.22) does not exceed $\eta = 0.0332$, or if the most recent $N = 4$ steps have all been centering steps, then we compute the prediction direction δ^p (note these parameter values are taken directly from Alfonso and are based on the theoretical analysis of Papp and Yıldız (2017)). Otherwise, we compute the centering

direction δ^c from (1.30). Letting j be the number of consecutive centering steps taken immediately before the current ith iteration, the search direction is:

$$\delta := \begin{cases} \delta^p & \text{if } \pi_{\ell_2}(\omega^{i-1}) \leq \eta \text{ or } j \geq N, \\ \delta^c & \text{otherwise.} \end{cases} \tag{1.43}$$

Next, we perform a backtracking line search in the direction δ. The search finds a step length $\hat{\alpha} \in (0, 1)$ from a fixed schedule of decreasing values $\mathcal{A} = \{\alpha_l\}_{l \in [\![L]\!]}$, where $L = 18$, $\alpha_1 = 0.9999$, and $\alpha_L = 0.0005$. The next iterate $\omega^i = \omega^{i-1} + \hat{\alpha}\delta$ becomes the first point in the backtracking line search that satisfies $\pi_{\ell_2}(\omega^i) \leq \beta_1$ for $\beta_1 = 0.2844$, which guarantees interiority by Lemma 1.5.1 (note β_1 is again taken directly from Alfonso and is based on the theoretical analysis of Papp and Yıldız (2017)). If the backtracking search terminates without a step length satisfying the proximity condition (i.e. α_L is too large), the IPM algorithm terminates without a solution. In Section 1.7 we discuss our implementation of the proximity check that we run for each candidate point in the backtracking search.

The *basic* stepping procedure is summarized as follows. Note the centering step count j is initialized to zero before the first iteration $i = 1$. Since ω^0 is exactly on the central path (i.e. the proximity is zero), the first iteration uses a prediction step.

Procedure `BasicStep`(ω^{i-1}, j)**:**
- **if** $\pi_{\ell_2}(\omega^{i-1}) \leq \eta$ *or* $j \geq N$ **then** ▷ choose predict or center
  - $\delta \leftarrow \delta^p$ from (1.38) ▷ compute prediction direction
  - $j \leftarrow 0$
- **else**
  - $\delta \leftarrow \delta^c$ from (1.30) ▷ compute centering direction
  - $j \leftarrow j + 1$
- $\hat{\alpha} \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_2}(\omega^{i-1} + \alpha\delta) \leq \beta_1\}$ ▷ compute step length by backtracking search
- $\omega^i \leftarrow \omega^{i-1} + \hat{\alpha}\delta$ ▷ update current iterate
- **return** ω^i, j

1.5.5.2 Less restrictive proximity

The *basic* stepping procedure in Section 1.5.5.1 requires iterates to remain in close proximity to the central path and usually only takes prediction steps from iterates that are very close to the central path. Although conservative proximity conditions are used to prove polynomial iteration complexity in Papp and Yıldız (2017), they may be too restrictive from the perspective of practical performance. To allow prediction steps from a larger neighborhood of the central path, we use the π_{ℓ_∞} proximity measure from (1.23) instead of π_{ℓ_2} to compute the proximity of ω^{i-1}, though we do not change the proximity bound η. To allow the step length to be as large as possible, we use π_{ℓ_∞} instead of π_{ℓ_2} for the backtracking search proximity checks and we replace β_1 from Section 1.5.5.1 by

a larger proximity bound $\beta_2 = 0.99$. By Lemma 1.5.1, $\beta_2 < 1$ guarantees interiority, and our offline sensitivity analysis on β_2 suggests that 0.99 is a reasonable choice.[2]

The *prox* stepping procedure, which enhances the *basic* stepping procedure by relaxing the proximity conditions somewhat, is summarized as follows.

Procedure `ProxStep`(ω^{i-1}, j)**:**

- **if** $\pi_{\ell_\infty}(\omega^{i-1}) \leq \eta$ *or* $j \geq N$ **then** ▷ use less restrictive proximity measure π_{ℓ_∞}
  - $\delta \leftarrow \delta^p$ from (1.38)
  - $j \leftarrow 0$
- **else**
  - $\delta \leftarrow \delta^c$ from (1.30)
  - $j \leftarrow j + 1$
- $\hat{\alpha} \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_\infty}(\omega^{i-1} + \alpha\delta) \leq \beta_2\}$ ▷ use π_{ℓ_∞} and larger proximity bound β_2
- $\omega^i \leftarrow \omega^{i-1} + \hat{\alpha}\delta$
- **return** ω^i, j

1.5.5.3 Third order adjustments

We modify the *prox* stepping procedure in Section 1.5.5.2 to incorporate the new TOA directions associated with the prediction and centering directions. In symmetric conic IPMs, it is common to compute a step length in the unadjusted prediction direction, use this step length to compute an adjusted direction, and then compute a step length in this final direction (see e.g. Vandenberghe (2010, Section 5.1) for CVXOPT's approach using the Mehrotra correction). Our approach is similar.

After deciding whether to predict or center (using the same criteria as *prox*), we compute the unadjusted direction δ^u (i.e. δ^p or δ^c) and its associated TOA direction δ^t (i.e. δ^{pt} or δ^{ct}). We perform a backtracking line search in direction δ^u (like *prox*) and use this unadjusted step length $\hat{\alpha}^u \in (0, 1)$ to scale down the TOA direction, letting the adjusted direction be $\delta^u + \hat{\alpha}^u\delta^t$. We perform a second backtracking line search in this final direction to compute the final step length $\hat{\alpha}$, using the same techniques and proximity condition as the first line search. If we think of $\hat{\alpha}^u$ as an approximation of $\hat{\alpha}$, then essentially the final step applies an adjustment of $\hat{\alpha}^2\delta^t$ to $\hat{\alpha}\delta^u$. Our derivations of the adjustment directions in Section 1.5.4 (particularly the centering direction) suggest that this is a reasonable heuristic for adjustment.

The *TOA* stepping procedure, which enhances the *prox* stepping procedure by incorporating the TOA directions, is summarized as follows.

[2] These results are available from the Hypatia wiki at https://github.com/chriscoey/Hypatia.jl/wiki. The experiments are run on our benchmark set from Section 1.9.

1 **Procedure** `TOAStep`(ω^{i-1}, j)**:**
2 **if** $\pi_{\ell_\infty}(\omega^{i-1}) \leq \eta$ *or* $j \geq N$ **then**
3 $\delta^u \leftarrow \delta^p$ from (1.38)
4 $\delta^t \leftarrow \delta^{pt}$ from (1.42) ▷ compute prediction TOA direction
5 $j \leftarrow 0$
6 **else**
7 $\delta^u \leftarrow \delta^c$ from (1.30)
8 $\delta^t \leftarrow \delta^{ct}$ from (1.32) ▷ compute centering TOA direction
9 $j \leftarrow j + 1$
10 $\hat{\alpha}^u \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_\infty}(\omega^{i-1} + \alpha\delta^u) \leq \beta_2\}$ ▷ perform line search for unadjusted direction
11 $\delta \leftarrow \delta^u + \hat{\alpha}^u\delta^t$ ▷ compute final direction
12 $\hat{\alpha} \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_\infty}(\omega^{i-1} + \alpha\delta) \leq \beta_2\}$
13 $\omega^i \leftarrow \omega^{i-1} + \hat{\alpha}\delta$
14 **return** ω^i, j

1.5.5.4 Curve search

The *TOA* stepping procedure in Section 1.5.5.3 performs two backtracking line searches, which can be quite expensive. We propose using a single backtracking search along a curve that is quadratic in the step parameter α and linear in the unadjusted and TOA directions. Recall from Line 11 of the *TOA* procedure that we compute a direction δ as a linear function of the step parameter from the first line search. Substituting this δ function into the usual linear trajectory gives the curved trajectory $\omega^{i-1} + \alpha(\delta^u + \alpha\delta^t)$ for $\alpha \in (0, 1)$, where δ^u and δ^t are the unadjusted and TOA directions (as in the *TOA* procedure). Intuitively, a backtracking search along this curve achieves a more dynamic rescaling of the TOA direction.

The *curve* stepping procedure, which enhances the *TOA* stepping procedure by using a search on a curve instead of two line searches, is summarized as follows.

1 **Procedure** `CurveStep`(ω^{i-1}, j)**:**
2 **if** $\pi_{\ell_\infty}(\omega^{i-1}) \leq \eta$ *or* $j \geq N$ **then**
3 $\delta^u \leftarrow \delta^p$ from (1.38)
4 $\delta^t \leftarrow \delta^{pt}$ from (1.42)
5 $j \leftarrow 0$
6 **else**
7 $\delta^u \leftarrow \delta^c$ from (1.30)
8 $\delta^t \leftarrow \delta^{ct}$ from (1.32)
9 $j \leftarrow j + 1$
10 let $\hat{\omega}(\alpha) := \omega^{i-1} + \alpha(\delta^u + \alpha\delta^t)$ ▷ use curved trajectory
11 $\hat{\alpha} \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_\infty}(\hat{\omega}(\alpha)) \leq \beta_2\}$
12 $\omega^i \leftarrow \hat{\omega}(\hat{\alpha})$
13 **return** ω^i, j

1.5.5.5 Combined directions

Unlike Skajaa and Ye (2015) and Papp and Yıldız (2021), most conic IPMs combine the prediction and centering phases (e.g. Vandenberghe (2010) and Dahl and E. D. Andersen (2021)). We propose using a single search on a curve that is quadratic in the step parameter α and linear in all four directions $\delta^c, \delta^{ct}, \delta^p, \delta^{pt}$ from Section 1.5.5.3. Intuitively, we can step further in a convex combination of the prediction and centering directions than we can in just the prediction direction. In practice, a step length of one is usually ideal for the centering phase, so we can imagine performing a backtracking search from the point obtained from a pure prediction step (with step length one) towards the point obtained from a pure centering step, terminating when we are close enough to the centering point to satisfy the proximity condition. This approach fundamentally differs from the previous procedures we have described because the search trajectory does not finish at the current iterate ω^{i-1}. If $\hat{\omega}^p(\alpha)$ and $\hat{\omega}^c(\alpha)$ are the prediction and centering curve search trajectories from Line 10 of the *curve* procedure, then we define the combined trajectory as $\hat{\omega}(\alpha) = \hat{\omega}^p(\alpha) + \hat{\omega}^c(1-\alpha)$. Note that $\alpha = 1$ corresponds to a full step in the adjusted prediction direction $\delta^p + \delta^{pt}$, and $\alpha = 0$ corresponds to a full step in the adjusted centering direction $\delta^c + \delta^{ct}$.

The *comb* stepping procedure, which enhances the *curve* stepping procedure by combining the prediction and centering phases, is summarized as follows. Note that unlike the previous procedures, there is no parameter j counting consecutive centering steps. Occasionally in practice, the backtracking search on Line 4 below fails to find a positive step value, in which case we perform a centering step according to Lines 10 to 12 of the *curve* procedure.

Procedure `CombStep`(ω^{i-1})**:**

compute $\delta^c, \delta^{ct}, \delta^p, \delta^{pt}$ from (1.30), (1.32), (1.38) and (1.42) ▷ use four directions instead of two

let $\hat{\omega}(\alpha) := \omega^{i-1} + \alpha(\delta^p + \alpha\delta^{pt}) + (1-\alpha)(\delta^c + (1-\alpha)\delta^{ct})$ ▷ use combined trajectory

$\hat{\alpha} \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_\infty}(\hat{\omega}(\alpha)) \leq \beta_2\}$

$\omega^i \leftarrow \hat{\omega}(\hat{\alpha})$

return ω^i

1.6 Preprocessing and solving for search directions

In this section, we discuss preprocessing and initial point finding procedures and techniques for solving structured linear systems for directions. Although Hypatia has various alternative options for these procedures, we only describe the set of options we fix in our computational experiments in Section 1.9, to give context for these results. These techniques are likely to be useful for other conic IPM implementations.

Given a conic model specified in the general primal conic form (1.7), we first rescale the primal and dual equality constraints (1.7b) and (1.8b) to improve the conditioning of the affine data. Next, we perform a QR factorization of A' and check whether any primal equalities are inconsistent (terminating if so). We use this factorization to modify c, G, h and eliminate all p primal equalities (removing dual variable y), reducing the dimension of the primal variable x from n to $n - p$. Next, we perform a QR factorization of the modified G. We use this factorization to check whether any dual equalities are inconsistent (terminating if so) and to remove any redundant dual equalities, further reducing the dimension of x. This factorization also allows us to cheaply compute an initial x^0 satisfying (1.17c). Since y is eliminated, we do not need to solve (1.18b) for y^0.

Starting from the initial interior point ω^0 defined in Section 1.5.1, we perform IPM iterations until the convergence conditions in Section 1.5.3 (in the preprocessed space) are met. Finally, we re-use the two QR factorizations to lift the approximate certificate for the preprocessed model to one for the original model. The residual norms for the lifted certificate could violate the convergence tolerances, but we have not found such violations to be significant on our benchmark instances.

During each IPM iteration, we solve the linear system (1.28) for a single LHS matrix and between one and four RHS vectors, to obtain directions vectors needed for one of the stepping procedures described in Section 1.5.5. Instead of factorizing the large square nonsymmetric block-sparse LHS matrix, we utilize its structure to reduce the size of the factorization needed. Some of these techniques are adapted from methods in CVXOPT (see Vandenberghe (2010, Section 10.3)).

First we eliminate s and κ, yielding a square nonsymmetric system, then we eliminate τ to get a symmetric indefinite system in x and z. Most interior point solvers use a sparse LDL factorization (with precomputed symbolic factorization) to solve this system. Although Hypatia can optionally do the same, we see improved performance on our benchmark instances by further reducing the

system. After eliminating z, we have a (generally dense) positive definite system, which we solve via a dense Cholesky factorization. In terms of the original dimensions of the model before preprocessing (assuming no redundant equalities), the side dimension of this system is $n - p$. Finally, after finding a solution to (1.28), we apply several rounds of iterative refinement in working precision to improve the solution quality.

We note that this Cholesky-based system solver method does not require explicit Hessian oracles, only oracles for left-multiplication by the Hessian or inverse Hessian. As we discuss in Sections 2.5, 2.6, 3.5 and 3.7, these optional oracles can be more efficient and numerically stable to compute for many exotic cones. For cones without these oracles, Hypatia calls the explicit Hessian matrix oracle, performing a Cholesky factorization of the Hessian if necessary. A deeper discussion of Hypatia's linear system solving techniques and optional cone oracles is outside the scope of this chapter.

1.7 Efficient proximity checks

In this section, we describe how Hypatia performs efficient backtracking searches and proximity checks. Recall that each stepping procedure in Section 1.5.5 uses at least one backtracking search (on a line or a curve) to find a point ω satisfying an aggregate proximity condition: $\pi_{\ell_2}(\omega) \leq \beta_1$ for the *basic* procedure in Section 1.5.5.1 or $\pi_{\ell_\infty}(\omega) \leq \beta_2$ for the procedures in Sections 1.5.5.2 to 1.5.5.5. In Section 1.5.2, we define π_{ℓ_2} and π_{ℓ_∞} in (1.22) and (1.23). For each primitive cone $k \in [\![\bar{K}]\!]$, $0 \leq \pi_k(\omega) \leq \pi_{\ell_\infty}(\omega) \leq \pi_{\ell_2}(\omega)$, and by Lemma 1.5.1, $\pi_k(\omega) < 1$ implies $\bar{s}_k \in \operatorname{int}\big(\bar{\mathcal{K}}_k\big)$ and $\bar{z}_k \in \operatorname{int}\big(\bar{\mathcal{K}}_k^*\big)$. We use a schedule of decreasing trial values for the step parameter α and accept the first value that yields a candidate point satisfying the aggregate proximity condition.

Suppose at a particular iteration of the backtracking search, we have the candidate point ω. We check a sequence of increasingly expensive conditions that are necessary for the proximity condition to hold for ω. First, we verify that $\bar{s}_k' \bar{z}_k > 0, \forall k \in [\![\bar{K}]\!]$, which is necessary for interiority (by a strict version of the dual cone inequality (1.1)). Note that this condition implies $\mu(\omega) > 0$. Next, we verify that $\rho_k(\omega) < \beta, \forall k \in [\![\bar{K}]\!]$, where $\rho_k(\omega)$ is:

$$\rho_k(\omega) \coloneqq \nu_k^{-1/2} |\bar{s}_k' \bar{z}_k / \mu - \nu_k| \geq 0. \tag{1.44}$$

In Lemma 1.7.1 below, we show that $\rho_k(\omega)$ is a lower bound on $\pi_k(\omega)$, so if $\rho_k(\omega) > \beta$ then then $\pi_k(\omega) > \beta$. Computing ρ_k is much cheaper than computing $\pi_k(\omega)$ as it does not require evaluating any cone oracles.

Next, we iterate over $k \in [\![\bar{K}]\!]$ to check first the primal feasibility oracle, then the optional dual feasibility oracle if implemented, and finally the proximity condition $\pi_k(\omega) < \beta$. Before computing $\pi_k(\omega)$, we check that the gradient and Hessian oracle evaluations approximately satisfy two logarithmic homogeneity conditions (Nesterov and Nemirovski, 1994, Proposition 2.3.4):

$$(g_k(\bar{s}_k))'(H_k(\bar{s}_k))^{-1} g_k(\bar{s}_k) = -\bar{s}_k' g_k(\bar{s}_k) = \nu_k. \tag{1.45}$$

This allows us to reject ω if the cone oracles and the proximity value $\pi_k(\omega)$ are likely to be numerically

inaccurate.

Lemma 1.7.1. Given a point ω for which $\mu(\omega) > 0$, for each $k \in [\![\bar{K}]\!]$, $0 \leq \rho_k(\omega) \leq \pi_k(\omega)$.

Proof. We fix $\mu = \mu(\omega) > 0$ for convenience. Let f_k be the ν_k-LHSCB for $\bar{\mathcal{K}}_k$, and let the conjugate of f_k be f_k^* (see (1.3)), which is a ν_k-LHSCB for $\bar{\mathcal{K}}_k^*$. Let $g_k^* \coloneqq \nabla f_k^*$ and $H_k^* \coloneqq \nabla^2 f_k^*$ denote the gradient and Hessian operators for f_k^*. Using the logarithmic homogeneity properties from Nesterov and Nemirovski (1994, Proposition 2.3.4), and from the definition of $\pi_k(\omega)$ in (1.20), we have:

$$(\pi_k(\omega))^2 = (\bar{z}_k/\mu + g_k(\bar{s}_k))'(H_k(\bar{s}_k))^{-1}(\bar{z}_k/\mu + g_k(\bar{s}_k)) \tag{1.46a}$$
$$= \mu^{-2}\bar{z}_k'(H_k(\bar{s}_k))^{-1}\bar{z}_k + 2\mu^{-1}\bar{z}_k'(H_k(\bar{s}_k))^{-1}g_k(\bar{s}_k) + (g_k(\bar{s}_k))'(H_k(\bar{s}_k))^{-1}g_k(\bar{s}_k) \tag{1.46b}$$
$$= \mu^{-2}\bar{z}_k'(H_k(\bar{s}_k))^{-1}\bar{z}_k - 2\mu^{-1}\bar{z}_k'\bar{s}_k + \nu_k. \tag{1.46c}$$

By Papp and Yıldız (2017, Equation 13), $(H_k(\bar{s}_k))^{-1} = H_k^*(-g_k(\bar{s}_k))$. Since f_k^* is a self-concordant barrier with parameter ν_k, by Nesterov et al. (2018, Equation 5.3.6) we have: $(\bar{z}_k' g_k^*(-g_k(\bar{s}_k)))^2 \leq \nu_k \bar{z}_k' H_k^*(-g_k(\bar{s}_k))\bar{z}_k$. Furthermore, $g_k^*(-g_k(\bar{s}_k)) = \bar{s}_k$. Using these facts, from (1.46) we have $\rho_k(\omega) \geq 0$ and:

$$(\pi_k(\omega))^2 = \mu^{-2}\bar{z}_k' H_k^*(-g_k(\bar{s}_k))\bar{z}_k - 2\mu^{-1}\bar{z}_k'\bar{s}_k + \nu_k \tag{1.47a}$$
$$\geq \nu_k^{-1}\mu^{-2}(\bar{z}_k'\bar{s}_k)^2 - 2\mu^{-1}\bar{z}_k'\bar{s}_k + \nu_k \tag{1.47b}$$
$$= \nu_k^{-1}(\bar{s}_k'\bar{z}_k/\mu - \nu_k) \tag{1.47c}$$
$$= (\rho_k(\omega))^2. \tag{1.47d}$$

Therefore, $\pi_k(\omega) \geq \rho_k(\omega) \geq 0$ for all $k \in [\![\bar{K}]\!]$. □

As an aside, we can use similar arguments to Lemma 1.7.1 to show that $\rho_k(\omega)$ also symmetrically bounds a conjugate proximity measure $\pi_k^*(\omega)$, which we define as:

$$\pi_k^*(\omega) \coloneqq \left\|(H_k^*(\bar{z}_k))^{-1/2}(\bar{s}_k/\mu + g_k^*(\bar{z}_k))\right\| \geq \nu_k^{-1/2}|\bar{z}_k'(\bar{s}_k/\mu + g_k^*(\bar{z}_k))| = \rho_k(\omega). \tag{1.48}$$

In general, we cannot check whether $\pi_k^*(\omega) < \beta$ because as we discuss in Section 1.1.2 we do not have access to fast and numerically stable conjugate barrier oracles (g_k^* and H_k^*).

1.8 Oracles for predefined exotic cones

Below we list two dozen exotic cone types that we have predefined through Hypatia's generic cone interface (see Section 1.3). Each of these cones is represented in the benchmark set of conic instances that we introduce in Section 1.9.1. Recall that we write any exotic cone $\mathcal{K}$ in vectorized form, i.e. as a subset of $\mathbb{R}^q$, where $q = \dim(\mathcal{K}) \geq 1$ is the cone dimension. We define vectorization operators in Section 0.3. Each cone is parametrized by at least one dimension and several cones have additional parameters such as numerical data; for convenience, we drop these parameters from the symbols we use to represent cone types. For each cone, we define the LHSCB that Hypatia uses below, and

we list the associated barrier parameter ν in Table 1.1. For several cones, we have implemented additional variants over complex numbers, but we omit these definitions here for simplicity. In general, the complex variants have the same oracle complexities and barrier parameters as the real variants.

Nonnegative cone. $\mathcal{K}_{\geq} \coloneqq \mathbb{R}_{\geq}$ is the (self-dual) nonnegative real vectors. We use the LHSCB $f(w) = -\log(w)$ (Nesterov, Todd, and Ye, 1997, Section 2.1).

Positive semidefinite (PSD) matrix cone. $\mathcal{K}_{\succeq} \coloneqq \left\{w \in \mathbb{R}^{\mathrm{sd}(d)} : \mathrm{mat}(w) \in \mathbb{S}^d_{\succeq}\right\}$ is the (self-dual) PSD matrices of side dimension d. We also implement a complex Hermitian variant; see Section 2.2.1.1. We use the LHSCB $f(w) = -\operatorname{logdet}(\mathrm{mat}(w))$ (Nesterov, Todd, and Ye, 1997, Section 2.2).

Doubly nonnegative cone. $\mathcal{K}_{\mathrm{DNN}} \coloneqq \mathcal{K}_{\geq} \cap \mathcal{K}_{\succeq}$ is the PSD matrices with nonnegative entries. For side dimension d, we use the LHSCB $f(w) = -\operatorname{logdet}(\mathrm{mat}(w)) - \sum_{j \in [\![d]\!], i \in [\![j-1]\!]} \log(\mathrm{mat}(w)_{i,j})$.

Sparse PSD cone. $\mathcal{K}_{\mathrm{sPSD}}$ is the PSD matrices of side dimension s with a fixed sparsity pattern $\mathcal{S}$ containing $d \geq s$ nonzeros (including all diagonal elements); see Section 2.5.3. The dual cone $\mathcal{K}^*_{\mathrm{sPSD}}$ is the symmetric matrices with pattern $\mathcal{S}$ for which there exists a PSD completion, i.e. an assignment of the elements not in $\mathcal{S}$ such that the full matrix is PSD. For simplicity, the complexity estimates in Table 1.1 assume the nonzeros are grouped under $J \geq 1$ supernodes, each containing at most l nodes, and the monotone degree of each node is no greater than a constant D (M. S. Andersen, Dahl, and Vandenberghe, 2013). We also implement a complex Hermitian variant. We use the LHSCB in Section 2.5.3.

Linear matrix inequality cone. $\mathcal{K}_{\mathrm{LMI}} \coloneqq \left\{w \in \mathbb{R}^d : \sum_{i \in [\![d]\!]} w_i P_i \in \mathbb{S}^s_{\succeq}\right\}$ are the vectors for which the matrix pencil of d matrices $P_i \in \mathbb{S}^s, \forall i \in [\![d]\!]$ is PSD. We assume $P_1 \succ 0$ so that we can use the initial interior point e_1. We also allow the P_i to be complex Hermitian. We use the LHSCB in Section 2.5.1.

Euclidean norm cone. $\mathcal{K}_{\ell_2} \coloneqq \{(u, w) \in \mathbb{R}_{\geq} \times \mathbb{R}^d : u \geq \|w\|\}$ is the (self-dual) epigraph of the ℓ_2 norm on $\mathbb{R}^d$ (AKA second order cone). We use the LHSCB $f(u, w) = -\log(u^2 - \|w\|^2)$ (Nesterov, Todd, and Ye, 1997, Section 2.3).

Euclidean norm square cone. $\mathcal{K}_{\mathrm{sqr}} \coloneqq \{(u, v, w) \in \mathbb{R}_{\geq} \times \mathbb{R}_{\geq} \times \mathbb{R}^d : 2uv \geq \|w\|^2\}$ is the (self-dual) epigraph of the perspective of the square of the ℓ_2 norm on $\mathbb{R}^d$ (AKA rotated second order cone). We use the LHSCB $f(u, v, w) = -\log(2uv - \|w\|^2)$ (Nesterov, Todd, and Ye, 1997, Section 2.3).

Infinity norm cone. $\mathcal{K}_{\ell_\infty} \coloneqq \{(u, w) \in \mathbb{R}_{\geq} \times \mathbb{R}^d : u \geq \|w\|_\infty\}$ is the epigraph of the ℓ_∞ norm on $\mathbb{R}^d$; see Section 2.2.2.1. The dual cone $\mathcal{K}^*_{\ell_\infty}$ is the epigraph of the ℓ_1 norm. We also implement a complex vector variant. We use the LHSCB $f(u, w) = (d-1)\log(u) - \sum_{i \in [\![d]\!]} \log(u^2 - w_i^2)$ (Güler, 1996, Section 7.5).

Symmetric matrix spectral norm cone. $\mathcal{K}_{\ell_{\text{sspec}}} \coloneqq \{(u,w) \in \mathbb{R}_{\geq} \times \mathbb{R}^{\text{sd}(d)} : u \geq \sigma_1(W)\}$, where $W \coloneqq \text{mat}(w) \in \mathbb{S}^d$ and $\sigma_1(W)$ is the ℓ_∞ norm of the eigenvalues of W, is the epigraph of the spectral norm on the symmetric matrices; see Section 2.2.2.2. Similarly, $\mathcal{K}^*_{\ell_{\text{sspec}}}$ is the epigraph of the symmetric matrix nuclear norm (ℓ_1 norm of eigenvalues). We also implement a complex Hermitian variant. We specialize the $\mathcal{K}_{\ell_{\text{spec}}}$ LHSCB given below.

Rectangular matrix spectral norm cone. $\mathcal{K}_{\ell_{\text{spec}}} \coloneqq \{(u,w) \in \mathbb{R}_{\geq} \times \mathbb{R}^{ds} : u \geq \sigma_1(W)\}$, where $W \coloneqq \text{mat}(w) \in \mathbb{R}^{d \times s}$ and σ_1 is the largest singular value function, is the epigraph of the spectral norm, assuming $d \leq s$ without loss of generality; see Sections 2.2.2.2 and 2.6. Similarly, $\mathcal{K}^*_{\ell_{\text{spec}}}$ is the epigraph of the matrix nuclear norm (sum of singular values). We also implement a complex matrix variant. We use the LHSCB $f(u,w) = (d-1)\log(u) - \text{logdet}(u^2 I(d) - WW')$ (Nesterov and Nemirovski, 1994, Section 5.4.6).

Matrix square cone. $\mathcal{K}_{\text{matsqr}} \coloneqq \big\{(u,v,w) \in \mathbb{R}^{\text{sd}(d)} \times \mathbb{R}_{\geq} \times \mathbb{R}^{ds} : U \in \mathbb{S}^d_{\succeq}, 2Uv \succeq WW'\big\}$, where $U \coloneqq \text{mat}(u)$ and $W \coloneqq \text{mat}(w) \in \mathbb{R}^{d \times s}$, is the homogenized symmetric matrix epigraph of the symmetric outer product, assuming $d \leq s$ without loss of generality (Güler and Tunçel, 1998). We also implement a complex matrix variant. We use the LHSCB $f(u,v,w) = (d-1)\log(v) - \text{logdet}(2vU - WW')$ (Tunçel et al., 2004).

Generalized power cone. $\mathcal{K}_{\text{gpow}} \coloneqq \big\{(u,w) \in \mathbb{R}^r_{\geq} \times \mathbb{R}^s : \prod_{i \in [\![r]\!]} u_i^{\alpha_i} \geq \|w\|\big\}$, parametrized by exponents $\alpha \in \mathbb{R}^r_{>}$ with $e'\alpha = 1$, is the generalized power cone (Chares, 2009, Section 3.1.2). We use the LHSCB $f(u,w) = -\log\big(\prod_{i \in [\![r]\!]} u_i^{2\alpha_i} - \|w\|^2\big) - \sum_{i \in [\![r]\!]} (1-\alpha_i)\log(u_i)$ (Roy and Xiao, 2021).

Power mean cone. $\mathcal{K}_{\text{pow}} \coloneqq \big\{(u,w) \in \mathbb{R} \times \mathbb{R}^d_{\geq} : u \leq \prod_{i \in [\![d]\!]} w_i^{\alpha_i}\big\}$, parametrized by exponents $\alpha \in \mathbb{R}^d_{>}$ with $e'\alpha = 1$, is the hypograph of the power mean on $\mathbb{R}^d_{\geq}$. We use the LHSCB $f(u,w) = -\log\big(\prod_{i \in [\![d]\!]} w_i^{\alpha_i} - u\big) - \sum_{i \in [\![d]\!]} \log(w_i)$ (Nesterov et al., 2018, Section 5.4.7).

Geometric mean cone. $\mathcal{K}_{\text{geo}}$ is the hypograph of the geometric mean on $\mathbb{R}^d_{\geq}$, a special case of $\mathcal{K}_{\text{pow}}$ with equal exponents; see Section 2.2.3.1.

Root-determinant cone. $\mathcal{K}_{\text{rtdet}} \coloneqq \big\{(u,w) \in \mathbb{R} \times \mathbb{R}^{\text{sd}(d)} : W \in \mathbb{S}^d_{\succeq}, u \leq (\det(W))^{1/d}\big\}$, where $W \coloneqq \text{mat}(w)$, is the hypograph of the dth-root-determinant on $\mathbb{S}^d_{\succeq}$; see Section 2.2.3.1. We also implement a complex Hermitian variant. We use the LHSCB $f(u,w) = -\log((\det(W))^{1/d} - u) - \text{logdet}(W)$ from Proposition 3.7.1.

Logarithm cone. $\mathcal{K}_{\log} \coloneqq \text{cl}\big\{(u,v,w) \in \mathbb{R} \times \mathbb{R}_{>} \times \mathbb{R}^d_{>} : u \leq \sum_{i \in [\![d]\!]} v\log(w_i/v)\big\}$ is the hypograph of the perspective of the sum of logarithms on $\mathbb{R}^d_{>}$; see Section 2.2.3.2. We use the LHSCB $f(u,v,w) = -\log\big(\sum_{i \in [\![d]\!]} v\log(w_i/v) - u\big) - \log(v) - \sum_{i \in [\![d]\!]} \log(w_i)$ from Proposition 3.6.1.

Log-determinant cone. $\mathcal{K}_{\text{logdet}} \coloneqq \text{cl}\big\{(u,v,w) \in \mathbb{R} \times \mathbb{R}_{>} \times \mathbb{R}^{\text{sd}(d)} : W \in \mathbb{S}^d_{\succ}, u \leq v\,\text{logdet}(W/v)\big\}$, where $W \coloneqq \text{mat}(w)$, is the hypograph of the perspective of the log-determinant on $\mathbb{S}^d_{\succ}$; see Section 2.2.3.2. We also implement a complex Hermitian variant. We use the LHSCB $f(u,v,w) = -\log(v\,\text{logdet}(W/v) - u) - \log(v) - \text{logdet}(W)$ from Proposition 3.6.1.

Separable spectral function cone. $\mathcal{K}_{\text{sep}} \coloneqq \text{cl}\{(u,v,w) \in \mathbb{R} \times \mathbb{R}_> \times \text{int}(\mathcal{Q}) : u \geq v\varphi(w/v)\}$, where $\mathcal{Q}$ is a cone of squares of a Jordan algebra, is the epigraph of the perspective of a convex separable spectral function $\varphi : \text{int}(\mathcal{Q}) \to \mathbb{R}$, such as the trace of the negative logarithm, negative entropy, or power in $(1,2]$; see Sections 2.2.3.3 and 3.6 for suitable cones of squares and separable spectral functions. The complexity estimates in Table 1.1 depend on whether $\mathcal{Q}$ is a vector domain ($\mathcal{K}_{\geq}$) or a symmetric/Hermitian matrix domain ($\mathcal{K}_{\succeq}$). We use the LHSCB $f(u,v,w) = -\log(u - v\varphi(w/v)) - \log(v) - \text{logdet}(w)$ from Proposition 3.6.1.

Relative entropy cone. $\mathcal{K}_{\text{relent}} \coloneqq \text{cl}\big\{(u,v,w) \in \mathbb{R} \times \mathbb{R}^d_> \times \mathbb{R}^d_> : u \geq \sum_{i \in [\![d]\!]} w_i \log(w_i/v_i)\big\}$ is the epigraph of the vector relative entropy function. We use the LHSCB $f(u,v,w) = -\log\big(u - \sum_{i \in [\![d]\!]} w_i \log(w_i/v_i)\big) - \sum_{i \in [\![d]\!]} (\log(v_i) + \log(w_i))$ (Karimi and Tunçel, 2020a, Appendix E).

Matrix relative entropy cone. $\mathcal{K}_{\text{matrelent}} \coloneqq \text{cl}\big\{(u,v,w) \in \mathbb{R} \times \mathbb{R}^{\text{sd}(d)} \times \mathbb{R}^{\text{sd}(d)} : V \in \mathbb{S}^d_\succ, W \in \mathbb{S}^d_\succ, u \geq \text{tr}(W(\log(W) - \log(V)))\big\}$, where $V \coloneqq \text{mat}(v)$ and $W \coloneqq \text{mat}(w)$, is the epigraph of the matrix relative entropy function. We use the logarithmically homogeneous barrier $f(u,v,w) = -\log(u - \text{tr}(W(\log(W) - \log(V)))) - \text{logdet}(V) - \text{logdet}(W)$, which is conjectured to be self-concordant by Karimi and Tunçel (2020a).

Polynomial weighted sum-of-squares (SOS) cones. An interpolant basis represents a polynomial implicitly by its evaluations at a fixed set of d points. Given a basic semialgebraic domain defined by r polynomial inequalities, the four SOS cones below are parameterized by matrices $P_l \in \mathbb{R}^{d \times s_l}$ for $l \in [\![r]\!]$. Each P_l is constructed by evaluating s_l independent polynomials (columns) at the d points (rows), following Papp and Yıldız (2019). For simplicity, the complexity estimates in Table 1.1 assume $s_l = s, \forall l \in [\![r]\!]$. Note that $s < d \leq s^2$. We define $\mathcal{K}_{\text{SOS}}$ and $\mathcal{K}_{\text{matSOS}}$ in Section 2.2.1.3, and $\mathcal{K}_{\ell_1\text{SOS}}$ and $\mathcal{K}_{\ell_2\text{SOS}}$ in Kapelevich, Coey, and Vielma (2021, Equations 2.7 and 4.10). We use LHSCBs for the dual cones of these SOS cones. For $\mathcal{K}^*_{\text{SOS}}$ and $\mathcal{K}^*_{\text{matSOS}}$, we discuss the LHSCBs in Section 2.5.2. For $\mathcal{K}^*_{\ell_1\text{SOS}}$ and $\mathcal{K}^*_{\ell_2\text{SOS}}$, the LHSCBs require more complex notation, so we refer the reader to Kapelevich, Coey, and Vielma (2021).

- **Scalar SOS cone.** $\mathcal{K}_{\text{SOS}}$ is a cone of polynomials that are guaranteed to be nonnegative pointwise on the domain. We also implement a real-valued complex polynomial (Hermitian SOS) variant, for which the P_l matrices are complex.
- **Symmetric matrix SOS cone.** $\mathcal{K}_{\text{matSOS}}$ is a cone of polynomial symmetric matrices (in an svec-like format) of side dimension t that are guaranteed to belong to $\mathbb{S}_\succeq$ pointwise on the domain. We let $m \coloneqq st + d$ in Table 1.1 for succinctness.
- ℓ_1 **epigraph SOS cone.** $\mathcal{K}_{\ell_1\text{SOS}}$ is a cone of polynomial vectors of length $1 + t$ that are guaranteed to belong to $\mathcal{K}^*_{\ell_\infty}$ pointwise on the domain.
- ℓ_2 **epigraph SOS cone.** $\mathcal{K}_{\ell_2\text{SOS}}$ is a cone of polynomial vectors of length $1 + t$ that are guaranteed to belong to $\mathcal{K}_{\ell_2}$ pointwise on the domain.

For each cone, we have an analytic/closed form for the feasibility check, gradient, Hessian, and TOO oracles defined in Section 1.3. That is, we always avoid iterative numerical procedures such as optimization, which are typically slow, numerically unstable, and require tuning. Hypatia's algorithm always evaluates the feasibility check before the gradient, Hessian, and TOO (which are only defined at strictly feasible points), and the gradient is evaluated before the Hessian and TOO. For most of these cones, the feasibility check and gradient oracles compute values and factorizations that are also useful for computing the Hessian and TOO, so this data is cached in the cone data structures and reused where possible. In Table 1.1, we estimate the time complexities (ignoring constants) of these four oracles for each cone, counting the cost of cached values and factorizations only once (for the oracle that actually computes them). Table 1.1 shows that the TOO is never more expensive than the feasibility check, gradient, and Hessian oracles (i.e. the oracles needed by SY). Indeed, our computational results in Section 1.9.3 demonstrate that the TOO is very rarely an algorithmic bottleneck in practice.

Table 1.1: Cone dimension $\dim(\mathcal{K})$, LHSCB parameter ν, and time complexity estimates (ignoring constants) for our feasibility check, gradient, Hessian, and TOO implementations, for the exotic cones defined in Section 1.8.

cone	$\dim(\mathcal{K})$	ν	feasibility	gradient	Hessian	TOO
$\mathcal{K}_{\geq}$	d	d	d	d	d	d
$\mathcal{K}_{\succeq}$	$\operatorname{sd}(d)$	d	d^3	d^3	d^4	d^3
$\mathcal{K}_{\text{DNN}}$	$\operatorname{sd}(d)$	$\operatorname{sd}(d)$	d^3	d^3	d^4	d^3
$\mathcal{K}_{\text{sPSD}}$	d	s	JD^2l	JD^2l	dJD^2l	JD^2l
$\mathcal{K}_{\text{LMI}}$	d	s	$ds^2 + s^3$	ds^3	d^2s^2	$ds^2 + s^3$
$\mathcal{K}_{\ell_2}$, $\mathcal{K}_{\text{sqr}}$	$1 + d$	2	d	d	d^2	d
$\mathcal{K}_{\ell_\infty}$	$1 + d$	$1 + d$	d	d	d	d
$\mathcal{K}_{\ell_{\text{sspec}}}$	$1 + \operatorname{sd}(d)$	$1 + d$	d^3	d^3	d^4	d^3
$\mathcal{K}_{\ell_{\text{spec}}}$	$1 + ds$	$1 + d$	$d^2s + d^3$	d^2s	d^2s^2	d^2s
$\mathcal{K}_{\text{matsqr}}$	$\operatorname{sd}(d) + 1 + ds$	$1 + d$	$d^2s + d^3$	$d^2s + d^3$	d^2s^2	ds^2
$\mathcal{K}_{\text{gpow}}$	$r + s$	$1 + r$	$r + s$	$r + s$	$r^2 + s^2$	$r + s$
$\mathcal{K}_{\text{pow}}$, $\mathcal{K}_{\text{geo}}$	$1 + d$	$1 + d$	d	d	d^2	d
$\mathcal{K}_{\text{rtdet}}$	$1 + \operatorname{sd}(d)$	$1 + d$	d^3	d^3	d^4	d^3
$\mathcal{K}_{\log}$	$2 + d$	$2 + d$	d	d	d^2	d
$\mathcal{K}_{\text{logdet}}$	$2 + \operatorname{sd}(d)$	$2 + d$	d^3	d^3	d^4	d^3
$\mathcal{K}_{\text{sep}}(\mathbb{R})$	$2 + d$	$2 + d$	d	d	d^2	d
$\mathcal{K}_{\text{sep}}(\mathbb{S})$	$2 + \operatorname{sd}(d)$	$2 + d$	d^3	d^3	d^5	d^3
$\mathcal{K}_{\text{relent}}$	$1 + 2d$	$1 + 2d$	d	d	d^2	d
$\mathcal{K}_{\text{matrelent}}$	$1 + 2\operatorname{sd}(d)$	$1 + 2d$	d^3	d^3	d^5	d^4
$\mathcal{K}^*_{\text{SOS}}$	d	sr	ds^2r	ds^2r	d^2sr	ds^2r
$\mathcal{K}^*_{\text{matSOS}}$	$d\operatorname{sd}(t)$	str	ms^2t^2r	ds^2t^2r	d^2st^3r	ms^2t^2r
$\mathcal{K}^*_{\ell_1\text{SOS}}$	$d(1 + t)$	str	ds^2tr	ds^2tr	d^2str	ds^2tr
$\mathcal{K}^*_{\ell_2\text{SOS}}$	$d(1 + t)$	$2sr$	ds^2tr	ds^2tr	d^2st^2r	ds^2t^2r

Our TOO in (1.5) is distinct from the 'higher order corrector' terms proposed by Mehrotra (1992)

and Dahl and E. D. Andersen (2021). The method by Mehrotra (1992) only applies to symmetric cones, and Dahl and E. D. Andersen (2021) test their technique only for the standard exponential cone. Compared to the third order term proposed by Dahl and E. D. Andersen (2021), our TOO has a simpler and more symmetric structure, as it relies on only one direction $\delta_{\bar{s}}$ rather than two. Like the gradient and Hessian oracles, our TOO is additive for sums of LHSCBs, which can be useful for cones (such as $\mathcal{K}^*_{\text{SOS}}$ and $\mathcal{K}_{\text{DNN}}$) that are defined as intersections of other cones. We leverage these properties to obtain fast and numerically stable TOO implementations.

To illustrate, we give the oracles for $\mathcal{K}_{\ell_2}, \mathcal{K}_{\text{sqr}} \subset \mathbb{R}^q$. We use the standard LHSCBs with parameter $\nu = 2$ from Vandenberghe (2010, Section 2.2) and Nesterov, Todd, and Ye (1997, Section 2.3). For $\mathcal{K}_{\ell_2}, \mathcal{K}_{\text{sqr}}$, the LHSCB is $f(s) = -\log(s'Js)$, where $J \in \mathbb{S}^q$ is defined according to, for $i, j \in [\![q]\!] : i \geq j$:

$$J_{i,j} \coloneqq \begin{cases} 1 & \text{if } j = 1 \text{ and } (i = 1 \text{ for } \mathcal{K}_{\ell_2} \text{ or } i = 2 \text{ for } \mathcal{K}_{\text{sqr}}), \\ -1 & \text{if } i = j \text{ and } (i > 1 \text{ for } \mathcal{K}_{\ell_2} \text{ or } i > 2 \text{ for } \mathcal{K}_{\text{sqr}}), \\ 0 & \text{otherwise.} \end{cases} \tag{1.49}$$

Consider $s \in \text{int}(\mathcal{K})$ and direction $\delta \in \mathbb{R}^q$, and let $\bar{J} = (s'Js)^{-1} > 0$. The gradient, Hessian product, and TOO oracles for $\mathcal{K}$ are:

$$g(s) = -2\bar{J}Js, \tag{1.50a}$$

$$H(s)\delta = 2\bar{J}(2\bar{J}Jss'J\delta - J\delta), \tag{1.50b}$$

$$\text{T}(s, \delta) = \bar{J}(Js\delta'H\delta + H\delta s'J\delta - s'H\delta J\delta). \tag{1.50c}$$

These oracles are computed in $\mathcal{O}(q)$ time. The explicit Hessian matrix is computed in $\mathcal{O}(q^2)$ time as:

$$H(s) = 2\bar{J}(2\bar{J}Jss'J - J). \tag{1.51}$$

1.9 Computational testing

In Section 1.9.1, we introduce a diverse set of exotic conic benchmark instances generated from a variety of applied examples. In Section 1.9.2, we describe our methodology for comparing the stepping procedures from Section 1.5.5, and in Section 1.9.3 we examine our computational results.

1.9.1 Exotic conic benchmark set

We generate 379 instances (in our primal general form (1.7)) from 37 applied examples in Hypatia's examples folder. All instances are primal-dual feasible except for 12 that are primal infeasible and one that is dual infeasible. For most examples, we construct multiple formulations using different predefined exotic cones from the list in Section 1.8. Each cone from this list appears in at least one instance, so we consider our benchmark set to be the most diverse collection of conic instances

available.

We generate most instances using JuMP, but for some we use Hypatia's native model interface. Due to the size of some instances and the lack of a standard instance storage format recognizing our cone types, we generate all instances on the fly in Julia. For instances that use random data, we set random seeds to ensure reproducibility. Figure 1.1 shows the distributions of instance dimensions and exotic cone counts. All instances have at least one cone (note any $\mathcal{K}_{\geq}$ cones are concatenated together, so $\mathcal{K}_{\geq}$ is counted at most once) and take at least one iteration to solve with Hypatia.

Below we briefly introduce each example. In Table 1.2, we summarize for each example the number of corresponding instances and the cone types represented in at least one of the instances. We do not distinguish dual cones and primal cones in this summary (for example, instances that use $\mathcal{K}^*_{\ell_\infty}$ are only listed as using $\mathcal{K}_{\ell_\infty}$). For some examples, we describe a subset of formulations in Sections 2.3 and 3.8 and Kapelevich, Coey, and Vielma (2021). Our benchmark set includes ten instances from CBLIB (a conic benchmark instance library, see Friberg (2016)). We chose to avoid running a larger sample of instances from CBLIB so that the relatively few cone types supported by CBLIB version 3 are not over-represented in our benchmark set.

Central polynomial matrix. Minimize a spectral function of a gram matrix of a polynomial. See Section 3.8.4.3.

Classical-quantum capacity. Compute the capacity of a classical-to-quantum channel. Adapted from H. Fawzi and O. Fawzi (2018, Section 3.1). See Section 3.8.4.4.

Condition number. Minimize the condition number of a matrix pencil subject to a linear matrix inequality. Adapted from Boyd, El Ghaoui, et al. (1994, Section 3.2).

Contraction analysis. Find a contraction metric that guarantees global stability of a dynamical system. Adapted from Aylward, Parrilo, and Slotine (2008, Section 5.3). Six instances are primal infeasible.

Convexity parameter. Find the strong convexity parameter of a polynomial function over a domain.

Covariance estimation. Estimate a covariance matrix that satisfies some given prior information and minimizes a given convex spectral function.

Density estimation. Find a valid polynomial density function maximizing the likelihood of a set of observations. Adapted from Papp and Alizadeh (2014, Section 4.3). See Section 2.3.6.

Discrete maximum likelihood. Maximize the likelihood of some observations at discrete points, subject to the probability vector being close to a uniform prior.

D-optimal design. Solve a D-optimal experiment design problem, i.e. maximize the determinant of the information matrix subject to side constraints. Adapted from Boyd and Vandenberghe (2004, Section 7.5). See Section 2.3.4.

Entanglement-assisted capacity. Compute a quantum channel's entanglement-assisted classical capacity. Adapted from H. Fawzi and O. Fawzi (2018, Section 3.2).

Experiment design. Solve a general experiment design problem that minimizes a given convex spectral function of the information matrix subject to side constraints. Adapted from Boyd and Vandenberghe (2004, Section 7.5). See Section 3.8.4.2.

Linear program. Solve a simple linear program.

Lotka-Volterra. Find an optimal controller for a Lotka-Volterra model of population dynamics. Adapted from Korda, Henrion, and Jones (2016, Section 7.2).

Lyapunov stability. Minimize an upper bound on the root mean square gain of a dynamical system. Adapted from Boyd, El Ghaoui, et al. (1994, Section 6.3.2) and Boyd (2009, Page 6).

Matrix completion. Complete a rectangular matrix by minimizing the nuclear norm and constraining the missing entries. Adapted from Agrawal, Diamond, and Boyd (2019, Equation 8). See Section 2.3.2.

Matrix quadratic. Find a rectangular matrix that minimizes a linear function and satisfies a constraint on the outer product of the matrix.

Matrix regression. Solve a multiple-output (or matrix) regression problem with regularization terms, such as ℓ_1, ℓ_2, or nuclear norm. See Section 2.3.3.

Maximum volume hypercube. Find a maximum volume hypercube (with edges parallel to the axes) inside a given polyhedron or ellipsoid. Adapted from MOSEK ApS (2020a, Section 4.3.2).

Nearest correlation matrix. Compute the nearest correlation matrix in the quantum relative entropy sense. Adapted from H. Fawzi, Saunderson, and Parrilo (2019).

Nearest polynomial matrix. Given a symmetric matrix of polynomials H, find a polynomial matrix Q that minimizes the sum of the integrals of its elements over the unit box and guarantees $Q - H$ is pointwise PSD on the unit box.

Nearest PSD matrix. Find a sparse PSD matrix or a PSD-completable matrix (with a given sparsity pattern) with constant trace that maximizes a linear function. Adapted from Y. Sun and Vandenberghe (2015).

Nonparametric distribution. Given a random variable taking values in a finite set, compute the distribution minimizing a given convex spectral function over all distributions satisfying some prior information. See Section 3.8.4.1.

Norm cone polynomial. Given a vector of polynomials, check a sufficient condition for pointwise membership in $\mathcal{K}_{\ell_2}$ or $\mathcal{K}_{\ell_\infty}^*$. Four instances are primal infeasible.

Polynomial envelope. Find a polynomial that closely approximates, over the unit box, the lower envelope of a given list of polynomials. Adapted from Papp and Yıldız (2019, Section 7.2.1).

Polynomial minimization. Compute a lower bound for a given polynomial over a given semialgebraic set. Adapted from Papp and Yıldız (2019, Section 7.3.1). See Section 2.3.5. Some instances use polynomials with known optimal values from Burkardt (2016).

Polynomial norm. Find a polynomial that, over the unit box, has minimal integral and belongs pointwise to the epigraph of the ℓ_1 or ℓ_2 norm of other given polynomials. See Kapelevich, Coey, and Vielma (2021, Section 6).

Portfolio. Maximize the expected returns of a stock portfolio and satisfy various risk constraints. See Section 2.3.1.

Region of attraction. Find the region of attraction of a polynomial control system. Adapted from Henrion and Korda (2013, Section 9.1).

Relative entropy of entanglement. Compute a lower bound on relative entropy of entanglement with a positive partial transpose relaxation. Adapted from H. Fawzi and O. Fawzi (2018, Section 4).

Robust geometric programming. Bound the worst-case optimal value of an uncertain signomial function with a given coefficient uncertainty set. Adapted from Chandrasekaran and Shah (2017, Equation 39).

Semidefinite polynomial matrix. Check a sufficient condition for global convexity of a given polynomial. Two instances are primal infeasible and one is dual infeasible.

Shape constrained regression. Given a dataset, fit a polynomial function that satisfies shape constraints such as monotonicity or convexity over a domain. See Section 2.3.7. Several instances use real datasets from Mazumder et al. (2019).

Signomial minimization. Compute a global lower bound for a signomial function. Adapted from Murray, Chandrasekaran, and Wierman (2020). Several instances use signomials with known optimal values from Murray, Chandrasekaran, and Wierman (2020) and Chandrasekaran and Shah (2016).

Sparse LMI. Optimize over a simple linear matrix inequality with sparse data.

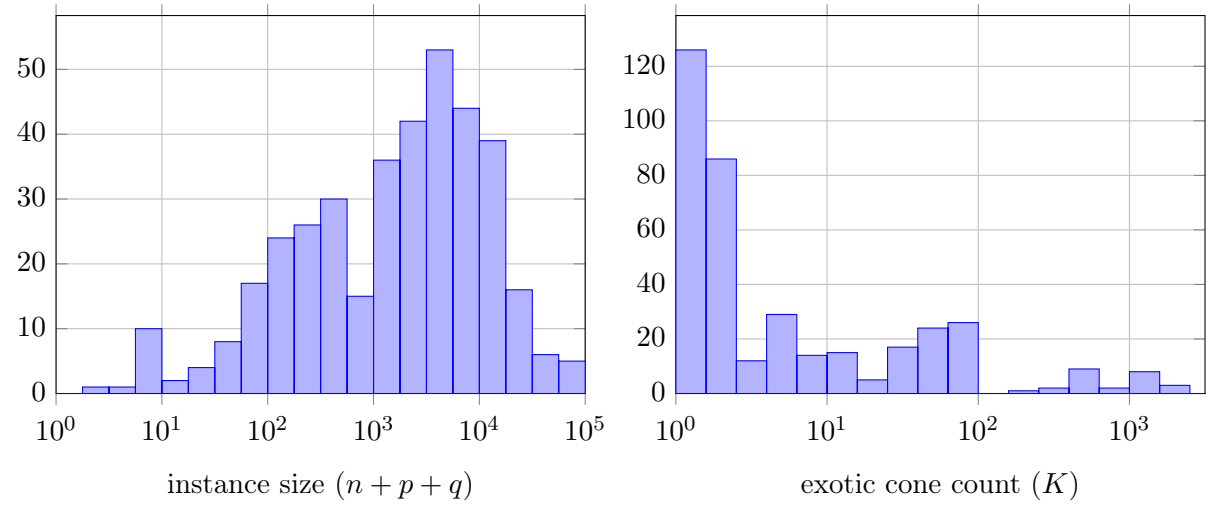
Sparse principal components. Solve a convex relaxation of the problem of approximating a symmetric matrix by a rank-one matrix with a cardinality-constrained eigenvector. Adapted from d'Aspremont et al. (2007, Section 2).

Stability number. Given a graph, solve for a particular strengthening of the theta function towards the stability number. Adapted from Laurent and Piovesan (2015, Equation 2.4).

Table 1.2: For each example, the count of instances and list of exotic cones (defined in Section 1.8) used in at least one instance.

example	#	cones in at least one instance
CBLIB	10	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{gpow}}$
central polynomial matrix	24	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{gpow}}$ $\mathcal{K}_{\text{rtdet}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sep}}$
classical-quantum capacity	9	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sep}}$
condition number	6	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{LMI}}$
contraction analysis	8	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{matSOS}}$
convexity parameter	7	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{matSOS}}$
covariance estimation	13	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{gpow}}$ $\mathcal{K}_{\text{rtdet}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sep}}$
density estimation	16	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{geo}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{SOS}}$
discrete maximum likelihood	7	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{pow}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sep}}$
D-optimal design	16	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{geo}}$ $\mathcal{K}_{\text{rtdet}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{logdet}}$
entanglement-assisted capacity	3	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{sep}}$ $\mathcal{K}_{\text{matrelent}}$
experiment design	13	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{gpow}}$ $\mathcal{K}_{\text{rtdet}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sep}}$
linear program	3	$\mathcal{K}_{\geq}$
Lotka-Volterra	3	$\mathcal{K}_{\succeq}$
Lyapunov stability	10	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{matsqr}}$
matrix completion	11	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\ell_{\text{spec}}}$ $\mathcal{K}_{\text{gpow}}$ $\mathcal{K}_{\text{geo}}$ $\mathcal{K}_{\log}$
matrix quadratic	8	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{matsqr}}$
matrix regression	11	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\ell_{\text{spec}}}$
maximum volume hypercube	15	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{geo}}$
nearest correlation matrix	3	$\mathcal{K}_{\text{matrelent}}$
nearest polynomial matrix	8	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{SOS}}$ $\mathcal{K}_{\text{matSOS}}$
nearest PSD matrix	28	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{sPSD}}$
nonparametric distribution	10	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{geo}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sep}}$
norm cone polynomial	10	$\mathcal{K}_{\ell_1\text{SOS}}$ $\mathcal{K}_{\ell_2\text{SOS}}$
polynomial envelope	7	$\mathcal{K}_{\text{SOS}}$
polynomial minimization	15	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{SOS}}$
polynomial norm	10	$\mathcal{K}_{\text{SOS}}$ $\mathcal{K}_{\text{matSOS}}$ $\mathcal{K}_{\ell_1\text{SOS}}$ $\mathcal{K}_{\ell_2\text{SOS}}$
portfolio	9	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\ell_2}$
region of attraction	6	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{SOS}}$
relative entropy of entanglement	6	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{matrelent}}$
robust geometric programming	6	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{relent}}$
semidefinite polynomial matrix	18	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{matSOS}}$
shape constrained regression	11	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{SOS}}$ $\mathcal{K}_{\text{matSOS}}$
signomial minimization	13	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{relent}}$
sparse LMI	15	$\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{sPSD}}$ $\mathcal{K}_{\text{LMI}}$
sparse principal components	6	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\ell_\infty}$
stability number	6	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\succeq}$ $\mathcal{K}_{\text{DNN}}$

Figure 1.1: Histograms summarizing the benchmark instances in the primal conic form (1.7). Instance size (log scale) is the sum of the primal variable, equality, and conic constraint dimensions. Exotic cone count (log scale) is the number of exotic cones comprising the Cartesian product cone.



1.9.2 Methodology

We can assess the practical performance of a stepping procedure on a given benchmark instance according to several metrics: whether the correct conic certificate (satisfying our numerical tolerances, discussed below) is found, and if so, the IPM iteration count and solve time. Across the benchmark set, we compare performance between consecutive pairs of the five stepping procedures outlined in Section 1.5.5.

basic. The basic prediction or centering stepping procedure without enhancements; described in Section 1.5.5.1, this is similar to the method in Alfonso solver (Papp and Yıldız, 2021), which is a practical implementation of the algorithm by Skajaa and Ye (2015) and Papp and Yıldız (2017).

prox. The *basic* procedure modified to use a less restrictive central path proximity condition; described in Section 1.5.5.2.

TOA. The *prox* procedure with the TOA enhancement to incorporate third order LHSCB information; described in Section 1.5.5.3.

curve. The *TOA* procedure adapted for a single backtracking search on a curve instead of two backtracking line searches; described in Section 1.5.5.4.

comb. The *curve* procedure modified to search along a curve of combinations of both the prediction and centering directions and their corresponding adjustment directions; described in Section 1.5.5.5.

We perform all instance generation, computational experiments, and results analysis using double precision floating point format, with Ubuntu 21.04, Julia 1.7, and Hypatia 0.5.2 (with default options),

on dedicated hardware with an AMD Ryzen 9 3950X 16-core processor (32 threads) and 128GB of RAM. In Section 1.6, we outline the default procedures Hypatia uses for preprocessing, initial point finding, and linear system solving for search directions. Simple scripts and instructions for reproducing all results are available in Hypatia's benchmarks/stepper folder. The benchmark script runs all solves twice and uses results from the second run, to exclude Julia compilation overhead. A CSV file containing raw results is available at the Hypatia wiki page.

When Hypatia converges for an instance, i.e. claims it has found a certificate of optimality, primal infeasibility, or dual infeasibility, our scripts verify that this is the correct type of certificate for that instance. For some instances, our scripts also check additional conditions, for example that the objective value of an optimality certificate approximately equals the known true optimal value. We do not set restrictive time or iteration limits. All failures to converge are caused by Hypatia 'stalling' during the stepping iterations: either the backtracking search cannot step a distance of at least the minimal value in the α schedule, or across several prediction steps or combined directions steps, Hypatia fails to make sufficient progress towards meeting the convergence conditions in Section 1.5.3.

Since some instances are more numerically challenging than others, we set the termination tolerances (described in Section 1.5.3) separately for each instance. Let $\epsilon \approx 2.22 \times 10^{-16}$ be the machine epsilon. For most instances, we use $\varepsilon_f = \varepsilon_r = 10\epsilon^{1/2} \approx 1.49 \times 10^{-7}$ for the feasibility and relative gap tolerances, $\varepsilon_i = \varepsilon_a = 10\epsilon^{3/4} \approx 1.82 \times 10^{-11}$ for the infeasibility and absolute gap tolerances, and $\varepsilon_p = 0.1\epsilon^{3/4} \approx 1.82 \times 10^{-13}$ for the ill-posedness tolerance. For 50 instances that are particularly numerically challenging, we loosen all of these tolerances by a factor of either 10 or 100, and for two challenging primal infeasible instances of the *contraction analysis* example, we set $\varepsilon_i = 10^{-9}$. This ensures that for every benchmark instance, at least one of the five stepping procedures converges.

Following Fleming and Wallace (1986), we define the *shifted geometric mean* with shift $s \geq 0$, for d values $v \in \mathbb{R}^d_>$, as:

$$M(v, s) := \prod_{i \in [\![d]\!]} (v_i + s)^{1/d} - s. \tag{1.52}$$

We always apply a shift of one for iteration counts. Since different stepping procedures converge on different subsets of instances, in tables we show three types of shifted geometric means, each computed from a vector of values (v in (1.52)) obtained using one of the following approaches.

every. Values for the 353 instances on which every stepping procedure converged.

this. Values for instances on which this stepping procedure (corresponding to the row of the table) converged.

all. Values for all instances, but for any instances for which this stepping procedure (corresponding to the row of the table) failed to converge, the value is replaced with two times the maximum value for that instance across the stepping procedures that converged.

The shifted geometric means for the *every* approach are the most directly comparable because they are computed on a fixed subset of instances, so we usually quote the *every* results in our discussion in Section 1.9.3.

Table 1.3 shows counts of converged instances and shifted geometric means of iteration count and total solve time (in milliseconds), for the five stepping procedures. We use a shift of one millisecond for the solve times in Table 1.3, as some instances solve very quickly (see Figure 1.2).

Table 1.4 shows shifted geometric means of the time (in milliseconds) Hypatia spends performing each of the following key algorithmic components, for the five stepping procedures.

init. Performed once during an entire solve run, independently of the stepping iterations. Includes rescaling and preprocessing of model data, initial interior point finding, and linear system solver setup (see Section 1.6).

LHS. Performed at the start of each iteration. Includes updating data that the linear system solver (which has a fixed LHS in each iteration) uses to efficiently compute at least one direction (such as updating and factorizing the positive definite matrix in Section 1.6).

RHS. Performed between one and four times per iteration, depending on the stepping procedure. Includes updating an RHS vector (see (1.28)) for the linear system for search directions. Note that the TOO is only evaluated while computing the centering TOA RHS (1.32b) and the prediction TOA RHS (1.42b).

direc. Performed for each RHS vector. Includes solving the linear system for a search direction (see (1.28)) using the data computed during *LHS* and a single RHS vector computed during *RHS*, and performing iterative refinement on the direction (see Section 1.6).

search. Performed once or twice per iteration (occasionally more if the step length is near zero), depending on the stepping procedure. Includes searching using backtracking along a line or curve to find an interior point satisfying the proximity conditions (see Section 1.7).

For some instances that solve extremely quickly, these subtimings sum to only around half of the total solve time due to extraneous overhead. However for slower instances, these components account for almost the entire solve time. In Table 1.4, *total* is the time over all iterations, and *per iteration* is the average time per iteration (the arithmetic means are computed before the shifted geometric mean). We use a shift of 0.1 milliseconds for the *init* and *total* subtimings (left columns) and a shift of 0.01 milliseconds for the *per iteration* subtimings (right columns).

Finally, in Figures 1.4 and 1.7 we use *performance profiles* (Dolan and Moré, 2002; Gould and Scott, 2016) to compare iteration counts and solve times between pairs of stepping procedures. These should be interpreted as follows. The *performance ratio* for procedure i and instance j is the value (iterations or solve time) attained by procedure i on instance j divided by the better/smaller value attained by the two procedures on instance j. Hence a performance ratio is at least one, and smaller values indicate better relative performance. For a point (x, y) on a performance profile curve for a particular procedure, x is the logarithm (base 2) of performance ratio and y is the proportion of instances for which the procedure attains that performance ratio or better/smaller. For example, a curve crosses the vertical axis at the proportion of instances on which the corresponding procedure

performed at least as well as the alternative procedure. We use the Julia package BenchmarkProfiles.jl (Orban, 2019) to compute coordinates for the performance profile curves.

1.9.3 Results

Table 1.3 and Figure 1.7 demonstrate that each of the four cumulative stepping enhancements tends to improve Hypatia's iteration count and solve time. The enhancements do not have a significant impact on the number of instances Hypatia converges on. However, if we had enforced time or iteration limits, the enhancements would have also improved the number of instances solved. This is clear from Figure 1.2, which shows the distributions of iteration counts and solve times for the *basic* and *comb* stepping procedures.

We note that Figure 1.5 (left) supports the intuition that formulation size is strongly positively correlated with solve time for *comb*. Furthermore, Figure 1.3 shows a positive correlation between iteration count and instance barrier parameter ν, for both the *basic* and *comb* steppers. This is expected for the *basic* stepper, as the theoretical worst-case iteration complexity for the SY algorithm is proportional to $\sqrt{\nu}$ (Skajaa and Ye, 2015; Papp and Yıldız, 2017). However we note that on our benchmark set, ν is also correlated with the instance size (particularly the cone dimension q) and exotic cone count K, which may also affect iteration counts in practice.

Overall, Table 1.3 shows that on the subset of instances solved by every stepping procedure (*every*), the enhancements together reduce the shifted geometric means of iterations and solve time by more than 80% and 70% respectively (i.e. comparing *comb* to *basic*). Figure 1.4 shows that the iteration count and solve time improve on nearly every instance solved by both *basic* and *comb*, and the horizontal axis scale shows that the magnitude of these improvements is large on most instances. Figure 1.6 shows that for instances that take more iterations or solve time, the enhancements tend to yield a greater improvement in these measures. On every instance, the enhancements improve the iteration count by at least 33%. The few instances for which solve time regressed with the enhancements all solve relatively quickly.

Table 1.3: For each stepping procedure, the number of converged instances and shifted geometric means of iterations and solve times (in milliseconds).

		iterations			solve time		
step	conv	every	this	all	every	this	all
basic	371	101.3	100.9	102.4	2131	2207	2282
prox	369	64.7	65.3	67.2	1317	1390	1451
TOA	374	35.0	35.3	36.1	1014	1063	1103
curve	372	29.7	30.0	31.0	742	781	820
comb	367	18.3	18.6	20.0	624	656	706

Each enhancement, by design, changes one modular component or aspect of the stepping procedure. Below, we examine the impact of our algorithmic choices by discussing pairwise comparisons of consecutive stepping procedures.

Figure 1.2: Overlayed histograms of iteration count (left, log scale) and solve time (right, log scale, in seconds) for the *basic* and *comb* stepping procedures, excluding instances that fail to converge.

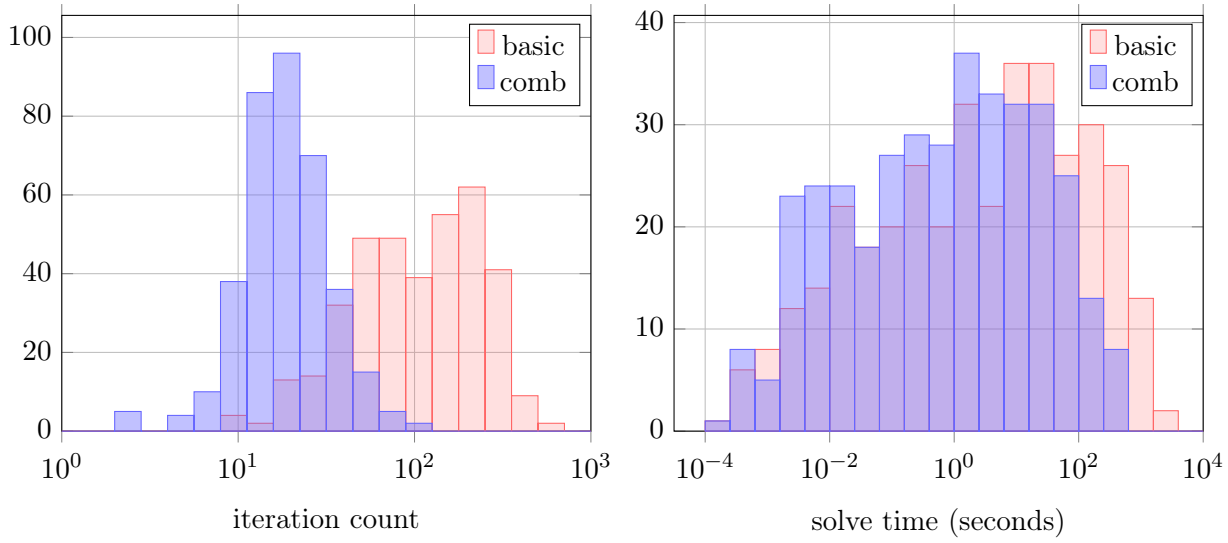


Figure 1.3: Iteration count against instance barrier parameter for the *basic* (left) and *comb* (right) stepping procedures, excluding instances that fail to converge.

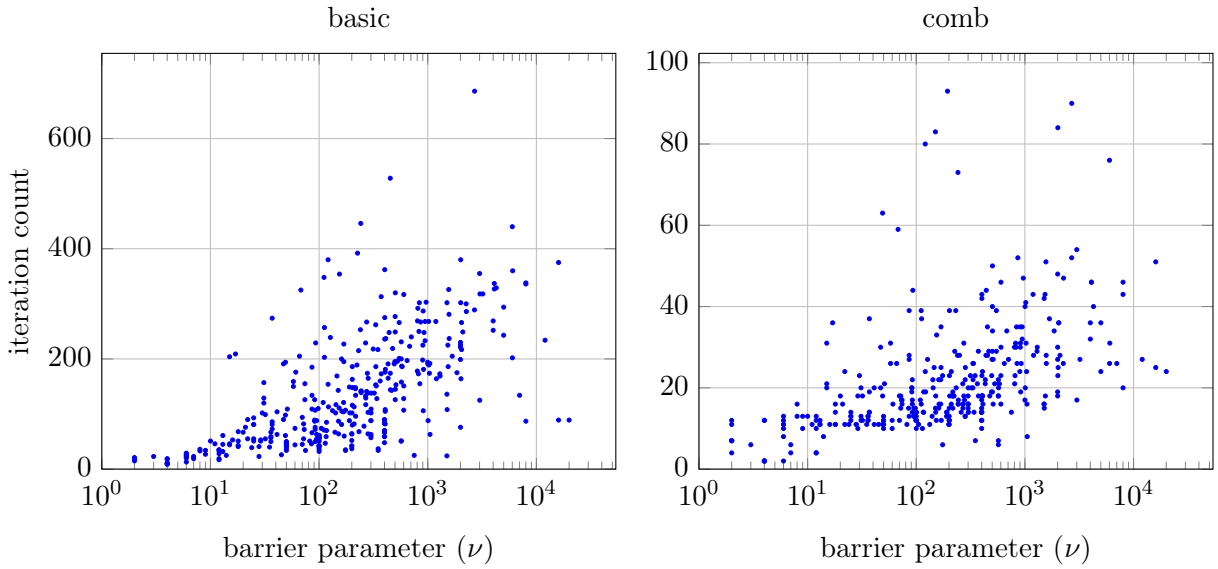


Figure 1.4: Performance profiles (see Section 1.9.2) of iteration count (left) and solve time (right) for the four stepping enhancements overall.

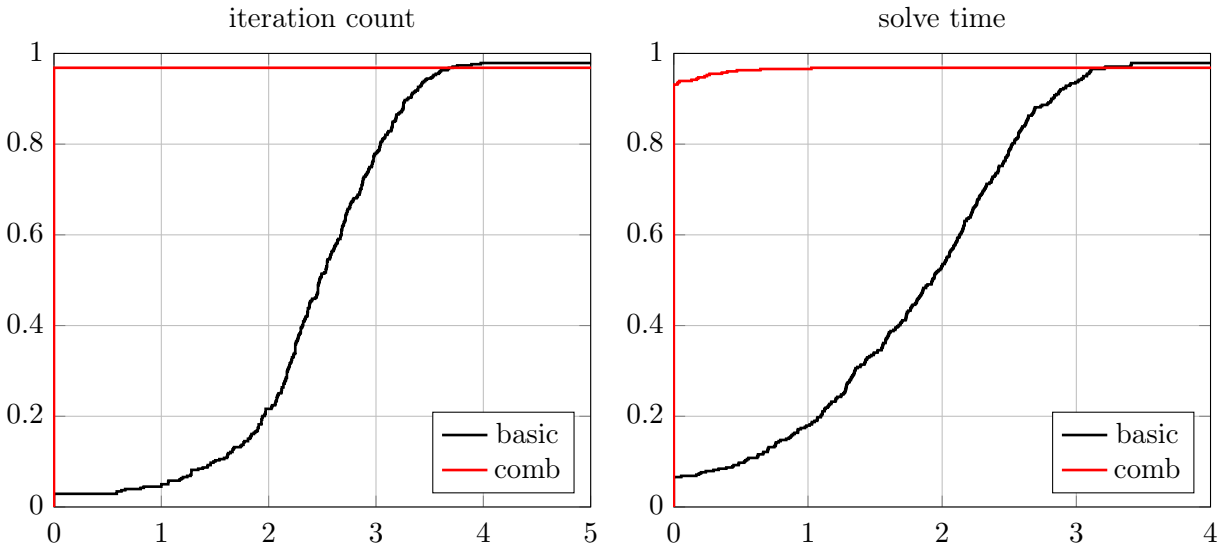


Figure 1.5: Solve time (log scale, in seconds) for the *comb* stepping procedure against (left) instance size (log scale) and (right) the proportion of solve time spent in *RHS*, excluding instances that fail to converge.

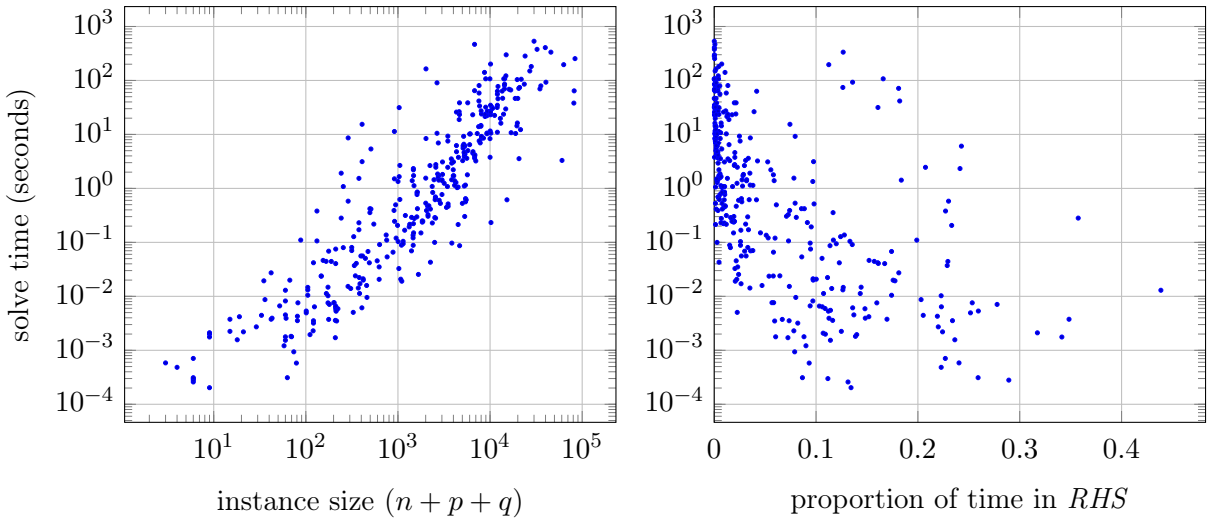


Table 1.4: For each stepping procedure, the shifted geometric means of subtimings (in milliseconds) for the key algorithmic components.

			total				per iteration			
set	step	init	LHS	RHS	direc	search	LHS	RHS	direc	search
every	basic	29.5	741	1.45	75.6	125.6	7.73	0.02	0.81	1.29
	prox	29.5	486	1.11	50.3	67.7	7.88	0.02	0.84	1.10
	TOA	29.4	285	10.96	52.2	73.3	8.28	0.32	1.53	2.14
	curve	29.6	244	9.24	44.6	33.4	8.33	0.32	1.53	1.15
	comb	29.3	160	10.51	57.6	35.2	8.74	0.58	3.14	1.94
this	basic	30.3	784	1.48	78.8	131.8	8.20	0.02	0.85	1.36
	prox	30.1	519	1.12	53.4	72.6	8.35	0.02	0.88	1.16
	TOA	30.2	302	11.97	55.4	78.3	8.70	0.35	1.61	2.26
	curve	30.5	261	9.99	47.3	35.1	8.80	0.34	1.60	1.20
	comb	30.5	171	11.03	60.7	36.4	9.23	0.60	3.27	1.98
all	basic	31.1	814	1.62	82.7	134.7	8.52	0.02	0.91	1.40
	prox	31.3	549	1.25	56.2	75.1	8.74	0.02	0.94	1.20
	TOA	31.2	317	12.23	57.5	79.7	9.04	0.36	1.66	2.28
	curve	31.4	276	10.40	49.6	37.3	9.17	0.36	1.68	1.26
	comb	31.4	188	11.88	64.2	40.0	9.66	0.63	3.35	2.10

Figure 1.6: Relative improvement, from *basic* to *comb*, in iteration count (left) or solve time (right) against iteration count or solve time (in seconds) respectively for *comb*, over the 356 instances on which both *basic* and *comb* converge.

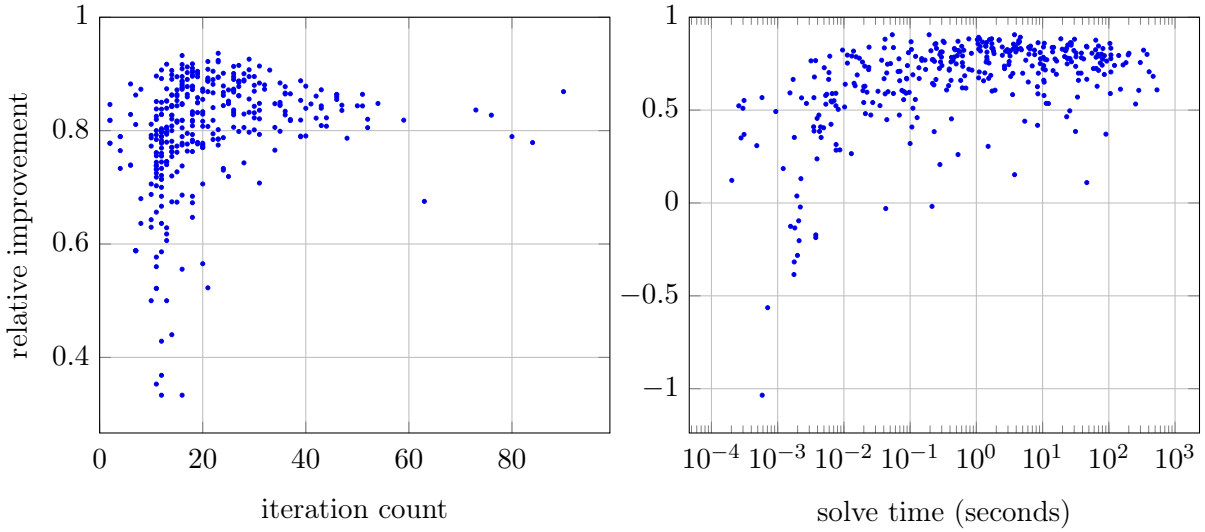
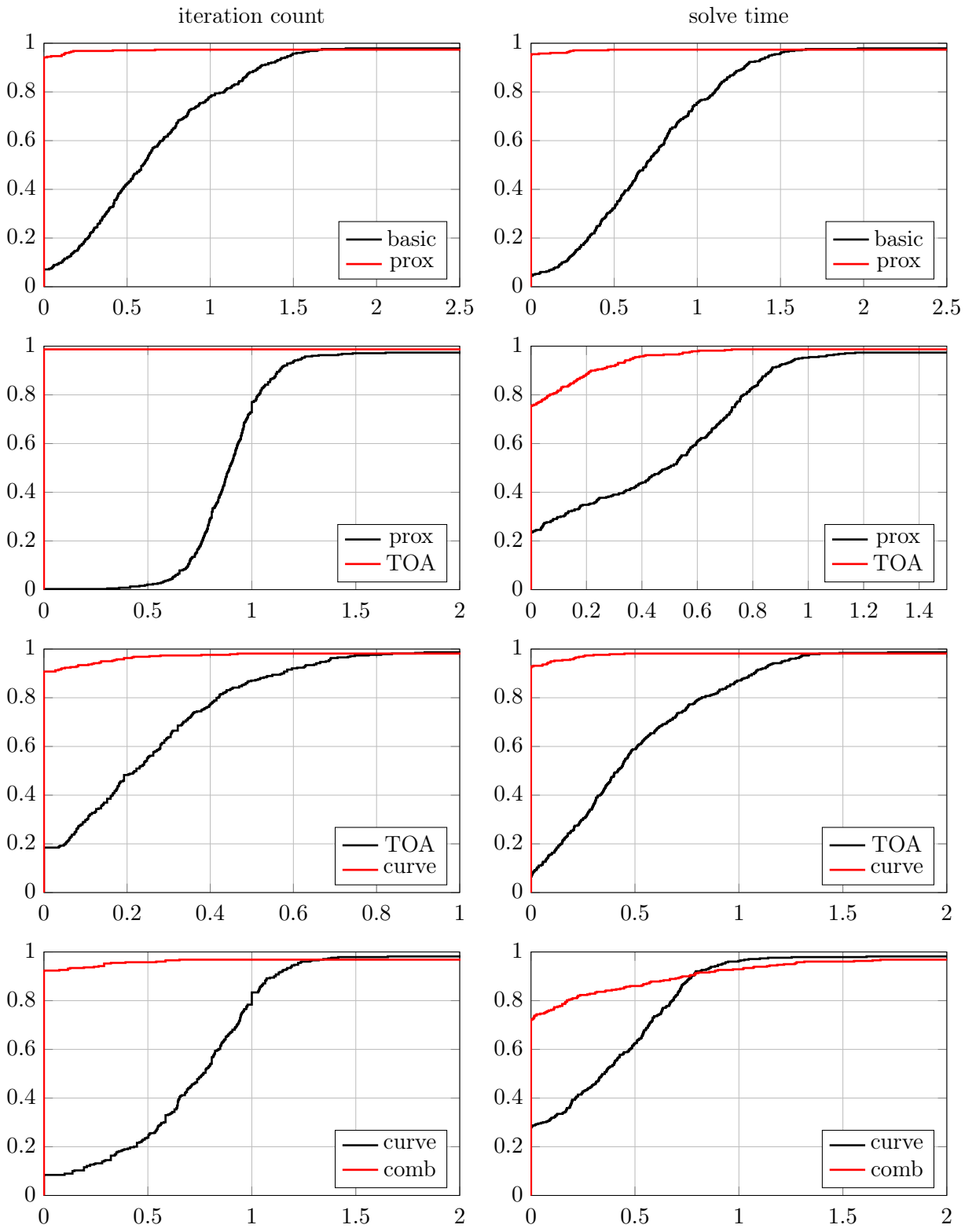


Figure 1.7: Performance profiles (see Section 1.9.2) of iteration count (left column) and solve time (right column) for the four stepping enhancements (rows).



1.9.3.1 Less restrictive proximity

We compare *basic* and *prox* to evaluate the central path proximity enhancement introduced in Section 1.5.5.2. Figure 1.7 (first row) shows that the iteration count and solve time improve for nearly all instances. From Table 1.3, the shifted geometric means of iteration count and solve time improve by over 35%.

The similarity between the iteration count and solve time performance profiles in Figure 1.7 and also between the per iteration subtimings in Table 1.4 suggests that the solve time improvement is driven mainly by the reduction in iteration count. The per iteration *search* time decreases slightly, since on average fewer backtracking search steps are needed per iteration for *prox* (because it tends to step further in the prediction directions, as evidenced by the smaller iteration counts). These results suggest that the central path proximity restrictions in the algorithms by Skajaa and Ye (2015) and Papp and Yıldız (2021) are too conservative from the perspective of practical performance, and that we need not restrict iterates to a very small neighborhood of the central path in order to obtain high quality prediction directions in practice.

1.9.3.2 Third order adjustments

We compare *prox* and *TOA* to evaluate the TOA enhancement introduced in Section 1.5.5.3. Figure 1.7 (second row) shows that the iteration count improves for all instances and by a fairly consistent magnitude, and the solve time improves for nearly 80% of instances. From Table 1.3, the shifted geometric means of iteration count and solve time improve by over 45% and over 20% respectively.

Since *TOA* computes an additional direction and performs an additional backtracking search every iteration, the per iteration times for *direc* and *search* in Table 1.4 nearly double. The *RHS* time increases substantially, because the TOO is evaluated for the second RHS vector (used to compute the TOA direction), but *RHS* is still much faster than the other components. Per iteration, *direc* and *search* also remain fast compared to *LHS*. We see an overall solve time improvement because the reduction in iteration count usually outweighs the additional cost at each iteration. This suggests that the TOO is generally relatively cheap to compute, and our TOA approach very reliably improves the quality of the search directions.

1.9.3.3 Curve search

We compare *TOA* and *curve* to evaluate the curve search enhancement introduced in Section 1.5.5.4. Figure 1.7 (third row) shows that the iteration count and solve time improve for most instances, with larger and more consistent improvements for the solve time. From Table 1.3, the shifted geometric means of iteration count and solve time improve by over 15% and over 25% respectively.

Since *curve* performs one backtracking search along a curve instead of the two backtracking line searches needed by *TOA*, the per iteration *search* time in Table 1.4 nearly halves. The other subtimings are unaffected, so *curve* improves the speed of each iteration. The improvement in

iteration count may stem from the more dynamic nature of the curve search compared to *TOA*'s approach of computing a fixed combination of the unadjusted and TOA directions as a function of the step distance in the unadjusted direction.

1.9.3.4 Combined directions

Finally, we compare *curve* and *comb* to evaluate the combined directions enhancement introduced in Section 1.5.5.5. Figure 1.7 (fourth row) shows that the iteration count and solve time improve on around 90% and 70% of instances respectively. From Table 1.3, the shifted geometric means of iteration count and solve time improve by nearly 40% and over 15% respectively.

Since *comb* computes four directions per iteration (unadjusted and TOA directions for both prediction and centering) instead of two, the per iteration times for *RHS* and *direc* approximately double in Table 1.4. The *search* time increases because on average more backtracking curve search steps are needed per iteration (for *curve*, the centering phase typically does not require multiple backtracking steps). Per iteration, *LHS* remains slower than the other components combined. Hence combining the prediction and centering phases generally improves practical performance, and should be more helpful when *LHS* is particularly expensive (such as when $n - p$, the side dimension of the PSD matrix we factorize during *LHS*, is large; see Section 1.6). Furthermore, Figure 1.5 (right) shows that for most instances, *RHS* accounts for a small proportion of the overall solve time for *comb*, especially for instances that take longer to solve. This suggests that the TOO is rarely a bottleneck for our *comb* stepping procedure.

Chapter 2

Solving natural conic formulations

Abstract

Many convex optimization problems can be represented through conic *extended formulations* (EFs) using only the small number of *standard cones* recognized by advanced conic solvers such as MOSEK 9. However, EFs are often significantly larger and more complex than equivalent conic *natural formulations* (NFs) represented using exotic cones supported by Hypatia. For three broad classes of cones in Hypatia, we describe general techniques for building EFs and compare computational properties of the NFs and EFs. We formulate seven applied problems over these cones, noting that the EFs are necessarily larger, more complex, and less efficient to build. Our computational experiments demonstrate the advantages, especially in terms of solve time and memory usage, of solving the NFs with Hypatia compared to solving the EFs with either Hypatia or MOSEK 9. A key reason for Hypatia's success is our implementation of efficient and numerically stable cone oracles. We derive oracles for the first two classes - the *positive semidefinite slice cones* and the *infinity/spectral norm cones* - in this chapter, and for the third class - the *spectral function cones* - in Chapter 3. A surprising and important result is an analytic formula for the inverse Hessian operator of the spectral norm cone barrier function, which avoids the need to form and factorize an explicit Hessian matrix.

2.1 Introduction

Historically, IPM solvers were based on efficient algorithms specialized for symmetric cones, in particular, the nonnegative, (rotated) second order, and positive semidefinite (PSD) cones. However, many useful nonsymmetric conic constraints (such as $u \leq \log(w)$, representable with an exponential cone) are not exactly representable with symmetric cones. Skajaa and Ye (2015) (henceforth referred to as SY) introduced a nonsymmetric conic interior point method (IPM) that requires just a few oracles for the primal cone only. In Chapter 1, we describe an IPM that enhances the practical performance of SY. Our algorithm recognizes any exotic cone, which we define as a proper cone for which we can implement a small set of easily computable (i.e. fast, numerically stable, analytic) oracles for a logarithmically homogeneous self-concordant barrier (LHSCB) function for the cone or for its dual cone (not both). We implement this algorithm in our IPM solver, Hypatia, introduced in

Section 1.2.

2.1.1 Natural and extended formulations

Although advanced conic solvers such as MOSEK 9 currently recognize at most only a handful of *standard cones* (the common symmetric cones and the three-dimensional nonsymmetric exponential and power cones), these cones are sufficient for representing many problems of interest (Lubin, Yamangil, et al., 2016; MOSEK ApS, 2020a). Modeling tools such as disciplined convex programming (DCP) packages (see CVX (Grant and Boyd, 2014), CVXPY (Diamond and Boyd, 2016), and Convex.jl (Udell et al., 2014)) and MathOptInterface's bridges (Benoit Legat et al., 2020) are designed to facilitate transformations of convex problems into conic problems with standard cones. However, a representation limited to the standard cones is often not the most natural or efficient conic formulation. The process of transforming a general conic problem into a conic *extended formulation* (EF) that uses only standard cones often requires introducing many artificial variables, linear equalities, and/or higher-dimensional conic constraints.

If conic solvers could recognize the much larger class of exotic cones, they could directly solve simpler, smaller conic *natural formulations* (NFs). NFs have several practical advantages over EFs. NFs tend to be easier to model, and converting conic certificates from the space of the EF back into the more meaningful NF space can be complicated. Furthermore, the convergence conditions used by IPMs can provide numerical guarantees about conic certificates, but if EF certificates are converted to NF space, the NF certificates might lack such guarantees. Perhaps more importantly, by increasing the size and complexity of problem data, EFs can increase the computational cost of, for example, preprocessing and linear system solving at each iteration. This raises the question of whether it can be more efficient to solve NFs using a generic conic solver than to solve equivalent EFs using an advanced conic solver specialized for standard cones.

In the particular context of polynomial weighted sum-of-squares (SOS) optimization, Papp and Yıldız (2019) illustrate the advantages of solving SOS cone formulations instead of equivalent but much larger semidefinite programming EFs. The authors describe easily computable LHSCB oracles for dual SOS cones, noting that easily computable oracles are not known for primal SOS cones. They show that their SOS NF-based approach has lower theoretical time and space complexity overall compared to a standard EF-based semidefinite programming method. After implementing SY in the MATLAB solver Alfonso (Papp and Yıldız, 2017; Papp and Yıldız, 2021), the authors observe improved solve times and scaleability from solving the NFs with Alfonso compared to solving the EFs with MOSEK.

To broaden the computational argument for NFs, in Section 2.2 we define a variety of exotic cone types in Hypatia. We group these cones into three classes: the PSD slice cones in Section 2.2.1, the infinity/spectral norm cones in Section 2.2.2, and the spectral function cones in Section 2.2.3. We describe general techniques for constructing equivalent EFs of NF constraints involving these cones, and we analyze how these EF techniques necessarily increase formulation size, which impacts the dimensions of the linear systems that must be solved by an IPM (see Section 1.6). We also observe

that the EFs are often associated with larger values of the LHSCB parameter ν. This parameter impacts the number of iterations $\mathcal{O}(\sqrt{\nu}\log(1/\varepsilon))$ needed in the worst case by an idealized IPM such as SY to obtain a solution within ε tolerance (Nesterov, Todd, and Ye, 1997), though since most performance-oriented conic solvers do not directly implement idealized IPMs, the practical impact of the parameter ν is unclear.

2.1.2 Examples and computational testing

In Section 2.3, we present a series of example problems from applications such as matrix completion, experiment design, and smooth density estimation. For these examples, we describe simple NFs using the exotic cones we define in Section 2.2. Some of these NFs are new and may be valuable to try in real-world applications. We randomly generate NF instances of a wide variety of sizes, construct equivalent EFs using the general EF techniques in Section 2.2, and observe that the EFs are significantly larger. For example, in our density estimation example problem in Section 2.3.6, these dimensions are typically orders of magnitude larger for the EFs than for the NFs.

Our computational experiments demonstrate significant improvements in solve time and memory overhead from solving the NFs with Hypatia compared to solving the EFs with Hypatia or MOSEK 9. Our experience also suggests that since EFs are often larger and more complex than NFs, they can be less convenient for the modeler, and noticeably slower and more memory-intensive to construct using JuMP or Hypatia's native interface. For many instances, we could build the NF efficiently, but we hit time or memory limits while constructing the EF. We discuss these results and our conclusions further in Section 2.4. In Section 3.8, we present more applied examples over the spectral function cones and our computational experiments lead to the same conclusions.

2.1.3 Efficient oracle procedures

In the final two sections of this chapter, we discuss oracles needed by Hypatia's IPM (see Section 1.3) for two of the three broad cone classes. The third class - the spectral function cones - are addressed in detail in Chapter 3, where we also develop new LHSCBs with near-optimal barrier parameters.

In Section 2.5, we define LHSCBs and derive efficient, numerically stable procedures for gradient, Hessian product, and third order directional derivative oracles for the PSD slice cones, which can be characterized as intersections of slices of the PSD cone. We consider the linear matrix inequality (LMI) cone $\mathcal{K}_{\text{LMI}}$ in Section 2.5.1 and the dual SOS and SOS matrix cones $\mathcal{K}^*_{\text{SOS}}$ and $\mathcal{K}^*_{\text{matSOS}}$ in Section 2.5.2. In Section 2.5.3, we consider the sparse PSD cone $\mathcal{K}_{\text{sPSD}}$, obtaining the third order oracle by differentiating a Hessian product procedure described by M. S. Andersen, Dahl, and Vandenberghe (2013).

In Section 2.6, we discuss the infinity/spectral norm cones, which are epigraphs of real/complex vector ℓ_∞ norms or matrix spectral norms (for symmetric/Hermitian or real/complex rectangular matrices). Our LHSCBs and oracles for this class are all specializations of the most general case, the rectangular matrix spectral norm cone $\mathcal{K}_{\ell_{\text{spec}}}$, so we focus on this case. We use a thin singular value decomposition of the matrix to check feasibility and efficiently compute gradients, Hessian products,

and third order directional derivatives. Importantly, we also derive a closed form expression for the inverse Hessian product, which allows Hypatia to avoid ever forming or factorizing an explicit Hessian matrix. This formula improves the speed, memory requirements, and numerical performance of inverse Hessian products, as we demonstrate in comparisons against a naive explicit Hessian factorization approach.

2.2 Cones and extended formulations

Let $\mathcal{K} \subset \mathbb{R}^q$ be a proper cone, i.e. a closed, convex, pointed, and full-dimensional conic set. We call $\mathcal{K}$ a primitive (or irreducible) cone if it cannot be written as a Cartesian product of two or more lower-dimensional cones. The dual cone $\mathcal{K}^*$ is a primitive proper cone if and only if $\mathcal{K}$ is a primitive proper cone.

As we discuss in Section 1.3, Hypatia's generic cone interface allows defining any primitive proper cone $\mathcal{K}$ by specifying a small list of oracles: an initial interior point $t \in \text{int}(\mathcal{K})$, a feasibility test for $\text{int}(\mathcal{K})$, and gradient, Hessian, and third order directional derivative evaluations for an LHSCB f for $\mathcal{K}$. The cone interface also allows optional specification of other oracles that can improve performance, such as dual cone feasibility checks and inverse Hessian product oracles. Once $\mathcal{K}$ is defined through Hypatia's cone interface, both $\mathcal{K}$ and $\mathcal{K}^*$ may be used in any combination with other recognized cones to construct the Cartesian product cone in the primal conic form (2.1) below.

Hypatia's primal conic form over variable $x \in \mathbb{R}^n$ is:

$$\inf_x \quad c'x : \tag{2.1a}$$
$$b - Ax = 0, \tag{2.1b}$$
$$h - Gx \in \mathcal{K}, \tag{2.1c}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^p$, and $h \in \mathbb{R}^q$ are vectors, $A : \mathbb{R}^n \to \mathbb{R}^p$ and $G : \mathbb{R}^n \to \mathbb{R}^q$ are linear maps, and $\mathcal{K} \subset \mathbb{R}^q$ is a Cartesian product $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K$ of primitive proper cones. Henceforth we use n, p, q to denote the variable, equality, and conic constraint dimensions of a conic problem in this form.

As we discuss in Section 2.1.1, advanced conic solvers such as MOSEK 9 currently only recognize a handful of standard cones, which we now define. The nonnegative cone is $\mathcal{K}_{\geq} := \mathbb{R}_{\geq}$. The Euclidean norm cone (or second order cone) $\mathcal{K}_{\ell_2}$, Euclidean norm square cone (or rotated second order cone) $\mathcal{K}_{\text{sqr}}$, and vectorized PSD cone $\mathcal{K}_{\succeq}$ are:

$$\mathcal{K}_{\ell_2} := \{(u, w) \in \mathbb{R}_{\geq} \times \mathbb{R}^d : u \geq \|w\|\}, \tag{2.2a}$$
$$\mathcal{K}_{\text{sqr}} := \{(u, v, w) \in \mathbb{R}_{\geq} \times \mathbb{R}_{\geq} \times \mathbb{R}^d : 2uv \geq \|w\|^2\}, \tag{2.2b}$$
$$\mathcal{K}_{\succeq} := \left\{w \in \mathbb{R}^{\text{sd}(d)} : \text{mat}(w) \in \mathbb{S}^d_{\succeq}\right\}. \tag{2.2c}$$

These cones are symmetric (self-dual), i.e. $\mathcal{K} = \mathcal{K}^*$. The exponential cone in $\mathbb{R}^3$ is a special case of our logarithm cone $\mathcal{K}_{\log}$ defined in Section 2.2.3.2 (let $d = 1$ in (2.19)). The power cone in

$\mathbb{R}^3$ (MOSEK ApS, 2020a, Section 4.1) is a special case of Hypatia's generalized power cone $\mathcal{K}_{\text{gpow}}$ (defined in Section 1.8); we omit the definition here because we do not need power cones for the EFs in this chapter. The three-dimensional exponential and power cones are nonsymmetric.

DCP modeling tools and MathOptInterface's *bridges* (Benoit Legat et al., 2020, Section 5) automatically construct EFs to enable access to standard conic solvers. In Sections 2.2.1 to 2.2.3, we define three broad classes of cones supported by Hypatia and describe techniques for constructing EFs for conic problems over these cones. These techniques follow best practices from DCP software and descriptions such as Ben-Tal and Nemirovski (2001, Chapter 4). For convenience, in this section we refer to a particular exotic cone constraint as an NF, and an equivalent reformulation of such a constraint in terms of only standard cones as an EF. In general, an NF constraint has the form $h - Gx \in \mathcal{K}$, but for simplicity in this section we write $s \in \mathcal{K}$, since $s = h - Gx$ can be substituted into the EF description.

An EF may use auxiliary variables, linear equalities, and/or conic constraints, which affect the dimensions n, p, and q (respectively) of the primal conic form (2.1). In Table 2.1, we compare the dimensions and LHSCB parameters (ν) associated with the equivalent NFs and EFs in this section. Clearly, the EFs are associated with larger dimensions, which may impact the performance of IPMs, for example by increasing the size of linear systems that must be solved at each iteration. For some EFs with auxiliary variables and equalities, it is possible to perform eliminations to reduce certain dimensions, but this can impact the sparsity of problem data (note that for our experiments, Hypatia and MOSEK both perform preprocessing). Furthermore, the barrier parameters of the EFs are at least as large as those of the NFs; recall from Section 2.1.1 that ν impacts the worst-case number of iterations needed for idealized IPMs to converge. We contributed the EFs described in Sections 2.2.2.1, 2.2.2.2 and 2.2.3.1 to MathOptInterface's bridges.

2.2.1 Positive semidefinite slice cones

For each cone $\mathcal{K}$ in this class, either $\mathcal{K}$ or $\mathcal{K}^*$ has an obvious representation as an intersection of slices of the PSD cone, i.e. there exists an EF in terms of $\mathcal{K}_{\succeq}$ cones without auxiliary variables. We only describe a handful of Hypatia's PSD slice cones in this section. We summarize properties of the NFs and EFs in Table 2.1 (top). In Section 2.5, we define LHSCBs for this class and describe oracle procedures.

2.2.1.1 Hermitian positive semidefinite cone

The self-dual vectorized Hermitian PSD cone $\mathcal{K}_{c\succeq}$ is closely related to the self-dual vectorized real symmetric PSD cone $\mathcal{K}_{\succeq}$:

$$\mathcal{K}_{c\succeq} := \left\{ w \in \mathbb{R}^{d^2} : \text{mat}(w) \in \mathbb{H}^d_{\succeq} \right\}. \tag{2.3}$$

For side dimension d, this cone is representable with a standard $\mathcal{K}_{\succeq}$ constraint of side dimension $2d$, since:

$$W \in \mathbb{H}^d_{\succeq} \quad \Leftrightarrow \quad \begin{bmatrix} \Re(W) & -\Im(W) \\ \Im(W) & \Re(W) \end{bmatrix} \in \mathbb{S}^{2d}_{\succeq}. \tag{2.4}$$

Table 2.1: Properties of NFs and EFs for the PSD slice cones (top, see Section 2.2.1), infinity/spectral norm cones (middle, see Section 2.2.2), and spectral function cones (bottom, see Section 2.2.3). q and ν are the dimension and LHSCB parameter for the NF cone, and $\bar{q}$ and $\bar{\nu}$ are the corresponding values for the EF Cartesian product cone. $\bar{n}$ and $\bar{p}$ are the EF auxiliary variable and equality dimensions. Recall $\mathrm{sd}(d) \coloneqq d(d+1)/2$.

	NF		EF			
cone	q	ν	$\bar{q}$	$\bar{\nu}$	$\bar{n}$	$\bar{p}$
$\mathcal{K}_{c\succeq}$	d^2	d	$\mathrm{sd}(2d)$	$2d$		
$\mathcal{K}_{\mathrm{sPSD}}$	d	s	$\mathrm{sd}(s)$	ν		
$\mathcal{K}^*_{\mathrm{sPSD}}$	–"–	–"–	–"–	ν	$\mathrm{sd}(s)-d$	
$\mathcal{K}^*_{\mathrm{SOS}}$	d	$\sum_l s_l$	$\sum_l \mathrm{sd}(s_l)$	ν		
$\mathcal{K}_{\mathrm{SOS}}$	–"–	–"–	–"–	ν	$\bar{q}$	d
$\mathcal{K}^*_{\mathrm{matSOS}}$	$\mathrm{sd}(t)d$	$t\sum_l s_l$	$\sum_l \mathrm{sd}(ts_l)$	ν		
$\mathcal{K}_{\mathrm{matSOS}}$	–"–	–"–	–"–	ν	$\bar{q}$	q
$\mathcal{K}_{\ell_\infty}$	$1+d$	$1+d$	$2d$	$2d$		
$\mathcal{K}^*_{\ell_\infty}$	–"–	–"–	$1+2d$	$1+2d$	$2d$	d
$\mathcal{K}_{\ell_{\mathrm{sspec}}}$	$1+\mathrm{sd}(d)$	$1+d$	$2\,\mathrm{sd}(d)$	$2d$		
$\mathcal{K}^*_{\ell_{\mathrm{sspec}}}$	–"–	–"–	$1+2\,\mathrm{sd}(d)$	$1+2d$	$2\,\mathrm{sd}(d)$	$\mathrm{sd}(d)$
$\mathcal{K}_{\ell_{\mathrm{spec}}}$	$1+ds$	$1+d$	$\mathrm{sd}(d+s)$	$d+s$		
$\mathcal{K}^*_{\ell_{\mathrm{spec}}}$	–"–	–"–	$1+\mathrm{sd}(d+s)$	$1+d+s$	$\mathrm{sd}(d)+\mathrm{sd}(s)$	
$\mathcal{K}_{\mathrm{geo}}$	$1+d$	$1+d$	$2+3d$	$2+3d$	$1+d$	
$\mathcal{K}_{\mathrm{rtdet}}$	$1+\mathrm{sd}(d)$	–"–	$2+3d+\mathrm{sd}(2d)$	$2+5d$	$1+d+\mathrm{sd}(d)$	
$\mathcal{K}_{\log}$	$2+d$	$2+d$	$1+3d$	$1+3d$	d	
$\mathcal{K}_{\mathrm{logdet}}$	$2+\mathrm{sd}(d)$	–"–	$1+3d+\mathrm{sd}(2d)$	$1+5d$	$1+d+\mathrm{sd}(d)$	
$\mathcal{K}_{\mathrm{sep}}(\mathbb{R})$	$2+d$	–"–	$1+3d$	$1+3d$	d	
$\mathcal{K}_{\mathrm{sep}}(\mathbb{S})$	$2+\mathrm{sd}(d)$	–"–	$-1+4d+d^3$	$-1+3d+2d^2$	$-1+(5d+d^3)/2$	1
$\mathcal{K}_{\mathrm{sep}}(\mathbb{H})$	$2+d^2$	–"–	$-1+3d-2d^2+4d^3$	$-1+d+4d^2$	$-1+3d-d^2+d^3$	1

2.2.1.2 Sparse positive semidefinite cone

Suppose $\mathcal{S} = ((i_l, j_l))_{l \in [\![d]\!]}$ is a collection of row-column index pairs defining the sparsity pattern (including all diagonal elements) of the lower triangle of a symmetric matrix of side dimension s. Unlike prior work such as Burer (2003) and M. S. Andersen, Dahl, and Vandenberghe (2013), we do not require that $\mathcal{S}$ be a chordal sparsity pattern, hence the cone dimension $d = |\mathcal{S}|$ can be as small as possible. Note $s \leq d \leq \mathrm{sd}(s)$. Let $\mathrm{mat}_{\mathcal{S}} : \mathbb{R}^d \to \mathbb{S}^s$ be the linear operator satisfying:

$$(\mathrm{mat}_{\mathcal{S}}(w))_{i,j} = \begin{cases} w_l & \text{if } i = i_l = j = j_l, \\ w_l/\sqrt{2} & \text{if } i = i_l \neq j = j_l, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i, j \in [\![s]\!] : i \geq j. \tag{2.5}$$

We define the sparse PSD cone and its dual cone of PSD-completable matrices as:

$$\mathcal{K}_{\mathrm{sPSD}(\mathcal{S})} \coloneqq \left\{ w \in \mathbb{R}^d : \mathrm{mat}_{\mathcal{S}}(w) \in \mathbb{S}^s_{\succeq} \right\}, \tag{2.6a}$$

$$\mathcal{K}^*_{\mathrm{sPSD}(\mathcal{S})} \coloneqq \left\{ w \in \mathbb{R}^d : \exists \theta \in \mathbb{R}^{\mathrm{sd}(s)-d}, \mathrm{mat}_{\mathcal{S}}(w) + \mathrm{mat}_{\bar{\mathcal{S}}}(\theta) \in \mathbb{S}^s_{\succeq} \right\}, \tag{2.6b}$$

where $\bar{\mathcal{S}}$ is the lower triangle inverse sparsity pattern of $\mathcal{S}$ (with $|\bar{\mathcal{S}}| = \mathrm{sd}(s) - d$).

Note Hypatia also supports the complex Hermitian sparse PSD cone (and its dual cone), but since this a simple generalization of $\mathcal{K}_{\mathrm{sPSD}}$, we do not define it here. The corresponding EFs generalize (2.6) to the Hermitian case and then apply the $\mathcal{K}_{c\succeq}$ EF (2.4).

2.2.1.3 Polynomial weighted sum-of-squares cones

Given a collection of matrices $P_l \in \mathbb{R}^{d \times s_l}, \forall l \in [\![r]\!]$ from basis polynomials evaluated at d interpolation points as in Papp and Yıldız (2019), the interpolant basis polynomial weighted sum-of-squares (SOS) cone and its dual are:

$$\mathcal{K}_{\mathrm{SOS}(P)} \coloneqq \left\{ w \in \mathbb{R}^d : \exists \Theta_l \in \mathbb{S}^{s_l}_{\succeq}, \forall l \in [\![r]\!], w = \textstyle\sum_{l \in [\![r]\!]} \mathrm{diag}(P_l \Theta_l P_l') \right\}, \tag{2.7a}$$

$$\mathcal{K}^*_{\mathrm{SOS}(P)} \coloneqq \left\{ w \in \mathbb{R}^d : P_l' \,\mathrm{Diag}(w) P_l \in \mathbb{S}^{s_l}_{\succeq}, \forall l \in [\![r]\!] \right\}. \tag{2.7b}$$

SOS cones are useful for polynomial and moment modeling; for example, a point in $\mathcal{K}_{\mathrm{SOS}(P)}$ corresponds to a polynomial that is pointwise nonnegative on a semialgebraic domain defined by P. Note Hypatia also defines a new real-valued complex polynomial SOS (or Hermitian SOS) cone with complex P_l matrices.

Given a side dimension t of a symmetric matrix of polynomials (for simplicity, all using the same interpolant basis), and $P_l \in \mathbb{R}^{d \times s_l}, \forall l \in [\![r]\!]$ defined as for $\mathcal{K}_{\mathrm{SOS}(P)}$ in Section 2.2.1.3, the SOS matrix cone and its dual are:

$$\mathcal{K}_{\mathrm{matSOS}(P)} \coloneqq \left\{ \begin{array}{l} w \in \mathbb{R}^{\mathrm{sd}(t)d} : \exists \Theta_l \in \mathbb{S}^{s_l t}_{\succeq}, \forall l \in [\![r]\!], \\ W_{i,j,:} = \sum_{l \in [\![r]\!]} \mathrm{diag}(P_l (\Theta_l)_{i,j} P_l'), \forall i, j \in [\![t]\!] : i \geq j \end{array} \right\}, \tag{2.8a}$$

$$\mathcal{K}^*_{\mathrm{matSOS}(P)} \coloneqq \left\{ w \in \mathbb{R}^{\mathrm{sd}(t)d} : [P_l' \,\mathrm{Diag}(W_{i,j,:}) P_l]_{i,j \in [\![t]\!]} \in \mathbb{S}^{s_l t}_{\succeq}, \forall l \in [\![r]\!] \right\}, \tag{2.8b}$$

where $W_{i,j,:} \in \mathbb{R}^d$ is the contiguous slice of w (scaled to account for symmetry) corresponding to the interpolant basis values in the (i,j)th position of the symmetric matrix, $(S)_{i,j}$ is the (i,j)th block in a symmetric matrix S with square blocks of equal dimensions, and $[g(W_{i,j,:})]_{i,j \in [\![t]\!]}$ is the symmetric matrix with square matrix $g(W_{i,j,:})$ in the (i,j)th block. A point in $\mathcal{K}_{\text{matSOS}(P)}$ corresponds to a polynomial matrix that is pointwise PSD on a semialgebraic domain defined by P. See Kapelevich, Coey, and Vielma (2021) for more details.

Our examples in Sections 2.3.5 to 2.3.7 use the EFs implicit in the definitions of $\mathcal{K}^*_{\text{SOS}}$, $\mathcal{K}_{\text{SOS}}$, and $\mathcal{K}_{\text{matSOS}}$ in (2.7a), (2.7b) and (2.8a). These EFs each use r $\mathcal{K}_{\geq}$ cones.

Papp and Yıldız (2019) describe an LHSCB with analytic oracles for $\mathcal{K}^*_{\text{SOS}(P)}$, but they state that one is not known for $\mathcal{K}_{\text{SOS}(P)}$. We are not aware of an LHSCB with analytic oracles for $\mathcal{K}_{\text{matSOS}(P)}$ - indeed, for $t = 1$, $\mathcal{K}_{\text{matSOS}(P)}$ reduces to $\mathcal{K}_{\text{SOS}(P)}$. Noting that (2.8b) implicitly constrains a linear function of w to a Cartesian product of PSD cones, we can use Nesterov and Nemirovski (1994, Propositions 5.1.1 and 5.1.3) to derive an LHSCB for $\mathcal{K}^*_{\text{matSOS}(P)}$; see Section 2.5.2 and Kapelevich, Coey, and Vielma (2021) for details. For $t = 1$, this new LHSCB reduces to that of $\mathcal{K}_{\text{SOS}(P)}$.

2.2.2 Infinity/spectral norm cones

For each cone $\mathcal{K}$ in this class, either $\mathcal{K}$ or $\mathcal{K}^*$ is an epigraph of the vector ℓ_∞ norm or matrix spectral norm. Although these cones are technically PSD slice cones, the structures and LHSCBs for these cones are so closely related that we treat this class separately. We summarize properties of the NFs and EFs in Table 2.1 (middle). In Section 2.6, we define LHSCBs and derive new oracle procedures.

2.2.2.1 Vector infinity norm cones

The ℓ_∞ norm cone is the epigraph of ℓ_∞. Its dual cone is the epigraph of the ℓ_1 norm, which is the dual norm of ℓ_∞. For the real vector domain, these cones are defined as:

$$\mathcal{K}_{\ell_\infty} := \{(u, w) \in \mathbb{R}_{\geq} \times \mathbb{R}^d : u \geq \|w\|_\infty\}, \tag{2.9a}$$

$$\mathcal{K}^*_{\ell_\infty} := \{(u, w) \in \mathbb{R}_{\geq} \times \mathbb{R}^d : u \geq \|w\|_1\}. \tag{2.9b}$$

For $\mathcal{K}_{\ell_\infty}$, we use the LHSCB from Güler (1996, Section 7.5). We are not aware of an LHSCB for $\mathcal{K}^*_{\ell_\infty}$ with analytic oracles.

Our examples in Sections 2.3.1 and 2.3.4 use the following linear (LP) EFs (right):

$$(u, w) \in \mathcal{K}_{\ell_\infty} \subset \mathbb{R}^{1+d} \quad \Leftrightarrow \quad (ue - w, ue + w) \in \mathbb{R}^{2d}_{\geq}, \tag{2.10a}$$

$$(u, w) \in \mathcal{K}^*_{\ell_\infty} \subset \mathbb{R}^{1+d} \quad \Leftrightarrow \quad \exists \theta \in \mathbb{R}^d_{\geq}, \exists \lambda \in \mathbb{R}^d_{\geq}, w = \theta - \lambda, u - e'(\theta + \lambda) \in \mathbb{R}_{\geq}. \tag{2.10b}$$

For the complex vector domain case, these cones are nonpolyhedral. We can write EFs in terms of auxiliary variables and d three-dimensional $\mathcal{K}_{\ell_2}$ cones, which allow us to represent absolute values of complex numbers.

2.2.2.2 Matrix spectral norm cones

The spectral norm cone is the epigraph of the matrix spectral norm (the largest singular value). Its dual cone is the epigraph of the matrix nuclear norm (the sum of singular values), which is the dual norm of the spectral norm. These cones are analogous to $\mathcal{K}_{\ell_\infty}$ and $\mathcal{K}^*_{\ell_\infty}$ because the spectral norm is the ℓ_∞ norm of the singular values and the nuclear norm is the ℓ_1 norm of the singular values. Hypatia supports these cones for four types of matrix domains: real symmetric, complex Hermitian, real rectangular, and complex rectangular matrices. Note for symmetric/Hermitian matrices, the singular values are the absolute values of the eigenvalues. For a matrix W, let $\sigma_i(W) \geq 0$ be the ith largest singular value of W.

First we consider the real symmetric matrix case. For side dimension d, these cones are:

$$\mathcal{K}_{\ell_{\text{sspec}}} \coloneqq \{(u, w) \in \mathbb{R}_\geq \times \mathbb{R}^{\text{sd}(d)} : u \geq \sigma_1(W)\}, \tag{2.11a}$$
$$\mathcal{K}^*_{\ell_{\text{sspec}}} \coloneqq \left\{(u, w) \in \mathbb{R}_\geq \times \mathbb{R}^{\text{sd}(d)} : u \geq \textstyle\sum_{i \in [\![d]\!]} \sigma_i(W)\right\}, \tag{2.11b}$$

where $W \coloneqq \text{mat}(w) \in \mathbb{S}^d$.

The associated EFs compute the ℓ_∞ or ℓ_1 norm of the eigenvalues of W:

$$(u, w) \in \mathcal{K}_{\ell_{\text{sspec}}} \subset \mathbb{R}^{1+\text{sd}(d)} \quad \Leftrightarrow \quad uI(d) - W \in \mathbb{S}^d_\succeq, uI(d) + W \in \mathbb{S}^d_\succeq, \tag{2.12a}$$
$$(u, w) \in \mathcal{K}^*_{\ell_{\text{sspec}}} \subset \mathbb{R}^{1+\text{sd}(d)} \quad \Leftrightarrow \quad \begin{array}{l} \exists \theta \in \mathbb{R}^{\text{sd}(d)}, \exists \lambda \in \mathbb{R}^{\text{sd}(d)}, W = \Theta - \Lambda, \\ \Theta \in \mathbb{S}^d_\succeq, \Lambda \in \mathbb{S}^d_\succeq, u - \text{tr}(\Theta) - \text{tr}(\Lambda) \in \mathbb{R}_\geq, \end{array} \tag{2.12b}$$

where $\Theta \coloneqq \text{mat}(\theta) \in \mathbb{S}^d, \Lambda \coloneqq \text{mat}(\lambda) \in \mathbb{S}^d$.

Now we consider the real rectangular/nonsymmetric matrix case. Suppose W has d rows and s columns, and $d \leq s$; note this is without loss of generality, since W and W' have the same singular values. The spectral and nuclear norm cones are:

$$\mathcal{K}_{\ell_{\text{spec}}(d,s)} \coloneqq \{(u, w) \in \mathbb{R}_\geq \times \mathbb{R}^{ds} : u \geq \sigma_1(W)\}, \tag{2.13a}$$
$$\mathcal{K}^*_{\ell_{\text{spec}}(d,s)} \coloneqq \left\{(u, w) \in \mathbb{R}_\geq \times \mathbb{R}^{ds} : u \geq \textstyle\sum_{i \in [\![d]\!]} \sigma_i(W)\right\}, \tag{2.13b}$$

where $W \coloneqq \text{mat}_{d,s}(w) \in \mathbb{R}^{d \times s}$.

The example in Section 2.3.2 uses the EF for $\mathcal{K}_{\ell_{\text{spec}}}$ from Ben-Tal and Nemirovski (2001, Section 4.2), and Section 2.3.3 uses the EF for $\mathcal{K}^*_{\ell_{\text{spec}}}$ from Recht, Fazel, and Parrilo (2010):

$$(u, w) \in \mathcal{K}_{\ell_{\text{spec}}(d,s)} \subset \mathbb{R}^{1+ds} \quad \Leftrightarrow \quad \begin{bmatrix} uI(d) & W \\ W' & uI(s) \end{bmatrix} \in \mathbb{S}^{d+s}_\succeq, \tag{2.14a}$$
$$(u, w) \in \mathcal{K}^*_{\ell_{\text{spec}}(d,s)} \subset \mathbb{R}^{1+ds} \quad \Leftrightarrow \quad \begin{array}{l} \exists \theta \in \mathbb{R}^{\text{sd}(d)}, \exists \lambda \in \mathbb{R}^{\text{sd}(s)}, \begin{bmatrix} \Theta & W \\ W' & \Lambda \end{bmatrix} \in \mathbb{S}^{d+s}_\succeq, \\ u - (\text{tr}(\Theta) + \text{tr}(\Lambda))/2 \in \mathbb{R}_\geq, \end{array} \tag{2.14b}$$

where $\Theta \coloneqq \text{mat}(\theta) \in \mathbb{S}^r, \Lambda \coloneqq \text{mat}(\lambda) \in \mathbb{S}^s$.

For the Hermitian/complex matrix domains, the reformulations (2.12) and (2.14) have straightforward generalizations in terms of Hermitian PSD cones. To obtain $\mathcal{K}_{\succeq}$ EFs, we apply the $\mathcal{K}_{c\succeq}$ EF (2.4).

For $\mathcal{K}_{\ell_{\text{spec}}}$ we use the LHSCB from Nesterov and Nemirovski (1994), and since $\mathcal{K}_{\ell_{\text{sspec}}}$ is a symmetric matrix slice of $\mathcal{K}_{\ell_{\text{spec}}}$, we use the same LHSCB for $\mathcal{K}_{\ell_{\text{sspec}}}$. We are not aware of an LHSCB with analytic oracles for $\mathcal{K}^*_{\ell_{\text{spec}}}$ or $\mathcal{K}^*_{\ell_{\text{sspec}}}$.

2.2.3 Spectral function cones

Cones in this class are characterized as epigraphs or hypographs of homogeneous spectral functions or perspective functions of spectral functions. These spectral functions are defined on the real vector domain and the symmetric/Hermitian matrix domains. Spectral functions and associated cones are the topic of Chapter 3, where we also develop new LHSCBs with near-optimal parameters and efficient oracles for Hypatia. We summarize properties of the NFs and EFs in Table 2.1 (bottom).

2.2.3.1 Geometric mean and root-determinant cones

The geometric mean cone is the hypograph of the geometric mean function on the nonnegative real vectors:

$$\mathcal{K}_{\text{geo}} := \left\{(u, w) \in \mathbb{R} \times \mathbb{R}^d_{\geq} : u \leq \textstyle\prod_{i \in [\![d]\!]} w_i^{1/d}\right\}. \tag{2.15}$$

Analogously, the root-determinant cone is the hypograph of the root-determinant function (or the geometric mean of the eigenvalues) on the PSD matrices:

$$\mathcal{K}_{\text{rtdet}} := \left\{(u, w) \in \mathbb{R}^{1+\text{sd}(d)} : W \in \mathbb{S}^d_{\succeq}, u \leq (\det(W))^{1/d}\right\}, \tag{2.16}$$

where $W := \text{mat}(w)$. Hypatia also recognizes a complex Hermitian root-determinant cone. We do not define the dual cones here, as they do not provide additional modeling power. We describe a Jordan algebra domain generalization of $\mathcal{K}_{\text{geo}}$ and $\mathcal{K}_{\text{rtdet}}$ in Section 3.7 (including the dual cone definition).

The example in Section 2.3.2 uses an EF for $\mathcal{K}_{\text{geo}}$, and the root-determinant variant of the example in Section 2.3.4 uses an EF for $\mathcal{K}_{\text{geo}}$ indirectly through a $\mathcal{K}_{\text{rtdet}}$ EF. We are aware of three EFs for $\mathcal{K}_{\text{geo}}$: a second order cone EF (*EF-sec*) from Ben-Tal and Nemirovski (2001, Section 3.3.1), a power cone EF (*EF-pow*) from MOSEK ApS (2020a), and an exponential cone EF (*EF-exp*). We contributed *EF-exp* to MathOptInterface as a combination of two bridges (geometric mean cone to relative entropy cone to exponential cones):

$$(u, w) \in \mathcal{K}_{\text{geo}} \subset \mathbb{R}^{1+d} \quad \Leftrightarrow \quad \begin{array}{l} \exists \rho \in \mathbb{R}_{\geq}, \exists \theta \in \mathbb{R}^d, e'\theta \in \mathbb{R}_{\geq}, \\ (\theta_i, u + \rho, w_i) \in \mathcal{K}_{\log}, \forall i \in [\![d]\!]. \end{array} \tag{2.17}$$

EF-pow is not currently available through MathOptInterface bridges, and it has a very similar size and structure to *EF-exp*, so we do not describe or test it. A bridge exists for *EF-sec*; this EF uses multiple levels of variables and three-dimensional second order cone constraints and is complicated

to describe, so we refer the reader to Ben-Tal and Nemirovski (2001, Section 3.3.1). In our empirical comparisons in Sections 2.3.2 and 2.3.4, *EF-sec* typically has larger variable and conic constraint dimensions but smaller barrier parameter than *EF-exp*.

The example in Section 2.3.4 uses the EF from Ben-Tal and Nemirovski (2001, Section 4.2) for $\mathcal{K}_{\text{rtdet}}$:

$$(u, w) \in \mathcal{K}_{\text{rtdet}} \subset \mathbb{R}^{1+\text{sd}(d)} \quad \Leftrightarrow \quad \begin{array}{c} \exists \theta \in \mathbb{R}^{\text{sd}(d)}, (u, \text{diag}(\Theta)) \in \mathcal{K}_{\text{geo}}, \\ \begin{bmatrix} W & \Theta \\ \Theta' & \text{Diag}(\text{diag}(\Theta)) \end{bmatrix} \in \mathbb{S}^{2d}_{\succeq}, \end{array} \tag{2.18}$$

where $\Theta \coloneqq \text{mat}(\theta) \in \mathbb{S}^d$, and the $\mathcal{K}_{\text{geo}}$ constraint is itself replaced with one of the geometric mean cone EFs described above. For the Hermitian domain case, we generalize (2.18) and apply the $\mathcal{K}_{c\succeq}$ EF (2.4).

2.2.3.2 Logarithm and log-determinant cones

The logarithm cone is the hypograph of the perspective function of a sum of natural logarithms on the positive real vectors:

$$\mathcal{K}_{\log} \coloneqq \text{cl}\big\{(u, v, w) \in \mathbb{R}^{2+d} : v > 0, w > 0, u \leq \textstyle\sum_{i \in [\![d]\!]} v \log(w_i / v)\big\}. \tag{2.19}$$

Analogously, the log-determinant cone is the hypograph of the perspective function of the log-determinant function on the positive definite matrices:

$$\mathcal{K}_{\text{logdet}} \coloneqq \text{cl}\big\{(u, v, w) \in \mathbb{R}^{2+\text{sd}(d)} : v > 0, W \in \mathbb{S}^d_{\succ}, u \leq v \log\det(W/v)\big\}, \tag{2.20}$$

where $W \coloneqq \text{mat}(w)$. Hypatia also recognizes a complex Hermitian domain log-determinant cone. We do not define the dual cones here, as they do not provide additional modeling power.

The example in Section 2.3.6 uses the exponential cone EF (when $d > 1$) for $\mathcal{K}_{\log}$:

$$(u, v, w) \in \mathcal{K}_{\log} \subset \mathbb{R}^{2+d} \quad \Leftrightarrow \quad \exists \theta \in \mathbb{R}^d, e'\theta - u \in \mathbb{R}_{\geq}, (\theta_i, 1, w_i) \in \mathcal{K}_{\log}, \forall i \in [\![d]\!]. \tag{2.21}$$

The example in Section 2.3.4 uses the EF (when $d > 1$) for $\mathcal{K}_{\text{logdet}}$ (adapted from (2.18)):

$$(u, v, w) \in \mathcal{K}_{\text{logdet}} \subset \mathbb{R}^{2+\text{sd}(d)} \quad \Leftrightarrow \quad \begin{array}{c} \exists \theta \in \mathbb{R}^{\text{sd}(d)}, (u, v, \text{diag}(\Theta)) \in \mathcal{K}_{\log}, \\ \begin{bmatrix} W & \Theta \\ \Theta' & \text{Diag}(\text{diag}(\Theta)) \end{bmatrix} \in \mathbb{S}^{2d}_{\succeq}, \end{array} \tag{2.22}$$

where $\Theta \coloneqq \text{mat}(\theta) \in \mathbb{S}^d$, and the $\mathcal{K}_{\log}$ constraint is itself replaced with the $\mathcal{K}_{\log}$ EF (2.21). For the Hermitian domain case, we generalize (2.22) and apply the $\mathcal{K}_{c\succeq}$ EF (2.4).

2.2.3.3 Separable spectral function cones

The separable spectral function cone is parametrized by a cone of squares $\mathcal{Q}$ of a Jordan algebra (see Section 3.2) and a convex separable spectral function $\varphi : \text{int}(\mathcal{Q}) \to \mathbb{R}$ (see Section 3.3.2):

$$\mathcal{K}_{\text{sep}} \coloneqq \text{cl}\{(u, v, w) \in \mathbb{R}^{2+\dim(\mathcal{Q})} : v > 0, W \in \text{int}(\mathcal{Q}), u \geq v\varphi(w/v)\}, \tag{2.23}$$

In the special case where φ has the matrix monotone derivative (MMD) property, we propose an LHSCB for $\mathcal{K}_{\text{sep}}$ with near optimal parameter in Proposition 3.6.1. We discuss MMD spectral function cones and their dual cones in depth in Section 3.6. In Hypatia, we predefine several common MMD functions for φ, including the negative entropy, negative squareroot, and power in $(1, 2]$; see Table 3.1 for a complete list.

For the real vector domain with $\mathcal{Q} = \mathbb{R}^d_{\geq}$, $\dim(\mathcal{Q}) = d$ and $\varphi(w) = \sum_{i \in [\![d]\!]} \varphi(w_i)$. The EF is straightforward if there exists a standard conic reformulation of a three-dimensional $\mathcal{K}_{\text{sep}}$ constraint (as discussed in Section 3.8.2):

$$(u, v, w) \in \mathcal{K}_{\text{sep}}\big(\mathbb{R}^d_{\geq}\big) \quad \Leftrightarrow \quad \begin{array}{r} \exists \theta \in \mathbb{R}^d, u - e'\theta \in \mathbb{R}_{\geq}, \\ (\theta_i, v, w_i) \in \mathcal{K}_{\text{sep}}\big(\mathbb{R}^1_{\geq}\big), \forall i \in [\![d]\!], \end{array} \tag{2.24}$$

We use this EF for the example in Section 3.8.4.1.

For the real symmetric domain with $\mathcal{Q} = \mathbb{S}^d_{\succeq}$, $\dim(\mathcal{Q}) = \text{sd}(d)$ and $\varphi(w) = \sum_{i \in [\![d]\!]} \varphi(\lambda_i(W))$, where $\lambda_i(W)$ is the ith eigenvalue of $W = \text{mat}(w) \in \mathbb{S}^d_{\succeq}$. We adapt the 'eigenvalue ordering' EF from Ben-Tal and Nemirovski (2001, Proposition 4.2.1). This is a very large EF:

$$(u, v, w) \in \mathcal{K}_{\text{sep}}\big(\mathbb{S}^d_{\succeq}\big) \quad \Leftrightarrow \quad \begin{array}{r} \exists \lambda \in \mathbb{R}^d, s \in \mathbb{R}^{d-1}, Z_i \in \mathbb{S}^d_{\succeq}, \forall i \in [\![d-1]\!], \\ e'\lambda = \text{tr}(W), \lambda_i \geq \lambda_{i+1}, \forall i \in [\![d-1]\!], \\ \sum_{i \in [\![j]\!]} \lambda_i - j s_j - \text{tr}(Z_j) \geq 0, \forall j \in [\![d-1]\!], \\ Z_j - W + s_j I(d) \in \mathbb{S}^d_{\succeq}, \forall j \in [\![d-1]\!], \\ (u, v, \lambda) \in \mathcal{K}_{\text{sep}}\big(\mathbb{R}^d_{\geq}\big), \end{array} \tag{2.25}$$

where the constraint over $\mathcal{K}_{\text{sep}}\big(\mathbb{R}^d_{\geq}\big)$ is itself replaced with the vector domain EF (2.24). We use this EF for the examples in Sections 3.8.4.2 and 3.8.4.3.

For the complex Hermitian domain with $\mathcal{Q} = \mathbb{H}^d_{\succeq}$, $\dim(\mathcal{Q}) = d^2$ and we generalize (2.25) using Hermitian PSD cones. Finally, we apply the $\mathcal{K}_{c\geq}$ EF (2.4). We use this EF for the example in Section 3.8.4.4.

2.3 Examples and computational testing

In Sections 2.3.1 to 2.3.7, we present example problems with NFs using some of Hypatia's predefined cones and EFs constructed using the techniques from Section 2.2. For each example problem, we generate random instances of a wide variety of sizes, and we observe larger dimensions and often

larger barrier parameters for EFs compared to NFs. In Tables 2.2 to 2.11, ν and n, p, q refer to the NF barrier parameter and primal variable, linear equality, and cone inequality dimensions (in our general conic form (2.1)), and $\bar{\nu}$, $\bar{n}$, $\bar{p}$, $\bar{q}$ refer to the corresponding EF values. The EFs tend to be slower and more memory-intensive to construct than the NFs, and often we cannot even build an EF given time and memory constraints.

For three solver/formulation combinations - Hypatia with NF (*Hyp-NF*), Hypatia with EF (*Hyp-EF*), and MOSEK with EF (*MO-EF*) - we compare termination statuses, iteration counts, and solve times in seconds (columns *st*, *it*, and *time*) in Tables 2.2 to 2.11 and Figure 2.1. In Sections 2.3.2 and 2.3.4 we depend on a geometric mean cone EF, so we compare the *EF-exp* and *EF-sec* formulations from Section 2.2.3.1. For the sake of simplicity, all instances we generate are primal-dual feasible, so we expect solvers to return optimality certificates. Compared to Hyp-EF and MO-EF, Hyp-NF generally converges faster and more reliably, and solves larger instances within time and memory limits.

We perform instance generation, computational experiments, and results analysis with Ubuntu 21.04 and Julia 1.7. We use dedicated hardware with an AMD Ryzen 9 3950X 16-core processor (32 threads) and 128GB of RAM, and we limit each solver to 16 threads. We use JuMP 0.21.5 and MathOptInterface 0.9.18 to build all instances. For most examples we use Hypatia 0.5.0, but for the spectral norm cone examples in Sections 2.3.2 and 2.3.3 we use Hypatia 0.7.0, since we added the new $\mathcal{K}_{\ell_{\text{spec}}}$ oracles from Section 2.6 after version 0.5.0. For Hypatia, we use the default set of options that we describe and test in Section 1.9. We use MOSEK 9 through MosekTools.jl 0.9.4 (which is maintained in part by MOSEK).[1] MOSEK uses its conic interior point method for all solves. We note that MOSEK heuristically determines whether it is more efficient to solve the primal or dual of an instance (MOSEK ApS, 2020b, Section 13.1); Hypatia does not do this. We do not disable any MOSEK features.

Hypatia and MOSEK use similar convergence criteria (see MOSEK ApS (2020b, Section 13.3.2)), and we set their feasibility and optimality gap tolerances to 10^{-7}. In the solver statistics tables, asterisks indicate missing data, and we use the following codes for the termination status (st) columns:

co: the solver claims it has an approximate optimality certificate,

tl: the solver stops itself due to a solve time limit of 1800 seconds, or the solve run is killed because it takes at least 1.2×1800 seconds,

rl: the solve is terminated because insufficient RAM is available,

sp: the solver reports *slow progress* during iterations,

[1] MOSEK 9's primal conic form only recognizes conic constraints of the form $x \in \mathcal{K}$ (MOSEK ApS, 2020a, Section 8), whereas Hypatia accepts the more general affine form $h - Gx \in \mathcal{K}$ (see (2.1c)). MathOptInterface recognizes both *VectorOfVariables* form $x \in \mathcal{K}$ and *VectorAffineFunction* form $h - Gx \in \mathcal{K}$. Since JuMP and MathOptInterface (including bridges) use the $x \in \mathcal{K}$ form whenever possible, unnecessary high dimensional slack variables are not introduced when instances in Hypatia's general conic form (2.1) are converted into MOSEK 9's form.

er: the solver encounters a numerical error,

m: the model cannot be constructed with JuMP due to insufficient RAM or a model generation time limit of 1.2×1800 seconds (this means the EF columns often have missing data),

sk: we skip the solve run because a smaller instance has a tl or rl status, or we skip model generation because a smaller instance has an m status.

For each solve run that yields a primal-dual point (x, y, z, s) (see Section 1.4; $s \in \mathcal{K}$ and $z \in \mathcal{K}^*$ are the solver's primal and dual cone interior points at termination), we compute:

$$\epsilon := \max\left\{ \frac{\|A'y + G'z + c\|_\infty}{1 + \|c\|_\infty}, \frac{\|-Ax + b\|_\infty}{1 + \|b\|_\infty}, \frac{\|-Gx + h - s\|_\infty}{1 + \|h\|_\infty}, \frac{|c'x + b'y + h'z|}{1 + |b'y + h'z|} \right\}, \tag{2.26}$$

and if $\epsilon < 10^{-5}$, we underline the corresponding status code (e.g. co, tl) to indicate that the solution approximately satisfies the optimality certificate conditions from Section 1.4. In our solve time plots in Figure 2.1, we only plot solve runs with underlined status codes. Finally, for each instance and each pair of corresponding solve runs with co status codes, we compute the relative difference of the primal objective values g_1 and g_2 as $\tilde{\epsilon} := |g_1 - g_2|/(1 + \max(|g_1|, |g_2|))$. We note $\tilde{\epsilon} < 10^{-5}$ for most instances and pairs of solvers, and $\tilde{\epsilon} < 10^{-3}$ in all cases.

Simple scripts and instructions for reproducing all results are available in Hypatia's benchmarks/natvsext folder. A CSV file containing raw results is available at the Hypatia wiki page.

2.3.1 Portfolio rebalancing

Suppose there are k possible investments with expected returns $g \in \mathbb{R}^k_>$ and covariance matrix $\Sigma \in \mathbb{S}^k_\succ$. We let $\rho \in [-1, 1]^k$ be the investment variable, which must also satisfy side constraints $F\rho = 0$, where $F \in \mathbb{R}^{l \times k}$. We formulate a risk-constrained portfolio rebalancing optimization problem as:

$$\begin{aligned}
\max_{\rho \in \mathbb{R}^k} \quad & g'\rho : && \text{(2.27a)}\\
& e'\rho = 0, && \text{(2.27b)}\\
& F\rho = 0, && \text{(2.27c)}\\
& (1, \rho) \in \mathcal{K}_{\ell_\infty}, && \text{(2.27d)}\\
& (\gamma, \Sigma^{1/2}\rho) \in \mathcal{K}^*_{\ell_\infty}. && \text{(2.27e)}
\end{aligned}$$

Note (2.27d) expresses $\rho \in [-1, 1]^k$ and (2.27e) is a risk constraint. The EFs for (2.27d) and (2.27e) follow (2.10a) and (2.10b) and are constructed automatically from the NFs by MathOptInterface's bridges. Note the EF is a standard linear program (LP).

To build random instances of (2.27), we generate g with independent uniform positive entries, and $\Sigma^{1/2}$ and F with independent Gaussian entries, for $l = k/2$ and various values of k. We use $\Sigma^{1/2}$ to compute reasonable values for the risk parameter $\gamma > 0$, ensuring feasibility. Our results are summarized in Table 2.2 and Figure 2.1a. Note that $\nu = q = 2k + 2$, $\bar{\nu} = \bar{q} = 4k + 1$, $n = k$, $\bar{n} = 2k$,

$p = \bar{p} = k/2 + 1$. The variable and conic constraint dimensions of the EFs are approximately double those of the NFs.

In Table 2.3, we compare the solve times of several LP solvers on a subset of sizes for the EF instances. We run the open source conic IPM solver ECOS 2.0.5 (Domahidi, Chu, and Boyd, 2013), MOSEK's conic IPM (the *conic* column), MOSEK's *intpnt* and *simplex* algorithms (see MOSEK ApS (2020b, 13.2: Linear Optimization)), and Gurobi 9.5.0's *barrier* and primal and dual simplex algorithms (*P simplex*, *D simplex*; see Gurobi (2022)). Of these LP solvers, MOSEK's conic IPM solves the instances fastest. This is why we only include MOSEK's conic IPM in our main NF versus EF comparisons in Table 2.2. We emphasize that no general conclusions about LP solver performance should be made from these results on our small set of relatively dense random instances.

Table 2.2: Portfolio rebalancing solver statistics.

	Hyp-NF			Hyp-EF			MO-EF		
k	st	it	time	st	it	time	st	it	time
1000	co	31	0.6	co	25	2.7	co	9	1.7
2000	co	36	2.9	co	28	16	co	10	7.0
4000	co	45	20	co	29	92	co	10	34
6000	co	49	60	co	34	292	co	10	83
8000	co	51	131	co	33	615	co	10	160
10000	co	55	244	co	36	1192	co	12	305
12000	co	62	421	tl	32	1805	co	10	433
14000	co	61	624	sk	*	*	rl	*	*
16000	co	63	924	sk	*	*	sk	*	*
18000	co	64	1327	sk	*	*	sk	*	*
20000	co	66	1810	sk	*	*	sk	*	*

Table 2.3: Portfolio rebalancing EF LP solver time comparisons.

	ECOS	MOSEK			Gurobi		
k	conic	conic	intpnt	simplex	barrier	P simplex	D simplex
1000	192	1.7	2.3	39	2.8	68	57
4000	*	34	66	*	55	*	*
10000	*	305	783	*	548	*	*
20000	*	*	*	*	*	*	*

2.3.2 Matrix completion

Suppose there exists a matrix $F \in \mathbb{R}^{k \times l}$ and we know the entries $(F_{i,j})_{(i,j)\in\mathcal{S}}$ in the sparsity pattern $\mathcal{S}$. In the matrix completion problem, we seek to estimate the missing components $(F_{i,j})_{(i,j)\notin\mathcal{S}}$. We modify the formulation in Agrawal, Diamond, and Boyd (2019, Section 4.3) by replacing the spectral radius in the objective function with the spectral norm (allowing rectangular matrices) and using a

convex relaxation of the geometric mean equality constraint:

$$\min_{\rho\in\mathbb{R},X\in\mathbb{R}^{k\times l}} \quad \rho : \tag{2.28a}$$

$$X_{i,j} = F_{i,j} \qquad \forall (i,j) \in \mathcal{S}, \tag{2.28b}$$

$$(\rho, \mathrm{vec}(X)) \in \mathcal{K}_{\ell_{\mathrm{spec}}(k,l)}, \tag{2.28c}$$

$$\big(1, (X_{i,j})_{(i,j)\notin\mathcal{S}}\big) \in \mathcal{K}_{\mathrm{geo}}. \tag{2.28d}$$

To reduce the dimensionality of this problem, we eliminate the equality constraints and the variables $(X_{i,j})_{(i,j)\in\mathcal{S}}$. The EF for (2.28c) follows (2.14a), and for (2.28d) we compare *EF-exp* and *EF-sec* (see Section 2.2.3.1).

We generate random instances of (2.28) with column-to-row ratios $m \in \{10, 20\}$ and $l = mk$ for varying k. Our results are summarized in Tables 2.4 and 2.5 and Figure 2.1b. Note we only plot *EF-sec* results for Hyp-EF and MO-EF, as MOSEK performs better with *EF-sec* than with *EF-exp*, though Hypatia exhibits the opposite trend.

Table 2.4: Matrix completion formulation statistics. Note $p = \bar{p} = 0$.

		NF			EF-exp			EF-sec		
m	k	ν	n	q	$\bar{\nu}$	$\bar{n}$	$\bar{q}$	$\bar{\nu}$	$\bar{n}$	$\bar{q}$
	5	67	61	312	237	122	1722	182	124	1730
	10	201	190	1191	679	380	6674	621	445	6871
	15	471	455	2706	1529	910	15059	1188	966	15229
	20	826	805	4806	2634	1610	26724	2267	1828	27380
	25	1281	1255	7506	4039	2510	41714	4370	3302	44092
	30	1851	1820	10821	5789	3640	60074	4425	3867	60757
	35	2487	2451	14702	7737	4902	81657	8576	6546	86591
10	40	3237	3196	19197	10027	6392	106607	8631	7291	109306
	45	4104	4058	24309	12668	8116	134933	8686	8153	135046
	50	5029	4978	29979	15483	9956	166458	16933	13169	176099
	55	6085	6029	36280	18691	12058	201401	16988	14220	207889
	60	7210	7149	43150	22106	14298	239576	17043	15340	242704
	65	8419	8353	50604	25773	16706	281028	33482	24736	305120
	70	9798	9727	58728	29950	19454	326015	33537	26110	345985
	75	11198	11122	67373	34190	22244	374090	33592	27505	389875
	5	105	99	600	401	198	5861	360	226	5947
	10	415	404	2405	1421	808	23366	1233	915	23689
	15	925	909	5410	3041	1818	52496	2362	1932	52840
	20	1646	1625	9626	5294	3250	93284	4515	3672	94552
	25	2527	2501	15002	8027	5002	145577	8716	6596	150361
20	30	3645	3614	21615	11471	7228	209606	8821	7709	211051
	35	4918	4882	29383	15380	9764	285125	17118	13073	295054
	40	6405	6364	38365	19931	12728	372311	17223	14555	377794
	45	8068	8022	48523	25010	16044	471050	17328	16213	471559
	50	9969	9918	59919	30803	19836	581528	33817	26301	600925
	55	12076	12020	72521	37214	24040	703649	33922	28403	716740

Table 2.5: Matrix completion solver statistics.

		NF			EF-exp						EF-sec					
		Hypatia			Hypatia			MOSEK			Hypatia			MOSEK		
m	k	st	it	time	st	it	time	st	it	time	st	it	time	st	it	time
	5	co	13	0.0	co	18	0.1	co	13	0.7	co	18	0.1	co	9	0.5
	10	co	14	0.2	co	32	1.4	co	19	18	co	25	1.4	co	10	10
	15	co	18	0.7	co	41	11	co	20	103	co	36	12	co	11	65
	20	co	25	2.9	co	48	50	co	20	392	co	49	63	co	11	230
	25	co	27	8.2	co	51	144	co	21	1265	co	63	295	co	10	638
	30	co	26	24	co	49	325	tl	13	1892	co	47	373	co	11	1654
	35	co	36	77	co	53	726	sk	*	*	co	77	1690	rl	*	*
10	40	co	34	113	co	52	1415	sk	*	*	tl	50	1809	sk	*	*
	45	co	37	207	tl	26	1829	sk	*	*	sk	*	*	sk	*	*
	50	co	40	336	sk	*	*	sk	*	*	sk	*	*	sk	*	*
	55	co	39	519	sk	*	*	sk	*	*	sk	*	*	sk	*	*
	60	co	45	835	sk	*	*	sk	*	*	sk	*	*	sk	*	*
	65	co	47	1320	sk	*	*	sk	*	*	sk	*	*	sk	*	*
	70	ne	42	1802	sk	*	*	sk	*	*	sk	*	*	sk	*	*
	75	tl	23	1809	sk	*	*	sk	*	*	sk	*	*	sk	*	*
	5	co	11	0.1	co	24	0.5	co	18	14	co	21	0.5	co	10	8.4
	10	co	17	0.5	co	42	17	co	19	318	co	37	18	co	10	191
	15	co	24	3.4	co	56	132	tl	16	1877	co	52	141	co	11	1340
	20	co	27	14	co	58	510	sk	*	*	co	55	602	rl	*	*
	25	co	35	67	co	63	1582	sk	*	*	tl	39	1812	sk	*	*
20	30	co	44	163	tl	14	1807	sk	*	*	sk	*	*	sk	*	*
	35	co	44	382	sk	*	*	sk	*	*	sk	*	*	sk	*	*
	40	co	49	754	sk	*	*	sk	*	*	sk	*	*	sk	*	*
	45	co	53	1349	sk	*	*	sk	*	*	sk	*	*	sk	*	*
	50	tl	37	1801	sk	*	*	sk	*	*	sk	*	*	sk	*	*
	55	sk	*	*	sk	*	*	sk	*	*	sk	*	*	sk	*	*

2.3.3 Multi-response regression

In the multi-response linear regression problem, we seek to estimate a coefficient matrix $F \in \mathbb{R}^{m\times l}$ from a design matrix $X \in \mathbb{R}^{l\times k}$ and response matrix $Y \in \mathbb{R}^{m\times k}$. We use a similar formulation to the one proposed in Yang et al. (2016), with nuclear norm loss and ℓ_2 norm regularization:

$$\min_{\rho\in\mathbb{R},\mu\in\mathbb{R},F\in\mathbb{R}^{m\times l}} \quad \rho + \gamma\mu : \tag{2.29a}$$

$$(\rho, \mathrm{vec}(Y - FX)) \in \mathcal{K}^*_{\ell_{\mathrm{spec}}(m,k)}, \tag{2.29b}$$

$$(\mu, \mathrm{vec}(F)) \in \mathcal{K}_{\ell_2}. \tag{2.29c}$$

The EF for NF constraint (2.29b) follows (2.14b).

We generate random instances of (2.29) with $l = m \in \{15, 30\}$ for varying k, and let $\gamma = 0.1$. Our results are summarized in Table 2.6 and Figure 2.1c. Note that $\nu = 3 + m$, $\bar{\nu} = \nu + k$, $n = 2 + m^2$, $p = \bar{p} = 0$, $\bar{q} = \bar{n} + mk$. The variable dimensions for the NFs only depend on k and are much smaller than those of the EFs. The EFs also have much larger conic constraint dimensions.

Table 2.6: Multi-response regression formulation and solver statistics.

		form. stats.		Hyp-NF			Hyp-EF			MO-EF		
m	k	$\bar{n}$	q	st	it	time	st	it	time	st	it	time
15	30	812	677	co	9	0.1	co	10	0.4	co	5	0.2
	100	5397	1727	co	9	0.1	co	10	28	co	5	6.5
	300	45497	4727	co	9	0.3	tl	*	*	co	5	709
	1000	500847	15227	co	10	0.8	sk	*	*	m	*	*
	3000	4501847	45227	co	10	3.4	sk	*	*	sk	*	*
	10000	50005347	150227	co	9	9.8	sk	*	*	sk	*	*
	30000	450015347	450227	co	10	32	sk	*	*	sk	*	*
	100000	*	1500227	co	9	97	m	*	*	sk	*	*
	300000	*	4500227	co	10	321	sk	*	*	sk	*	*
	1000000	*	15000227	rl	*	*	sk	*	*	sk	*	*
30	30	1832	1802	co	11	0.5	co	13	1.8	co	4	0.4
	100	6417	3902	co	10	1.2	co	11	54	co	5	12
	300	46517	9902	co	11	4.3	tl	*	*	co	5	882
	1000	501867	30902	co	10	11	sk	*	*	m	*	*
	3000	4502867	90902	co	10	32	sk	*	*	sk	*	*
	10000	50006367	300902	co	10	99	sk	*	*	sk	*	*
	30000	450016367	900902	co	9	270	sk	*	*	sk	*	*
	100000	*	3000902	co	10	1058	m	*	*	sk	*	*
	300000	*	9000902	rl	*	*	sk	*	*	sk	*	*

2.3.4 D-optimal experiment design

In a continuous relaxation of the D-optimal experiment design problem (see Boyd and Vandenberghe (2004, Section 7.5)), the variable $\mu \in \mathbb{R}^m$ is the number of trials to run for each of m experiments, and our goal is to minimize the determinant of the error covariance matrix $(F\,\mathrm{Diag}(\mu)F')^{-1}$, given a

menu of experiments $F \in \mathbb{R}^{k\times m}$ useful for estimating a vector in $\mathbb{R}^k$. We require that a total of j experiments are performed and that each experiment can be performed between 0 and l times. We formulate this problem as:

$$\max_{\rho\in\mathbb{R},\mu\in\mathbb{R}^m} \quad \rho : \tag{2.30a}$$

$$e'\mu = j, \tag{2.30b}$$

$$(l/2, \mu - (l/2)e) \in \mathcal{K}_{\ell_\infty}, \tag{2.30c}$$

$$(\rho, \mathrm{vec}(F\,\mathrm{Diag}(\mu)F')) \in \mathcal{K}_{\mathrm{rtdet}}. \tag{2.30d}$$

In an alternative *logdet* variant of the *rtdet* variant (2.30), we replace (2.30d) with:

$$(\rho, 1, \mathrm{vec}(F\,\mathrm{Diag}(\mu)F')) \in \mathcal{K}_{\mathrm{logdet}}, \tag{2.31}$$

noting that both variants have the same optimal solution set for μ. The EFs for (2.30c), (2.30d) and (2.31) follow (2.10a), (2.18) and (2.22). Since the EF for $\mathcal{K}_{\mathrm{rtdet}}$ depends on a $\mathcal{K}_{\mathrm{geo}}$ EF, for the rtdet variant we compare *EF-exp* and *EF-sec* (see Section 2.2.3.1).

We generate random instances of (2.30) with $m = j = 2k$ and $l = 5$ for varying k. Our results are summarized in Tables 2.7 and 2.8 and Figure 2.1d. For the logdet variant, $\nu = 3+3k$, $\bar{\nu} = 1+9k$, $n = 1+2k$, $p = \bar{p} = 1$. The sizes for the rtdet formulations are similar to those of the logdet formulations, so we exclude these. Note for the rtdet variant, we only plot *EF-sec* results for Hyp-EF and MO-EF, as MOSEK typically performs slightly better with *EF-sec* than with *EF-exp*, though Hypatia exhibits the opposite trend. For both variants, the NFs have much lower variable and conic constraint dimensions than the EFs.

Table 2.7: D-optimal experiment design rtdet variant solver statistics.

	NF			EF-exp						EF-sec					
	Hypatia			Hypatia			MOSEK			Hypatia			MOSEK		
k	st	it	time	st	it	time	st	it	time	st	it	time	st	it	time
50	co	25	0.3	co	22	4.7	co	14	11	co	22	5.1	co	11	10
100	co	25	0.9	co	25	93	co	13	247	co	26	97	co	11	220
150	co	26	2.6	co	27	696	co	12	1580	sp	36	921	co	10	1432
200	co	23	5.8	tl	17	1821	tl	0	1821	tl	17	1848	tl	0	1868
300	co	31	31	sk	*	*	sk	*	*	sk	*	*	sk	*	*
400	co	29	67	m	*	*	sk	*	*	m	*	*	sk	*	*
500	co	32	152	sk	*	*	sk	*	*	sk	*	*	sk	*	*
600	co	33	281	sk	*	*	sk	*	*	sk	*	*	sk	*	*
700	co	32	530	sk	*	*	m	*	*	sk	*	*	sk	*	*
800	co	32	728	sk	*	*	sk	*	*	sk	*	*	sk	*	*
900	co	36	1253	sk	*	*	sk	*	*	sk	*	*	sk	*	*
1000	co	33	1729	sk	*	*	sk	*	*	sk	*	*	sk	*	*

Figure 2.1: Solve times (in seconds) for solve runs satisfying the convergence check in (2.26).

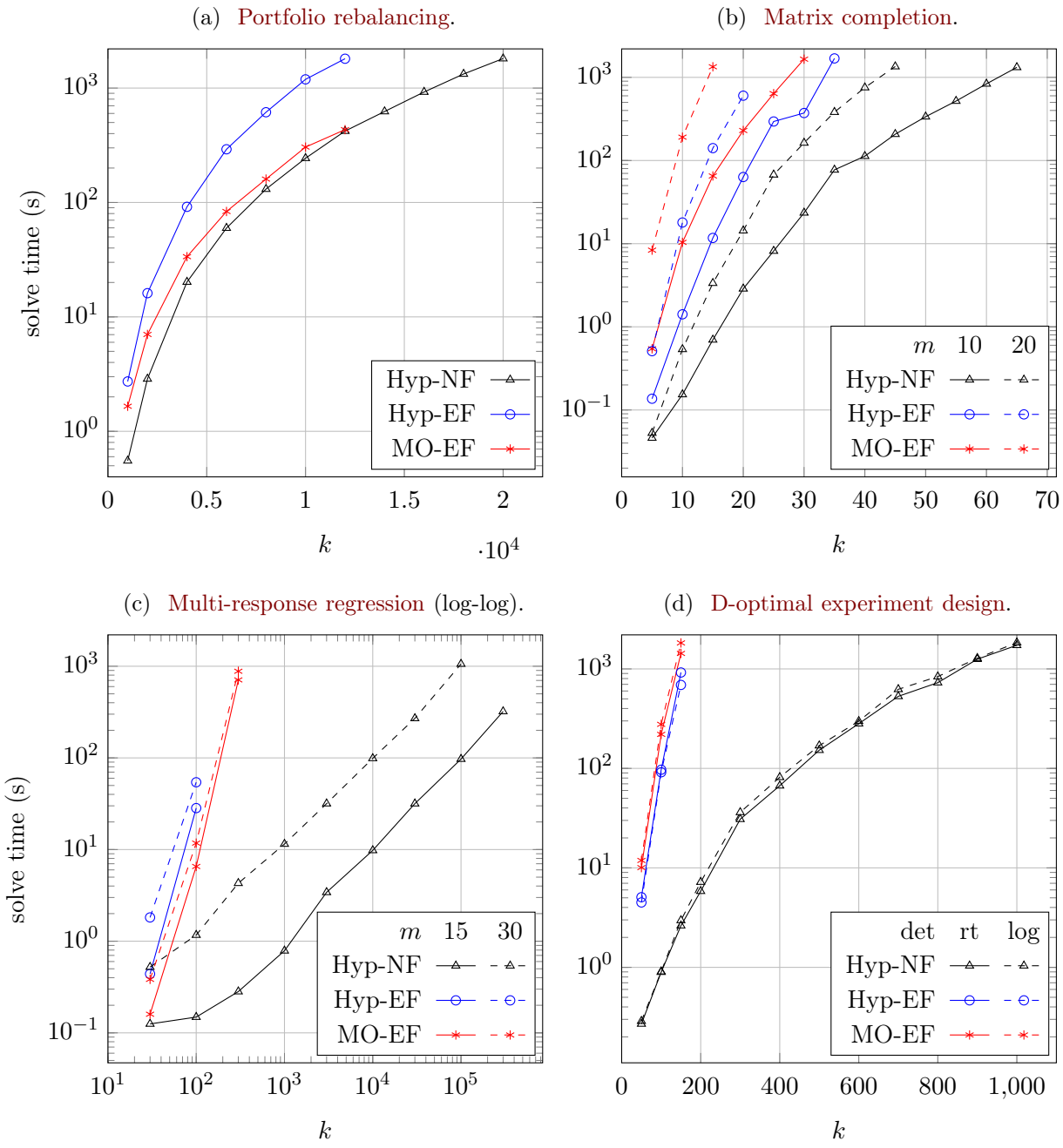


Table 2.8: D-optimal experiment design logdet variant formulation and solver statistics.

	form. stats.			Hyp-NF			Hyp-EF			MO-EF		
k	$\bar{n}$	q	$\bar{q}$	st	it	time	st	it	time	st	it	time
50	1426	1378	5401	co	25	0.3	co	21	4.5	co	15	12
100	5351	5253	20801	co	26	0.9	co	25	91	co	15	277
150	11776	11628	46201	co	29	3.0	co	27	690	tl	14	1825
200	20701	20503	81601	co	28	7.2	tl	17	1849	sk	$*$	$*$
300	46051	45753	182401	co	36	36	sk	$*$	$*$	sk	$*$	$*$
400	81401	81003	323201	co	36	81	m	$*$	$*$	sk	$*$	$*$
500	126751	126253	504001	co	36	169	sk	$*$	$*$	sk	$*$	$*$
600	182101	181503	724801	co	36	298	sk	$*$	$*$	sk	$*$	$*$
700	$*$	246753	$*$	co	39	624	sk	$*$	$*$	m	$*$	$*$
800	$*$	322003	$*$	co	37	838	sk	$*$	$*$	sk	$*$	$*$
900	$*$	407253	$*$	co	37	1282	sk	$*$	$*$	sk	$*$	$*$
1000	$*$	502503	$*$	tl	37	1838	sk	$*$	$*$	sk	$*$	$*$

2.3.5 Polynomial minimization

Following Papp and Yıldız (2019), we use an SOS formulation to find a lower bound for a multivariate polynomial f of maximum degree $2k$ in m variables over the unit hypercube $\mathcal{D} = [-1, 1]^m$. We let $U = \binom{m+2k}{m}$, $L = \binom{m+k}{m}$, $\tilde{L} = \binom{m+k-1}{m}$. We select multivariate Chebyshev basis polynomials $g_j, \forall j \in [\![L]\!]$ of increasing degree up to k, and suitable interpolation points $o_u \in \mathcal{D}, \forall u \in [\![U]\!]$. To parametrize $\mathcal{K}^*_{\text{SOS}(P)}$, we set up the collection of matrices P by evaluating functions of basis polynomials at the points:

$$(P_1)_{u,j} = g_j(o_u) \qquad \forall u \in [\![U]\!], j \in [\![L]\!], \tag{2.32a}$$

$$(P_{1+i})_{u,j} = g_j(o_u)\big(1 - o_{u,i}^2\big) \qquad \forall i \in [\![m]\!], u \in [\![U]\!], j \in [\![\tilde{L}]\!]. \tag{2.32b}$$

Letting $\bar{f} = (f(o_u))_{u \in U}$ be evaluations of f at the points, the conic formulation is:

$$\min_{\rho \in \mathbb{R}^U} \quad \bar{f}'\rho : \tag{2.33a}$$

$$e'\rho = 1, \tag{2.33b}$$

$$\rho \in \mathcal{K}^*_{\text{SOS}(P)}. \tag{2.33c}$$

The EF for NF constraint (2.33c) uses $\mathcal{K}_{\succeq}$ and is implicit in (2.7b).

We generate random instances of (2.33) for varying m and k. Our results are summarized in Table 2.9. Note that $\nu = \bar{\nu}$, $p = \bar{p} = 1$, $n = \bar{n} = q$. For fixed m, the conic constraint dimensions are larger for the EFs and grow much faster for the EFs as the degree k increases. Hyp-NF is faster than the EF solvers on all instances with $k > 1$.

Table 2.9: Polynomial minimization formulation and solver statistics.

		form. stats.			Hyp-NF			Hyp-EF			MO-EF		
m	k	ν	n	$\bar{q}$	st	it	time	st	it	time	st	it	time
1	100	201	201	10201	co	12	0.1	co	34	1.2	co	15	27
1	200	401	401	40401	co	14	0.3	co	39	13	co	11	409
1	500	1001	1001	251001	co	18	2.4	co	57	329	rl	*	*
1	1000	2001	2001	*	co	19	11	m	*	*	sk	*	*
1	2000	4001	4001	*	co	21	73	sk	*	*	sk	*	*
1	3000	6001	6001	*	co	24	235	sk	*	*	sk	*	*
1	4000	8001	8001	*	co	24	508	sk	*	*	sk	*	*
1	5000	10001	10001	*	co	24	916	sk	*	*	sk	*	*
2	15	376	496	23836	co	15	0.4	co	21	5.0	co	10	87
2	30	1426	1891	339946	co	25	10	co	49	751	rl	*	*
2	45	3151	4186	*	co	22	58	m	*	*	sk	*	*
2	60	5551	7381	*	co	28	300	sk	*	*	sk	*	*
2	75	8626	11476	*	co	30	1019	sk	*	*	sk	*	*
3	6	252	455	8358	co	17	0.3	co	17	1.6	co	9	9.1
3	9	715	1330	65395	co	20	3.1	co	24	104	co	9	799
3	12	1547	2925	303030	co	23	20	co	33	1775	rl	*	*
3	15	2856	5456	*	co	26	89	m	*	*	sk	*	*
3	18	4750	9139	*	er	34	1340	sk	*	*	sk	*	*
4	4	210	495	5005	co	18	0.4	co	16	1.7	co	8	3.9
4	6	714	1820	54159	co	15	4.8	co	18	222	co	10	579
4	8	1815	4845	*	co	20	63	m	*	*	m	*	*
4	10	3861	10626	*	co	22	458	sk	*	*	sk	*	*
8	2	117	495	1395	co	26	0.5	co	21	0.7	co	11	0.9
8	3	525	3003	21975	co	18	15	co	16	148	co	8	125
8	4	1815	12870	*	co	27	633	m	*	*	m	*	*
16	1	33	153	169	co	13	0.1	co	12	0.7	co	7	0.0
16	2	425	4845	14229	co	27	86	co	22	174	sp	10	192
32	1	65	561	593	co	15	0.7	co	12	1.0	co	7	0.2
64	1	129	2145	2209	co	15	14	co	12	3.1	co	9	3.0

2.3.6 Smooth density estimation

$\mathbb{R}_{m,2k}[x]$ is the ring of polynomials of maximum degree $2k$ in m variables (Papp and Yıldız, 2019). We seek a polynomial density function $f \in \mathbb{R}_{m,2k}[x]$ over the domain $\mathcal{D} = [-1,1]^m$ that maximizes the log-likelihood of N given observations $z_i \in \mathcal{D}, \forall i \in [\![N]\!]$ (compare to Papp and Alizadeh (2014, Section 4.3)). For f to be a valid density it must be nonnegative on $\mathcal{D}$ and integrate to one over $\mathcal{D}$, so we aim to solve:

$$\max_{f \in \mathbb{R}_{m,2k}[x]} \quad \sum_{i \in [\![N]\!]} \log(f(z_i)) : \tag{2.34a}$$

$$\int_{\mathcal{D}} f(x)\, dx = 1, \tag{2.34b}$$

$$f(x) \geq 0 \qquad \forall x \in \mathcal{D}. \tag{2.34c}$$

To find a feasible solution for (2.34), we build an SOS formulation. We obtain interpolation points and matrices P parametrizing $\mathcal{K}_{\text{SOS}(P)}$, using the techniques from Section 2.3.5. From the interpolation points and the domain $\mathcal{D}$, we compute a vector of quadrature weights $\mu \in \mathbb{R}^U$. We compute a matrix $B \in \mathbb{R}^{N \times U}$ by evaluating the U Lagrange basis polynomials corresponding to the interpolation points (see Papp and Yıldız (2019)) at the N observations. Letting variable ρ represent the coefficients on the Lagrange basis, the conic formulation is:

$$\max_{\psi \in \mathbb{R}, \rho \in \mathbb{R}^U} \quad \psi : \tag{2.35a}$$

$$\mu' \rho = 1, \tag{2.35b}$$

$$(\psi, 1, B\rho) \in \mathcal{K}_{\log}, \tag{2.35c}$$

$$\rho \in \mathcal{K}_{\text{SOS}(P)}. \tag{2.35d}$$

The EFs for NF constraints (2.35c) and (2.35d) follow (2.7a) and (2.21).

We generate random instances of (2.35) for varying m and k, with $N = 500$. Our method for computing μ is unstable for large m, so we use $m \leq 16$. Our results are summarized in Table 2.10. Note that $\bar{\nu} = 999 + \nu$, $p = 1$, $\bar{p} = n$, $q = 501 + n$, $\bar{q} = 1001 + \bar{n} - n$. All dimensions are larger for the EFs than for the NFs.

2.3.7 Shape constrained regression

A common type of shape constraint imposes monotonicity or convexity of a polynomial over a basic semialgebraic set (Hall, 2019, Section 6). Given an m-dimensional feature variable z and a scalar response variable g, we aim to fit a polynomial $f \in \mathbb{R}_{m,2k}[x]$ that is convex over $\mathcal{D} = [-1,1]^m$ to N given observations $(z_i, g_i)_{i \in [\![N]\!]}$ with $z_i \in \mathcal{D}, \forall i \in [\![N]\!]$:

$$\min_{f \in \mathbb{R}_{m,2k}[x]} \quad \sum_{i \in [\![N]\!]} (g_i - f(z_i))^2 : \tag{2.36a}$$

$$y'(\nabla^2 f(x)) y \geq 0 \qquad \forall x \in \mathcal{D}, y \in \mathbb{R}^m. \tag{2.36b}$$

Table 2.10: Smooth density estimation formulation and solver statistics.

		form. stats.			Hyp-NF			Hyp-EF			MO-EF		
m	$2k$	ν	n	$\bar{n}$	st	it	time	st	it	time	st	it	time
1	250	753	252	16628	co	35	0.2	sp	43	1040	co	25	112
1	500	1003	502	64003	co	42	1.1	rl	*	*	tl	18	1814
1	1000	1503	1002	*	co	41	5.1	m	*	*	m	*	*
1	2000	2503	2002	*	co	56	23	sk	*	*	sk	*	*
1	4000	4503	4002	*	co	82	185	sk	*	*	sk	*	*
1	6000	6503	6002	*	co	106	663	sk	*	*	sk	*	*
2	20	678	232	6023	co	50	0.3	co	29	73	co	19	7.4
2	40	1153	862	72468	co	34	2.9	rl	*	*	sp	22	1522
2	60	1928	1892	*	co	36	11	m	*	*	m	*	*
2	80	3003	3322	*	co	53	64	sk	*	*	sk	*	*
2	100	4378	5152	*	co	64	247	sk	*	*	sk	*	*
3	12	754	456	9314	co	57	0.9	co	25	293	sp	19	20
3	18	1217	1331	67226	co	55	6.6	rl	*	*	sp	17	1216
3	24	2049	2926	*	co	46	35	m	*	*	m	*	*
3	30	3358	5457	*	co	63	348	sk	*	*	sk	*	*
4	8	712	496	6001	co	57	1.0	co	25	133	sp	22	12
4	12	1216	1821	56480	co	72	16	tl	*	*	sp	20	934
4	16	2317	4846	*	co	70	192	m	*	*	m	*	*
8	4	619	496	2391	co	96	2.1	co	30	9.8	sp	17	2.1
8	6	1027	3004	25479	co	90	62	tl	*	*	sp	17	379

Constraint (2.36b) ensures the Hessian matrix $\nabla^2 f(x)$ of polynomials is PSD at every point $x \in \mathcal{D}$, which is equivalent to convexity of f over $\mathcal{D}$. To find a feasible solution for (2.36), we build an SOS formulation. The polynomial variable, represented in an interpolant basis with the optimization variable $\rho \in \mathbb{R}^U$, has degree $2k$ and $U = \binom{m+2k}{m}$ coefficients. Each polynomial entry of $\nabla^2 f(x)$ has degree $2k-2$ and $\bar{U} = \binom{m+2k-2}{m}$ coefficients. Following the descriptions in Sections 2.3.5 to 2.3.6, we obtain interpolation points and a Lagrange polynomial basis for these U-dimensional and $\bar{U}$-dimensional spaces, and we define the matrix $B \in \mathbb{R}^{N \times U}$ containing evaluations of the U-dimensional Lagrange basis at the N feature observations. Finally, we let $F \in \mathbb{R}^{\mathrm{sd}(m)\bar{U} \times U}$ be such that $F\rho$ is a vectorization of the tensor $H \in \mathbb{R}^{m \times m \times \bar{U}}$ (scaled to account for symmetry) with $H_{a,b,u}$ equal to the uth coefficient of the (a,b)th polynomial in $\nabla^2 f(x)$ for $a, b \in [\![m]\!]$ and $u \in [\![\bar{U}]\!]$. This yields the formulation:

$$\min_{\psi \in \mathbb{R}, \rho \in \mathbb{R}^U} \quad \psi : \tag{2.37a}$$

$$(\psi, g - B\rho) \in \mathcal{K}_{\ell_2}, \tag{2.37b}$$

$$F\rho \in \mathcal{K}_{\mathrm{matSOS}(P)}. \tag{2.37c}$$

Note that for $N > U$, we use a QR factorization to reduce the dimension of $\mathcal{K}_{\ell_2}$ in (2.37b) from $1+N$ to $2+U$. Let $[-B\ g] = QR$, where $Q \in \mathbb{R}^{N \times (U+1)}$ has orthonormal columns and $R \in \mathbb{R}^{(U+1)\times(U+1)}$ is upper triangular. Then $(\psi, g - B\rho) \in \mathcal{K}_{\ell_2}$ if and only if $(\psi, R(\rho, 1)) \in \mathcal{K}_{\ell_2}$. The EF for NF constraint (2.37c) follows (2.8a).

We generate random instances of (2.37) for varying m and k, with $N = \lceil 1.1U \rceil$. We exclude the case $m = 1$, since $\mathcal{K}_{\text{SOS}(P)}$ could be used in place of $\mathcal{K}_{\text{matSOS}(P)}$. Our results are summarized in Table 2.11. Note that $\nu = \bar{\nu}$, $p = 0$, $\bar{p} = q - n - 1$, $\bar{q} = \bar{n} + 1$. All dimensions are larger for the EFs. The instances are numerically challenging, and MO-EF often encounters slow progress.

Table 2.11: Shape constrained regression formulation and solver statistics.

		form. stats.				Hyp-NF			Hyp-EF			MO-EF		
m	$2k$	ν	n	$\bar{n}$	q	st	it	time	st	it	time	st	it	time
2	10	72	67	952	203	co	18	0.0	co	24	0.6	sp	19	0.4
2	20	292	232	14527	803	co	37	0.5	sp	68	1348	sp	20	60
2	30	662	497	73727	1803	co	58	5.6	rl	*	*	tl	12	1845
2	40	1182	862	*	3203	co	84	34	m	*	*	sk	*	*
2	50	1852	1327	*	5003	er	52	161	sk	*	*	sk	*	*
2	60	2672	1892	*	7203	er	120	939	sk	*	*	sk	*	*
3	8	152	166	3391	671	co	19	0.1	co	27	14	sp	23	4.3
3	12	485	456	31347	2173	co	38	3.5	tl	*	*	sp	15	244
3	16	1118	970	161584	5051	co	61	44	m	*	*	rl	*	*
3	20	2147	1772	*	9753	co	87	325	sk	*	*	sk	*	*
3	24	3668	2926	*	16727	co	111	1605	sk	*	*	sk	*	*
4	6	142	211	2881	912	co	17	0.4	co	23	7.8	sp	18	3.7
4	8	382	496	17686	2597	co	24	3.1	co	38	1621	sp	14	94
4	10	842	1002	79822	5953	co	38	35	rl	*	*	tl	12	2068
4	12	1626	1821	*	11832	co	58	283	m	*	*	sk	*	*
4	14	2858	3061	*	21262	co	72	1430	sk	*	*	sk	*	*
6	4	80	211	1240	800	co	13	0.3	co	15	0.7	co	9	0.6
6	6	422	925	20539	5336	co	22	15	co	32	1630	sp	17	260
6	8	1514	3004	215440	22409	co	29	638	m	*	*	rl	*	*
8	4	138	496	3412	2117	co	19	2.1	co	20	8.0	co	11	4.1
8	6	938	3004	89008	20825	co	31	515	rl	*	*	rl	*	*
10	4	212	1002	7657	4633	co	26	13	co	24	87	co	13	25
12	4	302	1821	15003	8920	co	29	73	co	28	504	co	11	125
14	4	408	3061	26686	15662	co	33	346	tl	0	1884	sp	21	767

2.4 Discussion of results

Although many convex problems are representable with conic EFs using the small number of standard cones currently recognized by some advanced conic solvers, these formulations can be much larger and more complex than NFs with exotic cones. In Section 2.2, we describe some of Hypatia's predefined exotic cones and analyze general techniques for constructing EFs from NFs that use these cones. For several example problems in Section 2.3, we propose NFs and generate instances of a wide range of sizes. Across these instances, we observe much higher empirical dimensions (variable, equality, and conic constraint dimensions in the conic general form (2.1)) for the EFs than for the NFs. We demonstrate significant computational advantages from solving the NFs with Hypatia compared to solving the EFs with either Hypatia or MOSEK 9, especially in terms of solve time and memory usage. We also observe that the NFs are typically faster and less memory-intensive to generate using

JuMP.

When there exists an NF that is significantly smaller or easier to model than any EF, it is probably worth trying Hypatia. In deciding whether to formulate an NF or an EF, it can be helpful to examine our summary in Table 2.1 of computational properties for NFs and EFs of exotic cone constraints. For spectral and nuclear norm constraints, when the matrix ($W \in \mathbb{R}^{d_1 \times d_2}$) has many more columns than rows ($d_2 \gg d_1$), the dimensions look relatively more favorable for the NF. For SOS and SOS matrix constraints, the dimensions grow much more slowly for the NF as the polynomial degree increases. Sometimes the modeler has to choose between different EFs. For our matrix completion problem and experiment design root-determinant variant, we compare two EFs for the geometric mean cone and find that Hypatia performs better with the exponential cone EF (*EF-exp*) and MOSEK performs better with the second order cone EF (*EF-sec*).

If the modeler has an NF that uses an exotic cone not already defined in Hypatia, the user can add support for the cone through Hypatia's generic cone interface. It may require some effort to make the cone oracles as efficient and numerically stable as possible. In Sections 2.5 and 2.6, we describe oracle implementations for the PSD slice cone class from Section 2.2.1 and the infinity/spectral norm cone class from Section 2.2.2, and in Chapter 3, we propose new LHSCBs and oracles for the spectral function cone class from Section 2.2.3.

2.5 Oracles for positive semidefinite slice cones

In this section, we define LHSCBs and derive efficient oracle procedures for cones that can be characterized as intersections of slices of the PSD cone. We introduce examples of these cones and their dual cones in Section 2.2.1.

First, we consider a proper cone $\mathcal{K} \subset \mathbb{R}^q$ that is an inverse linear image (or slice) of the PSD cone $\mathbb{S}^o_\succeq$ of side dimension o. Suppose:

$$\mathcal{K} \coloneqq \{s \in \mathbb{R}^q : \Lambda(s) \succeq 0\}, \tag{2.38}$$

where $\Lambda : \mathbb{R}^q \to \mathbb{S}^o$ is a linear operator, with adjoint linear operator $\Lambda^* : \mathbb{S}^o \to \mathbb{R}^q$. Then the dual cone can be characterized as:

$$\mathcal{K}^* \coloneqq \{s \in \mathbb{R}^q : \exists S \succeq 0, s = \Lambda^*(S)\}. \tag{2.39}$$

We note that for $\mathcal{K}_\succeq$ (the self-dual vectorized PSD cone), we can let $q = \text{sd}(o)$, $\Lambda(s) = \text{mat}(s)$, and $\Lambda^*(S) = \text{vec}(S)$. Given a point $s \in \mathbb{R}^q$, strict feasibility for $\mathcal{K}$ can be checked, for example, by attempting a Cholesky factorization $\Lambda(s) = LL'$, where L is lower triangular.

For $\mathcal{K}$ we have the LHSCB $f(s) = -\operatorname{logdet}(\Lambda(s))$ with parameter $\nu = o$. Given a point $s \in \text{int}(\mathcal{K})$, we have $\Lambda(s) \in \mathbb{S}^o_\succ$ and its inverse $\Lambda^{-1}(s) \in \mathbb{S}^o_\succ$. For a direction $\delta \in \mathbb{R}^q$, for f at s we can write the gradient, and the Hessian and third order oracle (TOO, defined in (1.5)) applied to δ, as:

$$g(s) = -\Lambda^*(\Lambda^{-1}(s)), \tag{2.40a}$$

$$H(s)\delta = \Lambda^*(\Lambda^{-1}(s)\Lambda(\delta)\Lambda^{-1}(s)), \tag{2.40b}$$
$$\mathrm{T}(s,\delta) = \Lambda^*(\Lambda^{-1}(s)\Lambda(\delta)\Lambda^{-1}(s)\Lambda(\delta)\Lambda^{-1}(s)). \tag{2.40c}$$

Note the expressions for g and H are from Papp and Yıldız (2019, Section 3). If we have, for example, a Cholesky factorization $\Lambda(s) = LL'$ (computed during the feasibility check), then the oracles in (2.40) are easy to compute if Λ and Λ^* are easy to apply. We can compute the TOO (2.40c) using the following steps:

$$Y \coloneqq L^{-1}\Lambda(\delta)\Lambda^{-1}(s), \tag{2.41a}$$
$$Z \coloneqq Y'Y = \Lambda^{-1}(s)\Lambda(\delta)\Lambda^{-1}(s)\Lambda(\delta)\Lambda^{-1}(s), \tag{2.41b}$$
$$\mathrm{T}(s,\delta) = \Lambda^*(Z). \tag{2.41c}$$

We note (2.41a) can be computed using back-substitutions with L, and (2.41b) is a simple symmetric outer product. We use this approach to derive simple TOO procedures for $\mathcal{K}_{\mathrm{LMI}}$ in Section 2.5.1 and for $\mathcal{K}^*_{\mathrm{SOS}}$ and $\mathcal{K}^*_{\mathrm{matSOS}}$ in Section 2.5.2 when $r = 1$.

Now we consider the more general case of a cone $\mathcal{K}$ that can be characterized as an intersection of slices of PSD cones, for example $\mathcal{K}^*_{\mathrm{SOS}}$ and $\mathcal{K}^*_{\mathrm{matSOS}}$ when $r > 1$. Suppose:

$$\mathcal{K} \coloneqq \{s \in \mathbb{R}^q : \Lambda_l(s) \succeq 0, \forall l \in [\![r]\!]\}, \tag{2.42}$$

where $\Lambda_l : \mathbb{R}^q \to \mathbb{S}^{o_l}$, for $l \in [\![r]\!]$. Then the dual cone can be characterized as:

$$\mathcal{K}^* \coloneqq \left\{s \in \mathbb{R}^q : \exists S_1, \ldots, S_r \succeq 0, s = \textstyle\sum_{l\in[\![r]\!]} \Lambda^*_l(S_l)\right\}. \tag{2.43}$$

Feasibility for $\mathcal{K}$ can be checked by performing r Cholesky factorizations. If we let $f_l(s) = -\operatorname{logdet}(\Lambda_l(s)), \forall l \in [\![r]\!]$, then $f(s) = \sum_{l\in[\![r]\!]} f_l(s)$ is an LHSCB for $\mathcal{K}$ with parameter $\nu = \sum_{l\in[\![r]\!]} o_l$. Clearly, $g(s)$, $H(s)\delta$ (and the explicit Hessian matrix), and $\mathrm{T}(s,\delta)$ can all be computed as sums over $l \in [\![r]\!]$ of the terms in (2.40c).

2.5.1 Linear matrix inequality cone

We denote the inner product of $X, Y \in \mathbb{S}^s$ as $\langle X, Y\rangle = \operatorname{tr}(XY) \in \mathbb{R}$, computable in $\mathcal{O}(s^2)$ time. For $\mathcal{K}_{\mathrm{LMI}}$ parametrized by $P_i \in \mathbb{S}^s, \forall i \in [\![d]\!]$, we define for $w \in \mathbb{R}^d$ and $W \in \mathbb{S}^s$:

$$\Lambda(w) \coloneqq \textstyle\sum_{i\in[\![d]\!]} w_i P_i \in \mathbb{S}^s, \tag{2.44a}$$
$$\Lambda^*(W) \coloneqq (\langle P_i, W\rangle)_{i\in[\![d]\!]} \in \mathbb{R}^d. \tag{2.44b}$$

Our implementation uses specializations of (2.40) and (2.41) for $\mathcal{K}_{\mathrm{LMI}}$. For $w \in \operatorname{int}(\mathcal{K}_{\mathrm{LMI}})$ and direction $\delta \in \mathbb{R}^d$, using the Cholesky factorization $\Lambda(w) = LL'$, we compute:

$$Q_i \coloneqq L^{-1}P_i(L^{-1})' \in \mathbb{S}^s \qquad \forall i \in [\![d]\!], \tag{2.45a}$$
$$g(w) = (-\operatorname{tr}(Q_i))_{i\in[\![d]\!]}, \tag{2.45b}$$

$$R \coloneqq \textstyle\sum_{j\in[\![d]\!]}\delta_j Q_j \in \mathbb{S}^s, \tag{2.45c}$$
$$H(w)\delta = (\langle Q_i, R\rangle)_{i\in[\![d]\!]}, \tag{2.45d}$$
$$\mathrm{T}(w,\delta) = (\langle Q_i, R'R\rangle)_{i\in[\![d]\!]}, \tag{2.45e}$$

and we compute the explicit Hessian oracle as:

$$(H(w))_{i,j} = \langle Q_i, Q_j\rangle \qquad \forall i,j \in [\![d]\!]. \tag{2.46}$$

The symmetric form of Q_i and the use of a symmetric outer product $R'R$ in (2.45e) are beneficial for efficiency and numerical performance.

2.5.2 Polynomial weighted sum-of-squares dual cones

Recall that Hypatia uses LHSCBs for $\mathcal{K}^*_{\mathrm{SOS}}, \mathcal{K}^*_{\mathrm{matSOS}}$, because LHSCBs for $\mathcal{K}_{\mathrm{SOS}}, \mathcal{K}_{\mathrm{matSOS}}$ with tractable oracles are not known (see Kapelevich, Coey, and Vielma (2021)). Since the scalar SOS dual cone $\mathcal{K}^*_{\mathrm{SOS}}$ is a special case of the matrix SOS dual cone $\mathcal{K}^*_{\mathrm{matSOS}}$ with $t = 1$, we only consider $\mathcal{K}^*_{\mathrm{matSOS}}$ here. In general, $\mathcal{K}^*_{\mathrm{matSOS}}$ is an intersection of r slices of $\mathcal{K}_{\succeq}$ (see (2.42)), so the gradient, Hessian, and TOO oracles are all additive; for simplicity, we only consider $r = 1$ (and $s_1 = s$, $P_1 = P$) below.

To enable convenient vectorization, we define $\rho_{i,j}$ for indices $i, j \geq 1$ as:

$$\rho_{i,j} \coloneqq \begin{cases} 1 & \text{if } i = j, \\ \sqrt{2} & \text{otherwise.} \end{cases} \tag{2.47}$$

For $\mathcal{K}^*_{\mathrm{matSOS}}$ parametrized by $P \in \mathbb{R}^{d\times s}$ and $t \geq 1$, we define for $w \in \mathbb{R}^{\mathrm{sd}(t)d}$ and $W \in \mathbb{S}^{st}$:

$$\Lambda(w) \coloneqq \left[P' \operatorname{Diag}\left(\rho_{i,j}^{-1} w_{\max(i,j),\min(i,j),:}\right) P\right]_{i,j\in[\![t]\!]} \in \mathbb{S}^{st}, \tag{2.48a}$$
$$\Lambda^*(W) \coloneqq (\rho_{i,j} \operatorname{diag}(P(W)_{i,j}P'))_{i\in[\![t]\!],j\in[\![i]\!]} \in \mathbb{R}^{\mathrm{sd}(t)d}, \tag{2.48b}$$

where $w = (w_{i,j,:})_{i\in[\![t]\!],j\in[\![i]\!]}$ and $w_{i,j,:} \in \mathbb{R}^d$ is the contiguous slice of w corresponding to the interpolant basis values in the (i,j)th (lower triangle) position, matrix $(S)_{i,j}$ is the (i,j)th block in a block matrix S (with blocks of equal dimension), and $[S_{i,j}]_{i,j\in[\![t]\!]}$ is the symmetric block matrix with matrix $S_{i,j}$ in the (i,j)th block.

We implement efficient and numerically stable specializations of the oracles in (2.40) and (2.41). Suppose we have $w \in \operatorname{int}(\mathcal{K}^*_{\mathrm{matSOS}})$ and direction $\delta \in \mathbb{R}^{\mathrm{sd}(t)d}$, and a Cholesky factorization $\Lambda(w) = LL'$. For each $i, j \in [\![t]\!] : i \geq j$ and $p \in [\![d]\!]$, we implicitly compute oracles according to:

$$(Q)_{i,j,p} \coloneqq ((L^{-1})_{i,j}P')e_p \in \mathbb{R}^s, \tag{2.49a}$$
$$(g(w))_{i,j,p} = -\rho_{i,j} Q'_{i,:,p} Q_{:,j,p}, \tag{2.49b}$$
$$(R)_{i,j,p} \coloneqq (L^{-1}\Lambda(\delta)(L^{-1})'Q)_{i,j}e_p \in \mathbb{R}^s, \tag{2.49c}$$
$$(H(w)\delta)_{i,j,p} = \rho_{i,j} Q'_{i,:,p} R_{:,j,p}, \tag{2.49d}$$

$$(\mathrm{T}(w,\delta))_{i,j,p} = \rho_{i,j} R'_{i,:,p} R_{:,j,p}. \tag{2.49e}$$

Letting $Q^2_{i,j} \coloneqq (Q'Q)_{i,j} \in \mathbb{S}^d$, we compute the Hessian oracle according to:

$$(H(w))_{(i,j,:),(k,l,:)} = \tfrac{1}{2}\rho_{i,j}\rho_{k,l}\big(Q^2_{i,k} \odot Q^2_{j,l} + Q^2_{i,l} \odot Q^2_{j,k}\big) \in \mathbb{S}^d \qquad \forall i,j,k,l \in [\![t]\!], \tag{2.50}$$

where $X \odot Y \in \mathbb{S}^d$ denotes the Hadamard (elementwise) product of $X, Y \in \mathbb{S}^d$.

2.5.3 Sparse positive semidefinite cone

Let $\mathcal{S} = ((i_l, j_l))_{l \in [\![d]\!]}$ be a collection of row-column index pairs defining the sparsity pattern of the lower triangle of a symmetric matrix of side dimension s (including all diagonal elements). We do not require $\mathcal{S}$ to be a chordal sparsity pattern (unlike M. S. Andersen, Dahl, and Vandenberghe (2013) and Burer (2003)), as this restriction is not necessary for the oracles Hypatia uses. Note $s \leq d \leq \mathrm{sd}(s)$. For $\mathcal{K}_{\mathrm{sPSD}}$ parametrized by $\mathcal{S}$, we define $\Lambda : \mathbb{R}^d \to \mathbb{S}^s$ as the linear operator satisfying, for all $i, j \in [\![s]\!] : i \geq j$:

$$(\Lambda(w))_{i,j} \coloneqq \begin{cases} \rho_{i,j}^{-1} w_l & \text{if } i = i_l = j = j_l, \\ 0 & \text{otherwise,} \end{cases} \tag{2.51}$$

where $\rho_{i,j}$ is given by (2.47). Then Λ^* is the vectorized projection onto $\mathcal{S}$, i.e. for $W \in \mathbb{S}^s$:

$$\Lambda^*(W) \coloneqq (\rho_{i,j} W_{i,j})_{(i,j) \in \mathcal{S}} \in \mathbb{R}^d. \tag{2.52}$$

Consider $w \in \mathrm{int}(\mathcal{K}_{\mathrm{sPSD}})$ and direction $\delta \in \mathbb{R}^d$. The gradient (2.40a) and Hessian product (2.40b) for $\mathcal{K}_{\mathrm{sPSD}}$ can be computed using M. S. Andersen, Dahl, and Vandenberghe (2013, Algorithms 4.1 and 5.1). To derive the TOO, we use the fact that:

$$-2\mathrm{T}(w,\delta) = \nabla^3 f(w)[\delta,\delta] = \frac{d^2}{dt^2} \nabla f(w + t\delta)\big|_{t=0}. \tag{2.53}$$

In order to succinctly describe our TOO approach as an extension of the procedures in M. S. Andersen, Dahl, and Vandenberghe (2013), we describe an approach based on a sparse LDL factorization of $\Lambda(w)$. However, our current implementation in Hypatia uses a sparse Cholesky (LL') factorization, which is very similar to the LDL-based approach here. We compute the sparse Cholesky factors using Julia's SuiteSparse wrapper of CHOLMOD (Y. Chen et al., 2008). We note that Hypatia implements a supernodal generalization (see M. S. Andersen, Dahl, and Vandenberghe (2013, Section 7)) of the TOO procedure we describe below. Before we describe the TOO procedure, we repeat useful definitions from M. S. Andersen, Dahl, and Vandenberghe (2013), define higher order derivative terms, and differentiate several equations that are used for the gradient and Hessian oracles. As discussed in Section 1.8, Hypatia computes the feasibility check and gradient oracles before the TOO, and our TOO procedure reuses cached values computed for these oracles.

We define:

$$R \coloneqq \Lambda(\nabla f(w + t\delta)). \tag{2.54}$$

Let $LDL' = \Lambda(w)$ be a sparse LDL factorization, i.e. L is a sparse unit lower triangular matrix and D is a positive definite diagonal matrix. The sparsity pattern of L is associated with an elimination tree (M. S. Andersen, Dahl, and Vandenberghe, 2013, Section 2), and each column of L corresponds to a node of this tree. Let $\mathcal{I}_k$ be the ordered row indices of nonzeros below the diagonal in column k of L, and let $\mathcal{J}_k = \mathcal{I}_k \cup \{k\}$. Let $\mathrm{ch}(i)$ denote the children of node i in the tree. For an index set $\mathcal{I}$ let $\mathcal{I}(i)$ denote the ith element. For index sets $\mathcal{J} \subset \mathcal{I}$, we define $E_{\mathcal{I},\mathcal{J}} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{J}|}$ satisfying, $i \in [\![|\mathcal{I}|]\!]$, $j \in [\![|\mathcal{J}|]\!]$:

$$(E_{\mathcal{I},\mathcal{J}})_{i,j} := \begin{cases} 1 & \text{if } \mathcal{I}(i) = \mathcal{J}(j), \\ 0 & \text{otherwise.} \end{cases} \tag{2.55}$$

Let U_i be the update matrix for node i (see M. S. Andersen, Dahl, and Vandenberghe (2013, Equation 14)):

$$U_i := -\textstyle\sum_{k\in\mathrm{ch}(i)\cup\{i\}} D_{k,k} L_{\mathcal{I}_i,k} L'_{\mathcal{I}_i,k}. \tag{2.56}$$

Let $\dot{D}$, $\dot{L}$, $\dot{U}$, $\dot{R}$ and $\ddot{D}$, $\ddot{L}$, $\ddot{U}$, $\ddot{R}$ denote the first and second derivatives of D, L, U, R with respect to the linearization variable t in (2.53). For convenience, we let:

$$\bar{L}_j := \begin{bmatrix} 1 & 0 \\ -L_{\mathcal{I}_j,j} & I(d) \end{bmatrix}. \tag{2.57}$$

Suppose we have computed $\dot{D}$, $\dot{L}$, $\dot{U}$ according to M. S. Andersen, Dahl, and Vandenberghe (2013, Equation 30). Differentiating M. S. Andersen, Dahl, and Vandenberghe (2013, Equation 30) once with respect to t gives:

$$\begin{bmatrix} \ddot{D}_{j,j} & P'_j \\ P_j & 2D_{j,j}\dot{L}_{\mathcal{I}_j,j}\dot{L}'_{\mathcal{I}_j,j} + \ddot{U}_j \end{bmatrix} = \bar{L}_j \big(\textstyle\sum_{i\in\mathrm{ch}(j)} E_{\mathcal{J}_j,\mathcal{I}_i} \ddot{U}_i E'_{\mathcal{J}_j,\mathcal{I}_i}\big) \bar{L}'_j, \tag{2.58}$$

where $P_j := 2\dot{D}_{j,j}\dot{L}_{\mathcal{I}_j,j} + D_{j,j}\ddot{L}_{\mathcal{I}_j,j}$ for convenience. This allows us to compute $\ddot{D}$, $\ddot{L}$, $\ddot{U}$. M. S. Andersen, Dahl, and Vandenberghe (2013, Equations 21 and 22) show that:

$$R_{\mathcal{I}_j,j} = -R_{\mathcal{I}_j,\mathcal{I}_j} L_{\mathcal{I}_j,j}, \tag{2.59a}$$

$$\begin{bmatrix} R_{j,j} & R'_{\mathcal{I}_j,j} \\ R_{\mathcal{I}_j,j} & R_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix} \begin{bmatrix} 1 \\ L_{\mathcal{I}_j,j} \end{bmatrix} = \begin{bmatrix} D_{j,j}^{-1} \\ 0 \end{bmatrix}, \tag{2.59b}$$

for each node j. Differentiating (2.59a) once with respect to t gives:

$$\dot{R}_{\mathcal{I}_j,j} = -R_{\mathcal{I}_j,\mathcal{I}_i}\dot{L}_{\mathcal{I}_j,j} - \dot{R}_{\mathcal{I}_j,\mathcal{I}_j} L_{\mathcal{I}_j,j}. \tag{2.60}$$

Differentiating (2.59b) twice and substituting (2.59a) and (2.60), we have:

$$\begin{bmatrix} \ddot{R}_{j,j} & \ddot{R}'_{\mathcal{I}_j,j} \\ \ddot{R}_{\mathcal{I}_j,j} & \ddot{R}_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix} = \bar{L}'_j \begin{bmatrix} 2\dot{D}^2_{j,j}D^{-3}_{j,j} - \ddot{D}_{j,j}D^{-2}_{j,j} + 2\dot{L}'_{\mathcal{I}_j,j}R_{\mathcal{I}_j,\mathcal{I}_j}\dot{L}_{\mathcal{I}_j,j} & Q'_j \\ Q_j & \ddot{R}_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix} \bar{L}_j, \tag{2.61}$$

where $Q_j \coloneqq -R_{\mathcal{I}_j,\mathcal{I}_j}\ddot{L}_{\mathcal{I}_j,j} - 2\dot{R}_{\mathcal{I}_j,\mathcal{I}_j}\dot{L}_{\mathcal{I}_j,j}$ for convenience. This allows us to compute $\ddot{R}$. Finally, by (2.53) and (2.54), we can compute the TOO as:

$$-2\mathrm{T}(w,\delta) = \Lambda^*\big(\ddot{R}\big). \tag{2.62}$$

We now write the high-level TOO procedure. For convenience, we let:

$$\Delta = \Lambda(\delta) \in \mathbb{S}^s. \tag{2.63}$$

Following E. D. Andersen, Roos, and Terlaky (2003), we define K and M as sparse matrices with the same structure as L, satisfying for all $j \in [\![s]\!]$:

$$K_{j,j} = \dot{D}_{j,j}, \tag{2.64a}$$
$$K_{\mathcal{I}_j,j} = D_{j,j}\dot{L}_{\mathcal{I}_j,j}, \tag{2.64b}$$
$$M_{j,j} = D^{-2}_{j,j}K_{j,j}, \tag{2.64c}$$
$$M_{\mathcal{I}_j,j} = D^{-1}_{j,j}R_{\mathcal{I}_j,\mathcal{I}_j}K_{\mathcal{I}_j,j}. \tag{2.64d}$$

The first three steps in the TOO procedure below compute $\dot{D}$, $\dot{L}$, $\dot{U}$, and $\dot{R}$ and are identical to steps in M. S. Andersen, Dahl, and Vandenberghe (2013, Algorithm 5.1).

1. Iterate over $j \in [\![s]\!]$ in topological order, computing $K_{\mathcal{J}_j,j}$ and $\dot{U}_j$ according to:

$$\begin{bmatrix} K_{j,j} & K'_{\mathcal{I}_j,j} \\ K_{\mathcal{I}_j,j} & U'_j \end{bmatrix} = \bar{L}_j \left(\begin{bmatrix} \Delta_{j,j} & \Delta'_{\mathcal{I}_j,j} \\ \Delta_{\mathcal{I}_j,j} & 0 \end{bmatrix} + \textstyle\sum_{i\in\mathrm{ch}(j)} E_{\mathcal{J}_j,\mathcal{I}_i}U'_iE'_{\mathcal{J}_j,\mathcal{I}_i} \right) \bar{L}'_j. \tag{2.65}$$

2. For $j \in [\![s]\!]$, store $\dot{D}_{j,j}$ and $\dot{L}_{\mathcal{I}_j,j}$ from (2.64a) and (2.64b), and compute $M_{\mathcal{J}_j,j}$ from (2.64c) and (2.64d).

3. Iterate over $j \in [\![s]\!]$ in reverse topological order, computing $\dot{R}_{\mathcal{J}_j,j}$ according to:

$$\begin{bmatrix} \dot{R}_{j,j} & \dot{R}'_{\mathcal{I}_j,j} \\ \dot{R}_{\mathcal{I}_j,j} & \dot{R}_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix} = \bar{L}'_j \begin{bmatrix} M_{j,j} & M'_{\mathcal{I}_j,j} \\ M_{\mathcal{I}_j,j} & \dot{R}_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix} \bar{L}_j, \tag{2.66}$$

and updating matrices $\dot{R}_{\mathcal{I}_j,\mathcal{I}_j}$ for each child $i \in \mathrm{ch}(j)$ of vertex j according to:

$$\dot{R}_{\mathcal{I}_i,\mathcal{I}_i} = E'_{\mathcal{J}_j,\mathcal{I}_i} \begin{bmatrix} \dot{R}_{j,j} & \dot{R}'_{\mathcal{I}_j,j} \\ \dot{R}_{\mathcal{I}_j,j} & \dot{R}_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix} E_{\mathcal{J}_j,\mathcal{I}_i}. \tag{2.67}$$

4. Iterate over $j \in [\![s]\!]$ in topological order, computing $\ddot{D}_{j,j}, \ddot{L}_{\mathcal{I}_j,j}, \ddot{U}_j$ from (2.58).

5. Iterate over $j \in [\![s]\!]$ in reverse topological order, computing $\ddot{R}_{j,j}$, $\ddot{R}_{\mathcal{I}_j,j}$, $\ddot{R}_{\mathcal{I}_j,\mathcal{I}_j}$ from (2.61).

6. Compute $\mathrm{T}(w, \delta)$ using $\ddot{R}$ and (2.62).

2.6 Oracles for infinity/spectral norm cones

In this section, we derive efficient and numerically stable oracle procedures for infinity/spectral norm cones. We discuss these cones and their dual cones (epigraphs of one/nuclear norms) in Section 2.2.2. The real/complex vector infinity norm cone $\mathcal{K}_{\ell_\infty}$ and the symmetric/Hermitian matrix spectral norm cone $\mathcal{K}_{\ell_{\mathrm{sspec}}}$ are slices of the most general case: the real/complex rectangular matrix spectral norm cone $\mathcal{K}_{\ell_{\mathrm{spec}}}$. Furthermore, the LHSCBs and oracle procedures for $\mathcal{K}_{\ell_\infty}$ and $\mathcal{K}_{\ell_{\mathrm{sspec}}}$ are specializations of those for $\mathcal{K}_{\ell_{\mathrm{spec}}}$. For simplicity, in this section we mainly focus on the real $\mathcal{K}_{\ell_{\mathrm{spec}}}$ case; the complex $\mathcal{K}_{\ell_{\mathrm{spec}}}$ case is almost identical.

The oracles we derive for $\mathcal{K}_{\ell_{\mathrm{spec}}}$ are all based on a single factorization: a thin singular value decomposition (SVD) of the matrix. These oracles are all analytic, i.e. closed form in terms of the SVD. As we discuss briefly in Section 1.6, Hypatia's QR-Cholesky linear system solver does not require explicit (inverse) Hessian matrices, only (inverse) Hessian products applied implicitly to arrays. A particularly important result is our derivation of a closed form inverse Hessian product formula in Section 2.6.4.

2.6.1 Barrier functions

Suppose the matrix W has d rows and s columns, and $d \leq s$. Let $\sigma_i(W) \geq 0$ be the ith largest singular value of W. Recall the real rectangular spectral norm cone definition from (2.13a):

$$\mathcal{K}_{\ell_{\mathrm{spec}}(d,s)} \coloneqq \{(u, w) \in \mathbb{R}_{\geq} \times \mathbb{R}^{ds} : u \geq \sigma_1(W)\}, \tag{2.68}$$

where $W \coloneqq \mathrm{mat}_{d,s}(w) \in \mathbb{R}^{d \times s}$. The complex W case is a simple generalization. We use the LHSCB from Nesterov and Nemirovski (1994) with $\nu = 1 + d$:

$$\begin{aligned} f(u, w) &= -\log(u) - \mathrm{logdet}(uI(d) - WW'/u) &\text{(2.69a)}\\ &= -\log(u) - \textstyle\sum_{i \in [\![d]\!]} \log(u - (\sigma_i(W))^2/u). &\text{(2.69b)} \end{aligned}$$

Note that for the slice $\mathcal{K}_{\ell_\infty}$ of $\mathcal{K}_{\ell_{\mathrm{spec}}}$, this LHSCB reduces to our $\mathcal{K}_{\ell_\infty}$ LHSCB from Güler (1996, Section 7.5) with $\nu = 1 + d$:

$$f(u, w) = -\log(u) - \textstyle\sum_{i \in [\![d]\!]} \log(u - |w_i|^2/u). \tag{2.70}$$

For $\mathcal{K}_{\ell_{\mathrm{sspec}}}$, the LHSCB has the same form (2.69) but W is symmetric or Hermitian. We are not aware of any LHSCBs with smaller parameters for these cones. Note the central initial interior point oracle (defined in Section 1.3) for these cones is $(\sqrt{\nu}, 0)$.

2.6.2 Feasibility checks

Suppose we need to check strict primal or dual cone feasibility for a point (u, w). First we check the simplest condition - that the epigraph variable u is positive. For $\mathcal{K}_{\ell_\infty}$, the norms are cheap to compute, so the feasibility checks are straightforward. For $\mathcal{K}_{\ell_{\text{sspec}}}$ and $\mathcal{K}_{\ell_{\text{spec}}}$, we can of course check feasibility by computing an SVD of W and accessing the singular values. However, to improve the efficiency of the feasibility checks, we start by checking certain easily-computable bounds on the spectral and nuclear norms.

For the spectral norm cone, we first compute a lower bound b^l on the spectral norm of W:

$$b^l \coloneqq \max\big(\max(\|w\|, \|w\|_1)/\sqrt{d}, \|w\|_\infty/\sqrt{s}\big). \tag{2.71}$$

If $b^l \geq u$, then (u, w) is not strictly feasible. Otherwise, we compute an upper bound on the spectral norm:

$$b^u \coloneqq \min\big(\|w\|, \sqrt{\|w\|_1\|w\|_\infty}\big). \tag{2.72}$$

If $b^u < u$, the point is strictly feasible. Otherwise, we perform a Cholesky factorization of $uI(d) - WW'/u$, which succeeds if and only if the point is strictly feasible. This check tends to be faster than the SVD-based check. Finally, if the point is strictly feasible, we compute the SVD of W. This factorization is cached and reused to compute all LHSCB oracles.

For the nuclear norm cone, we first check whether $\sqrt{d}\|w\| < u$, in which case (u, w) is strictly feasible. Next, we check whether $\|w\| \geq u$, in which case the point is not strictly feasible. Otherwise, we compute the nuclear norm of W by summing the squareroots of the eigenvalues of WW', since computing the eigendecomposition of WW' tends to be faster than computing the SVD of W.

2.6.3 Directional derivatives

Suppose we have a strictly feasible point $\tilde{u} \coloneqq (u, w) \in \operatorname{int}(\mathcal{K}_{\ell_{\text{spec}}})$. Let $W = U \operatorname{Diag}(\sigma) V'$ be the thin SVD of W, where $\sigma \in \mathbb{R}^r_{\geq}$ are the singular values and $U \in \mathbb{R}^{d\times d}$, $V \in \mathbb{R}^{s\times d}$ are orthogonal matrices i.e. $UU' = U'U = V'V = I(d)$ (note VV' is not the identity in general). For convenience, we define:

$$\mu_i \coloneqq u^{-1}\sigma_i, \tag{2.73a}$$

$$\zeta_i \coloneqq \tfrac{1}{2}(u - \mu_i\sigma_i), \tag{2.73b}$$

$$\tau_i \coloneqq \nabla_u \zeta_i = \tfrac{1}{2}(1 + \mu_i^2) = 1 - u^{-1}\zeta_i, \tag{2.73c}$$

and note:

$$\nabla_{\sigma_i}\zeta_i = -\mu_i. \tag{2.74}$$

Suppose $\tilde{p} \coloneqq (p, r)$ is a direction in the ambient space of $\tilde{u}$, i.e. $p \in \mathbb{R}$ and $R = \operatorname{mat}_{d,s}(r) \in \mathbb{R}^{d\times s}$. Let:

$$\hat{R} \coloneqq U'RV \in \mathbb{R}^{d\times d}, \tag{2.75a}$$

$$\bar{Z} \coloneqq \operatorname{Diag}((\zeta_i^{-1})_{i\in[\![d]\!]}) = (\operatorname{Diag}(\zeta))^{-1}, \tag{2.75b}$$

$$M \coloneqq \mathrm{Diag}(\mu), \tag{2.75c}$$
$$S_1 \coloneqq pI(d) - \tfrac{1}{2}(\hat{R}M + M\hat{R}') \in \mathbb{S}^r, \tag{2.75d}$$
$$S_2 \coloneqq \tfrac{1}{2}u^{-1}(p^2 I(d) - \hat{R}\hat{R}') - S_1 \bar{Z} S_1 \in \mathbb{S}^r. \tag{2.75e}$$

Note for any $S \in \mathbb{R}^{d\times d}$, we have $\langle USV', R\rangle = \langle S, \hat{R}\rangle$.

The components of the gradient $g \coloneqq \nabla f(\tilde{u})$ are:

$$g_u = (d-1)u^{-1} - \textstyle\sum_{i\in[\![d]\!]}\zeta_i^{-1}, \tag{2.76a}$$
$$g_W = UM\bar{Z}V'. \tag{2.76b}$$

Differentiating (2.76), the components of Hessian product $H \coloneqq \nabla^2 f(\tilde{u})[\tilde{p}]$ are:

$$H_u = \nabla_u g_u p + \nabla_u g_W[R] \tag{2.77a}$$
$$= -(d-1)u^{-2}p - \textstyle\sum_{i\in[\![d]\!]}\zeta_i^{-1}(u^{-1}p + \zeta_i^{-1}S_{1,i,i}), \tag{2.77b}$$
$$H_W = \nabla_W g_u p + \nabla_W g_W[R] \tag{2.77c}$$
$$= U\bar{Z}(u^{-1}U'R - S_1 M\bar{Z}V'). \tag{2.77d}$$

Differentiating (2.77), the third order directional derivative $T \coloneqq \nabla^3 f(\tilde{u})[\tilde{p},\tilde{p}]$ is:

$$T_u = \nabla_u H_u p + \nabla_u H_W[R] \tag{2.78a}$$
$$= 2(d-1)u^{-3}p^2 + \textstyle\sum_{i\in[\![d]\!]}2\zeta_i^{-2}(u^{-1}pS_{1,i,i} + S_{2,i,i}), \tag{2.78b}$$
$$T_W = \nabla_W H_u p + \nabla_W H_W[R] \tag{2.78c}$$
$$= -2U\bar{Z}(u^{-1}S_1\bar{Z}U'R + S_2 M\bar{Z}V'). \tag{2.78d}$$

Note Hypatia's third order oracle (TOO) defined in (1.5) is simply a rescaling of T by $-1/2$.

Our implementations of these three (directional) derivative oracles are efficient and numerically stable, using only $\mathcal{O}(ds)$ memory and $\mathcal{O}(d^2 s)$ time.

2.6.4 Inverse Hessian product

We represent the inverse Hessian product as $\bar{H} \coloneqq (\nabla^2 f(\tilde{u}))^{-1}[\tilde{p}]$. For conciseness, we omit a detailed derivation of the following result, which relies on Kronecker products.

For convenience, let:

$$\psi \coloneqq -(d-1)u^{-1} + \textstyle\sum_{i\in[\![d]\!]}(u-\zeta_i)^{-1} > 0, \tag{2.79a}$$
$$\Theta \coloneqq (2(u + u^{-1}\sigma_i\sigma_j)^{-1})_{i,j\in[\![d]\!]} \in \mathbb{S}^r. \tag{2.79b}$$

Then the u component of $\bar{H}$ is:

$$\bar{H}_u = \psi^{-1}u\big(p + \textstyle\sum_{i\in[\![d]\!]}\sigma_i\hat{R}_{i,i}\Theta_{i,i}\big). \tag{2.80}$$

Now let:

$$\Phi \coloneqq (\zeta_i(u - u^{-1}\sigma_i\sigma_j)^{-1})_{i,j\in[\![d]\!]} \in \mathbb{R}^{d\times d}, \tag{2.81a}$$

$$\Delta \coloneqq (\hat{R}\operatorname{Diag}(\sigma)) \odot \Phi \in \mathbb{R}^{d\times d}, \tag{2.81b}$$

$$S \coloneqq (\Delta + \Delta' - \zeta_i^{-1}\bar{H}_u I(d)) \odot \Theta \in \mathbb{S}^r. \tag{2.81c}$$

where $\odot$ denotes the Hadamard (elementwise) product of matrices. Note $\Phi_{i,i} = 1/2, \forall i \in [\![d]\!]$. Then the w component of $\bar{H}$ is:

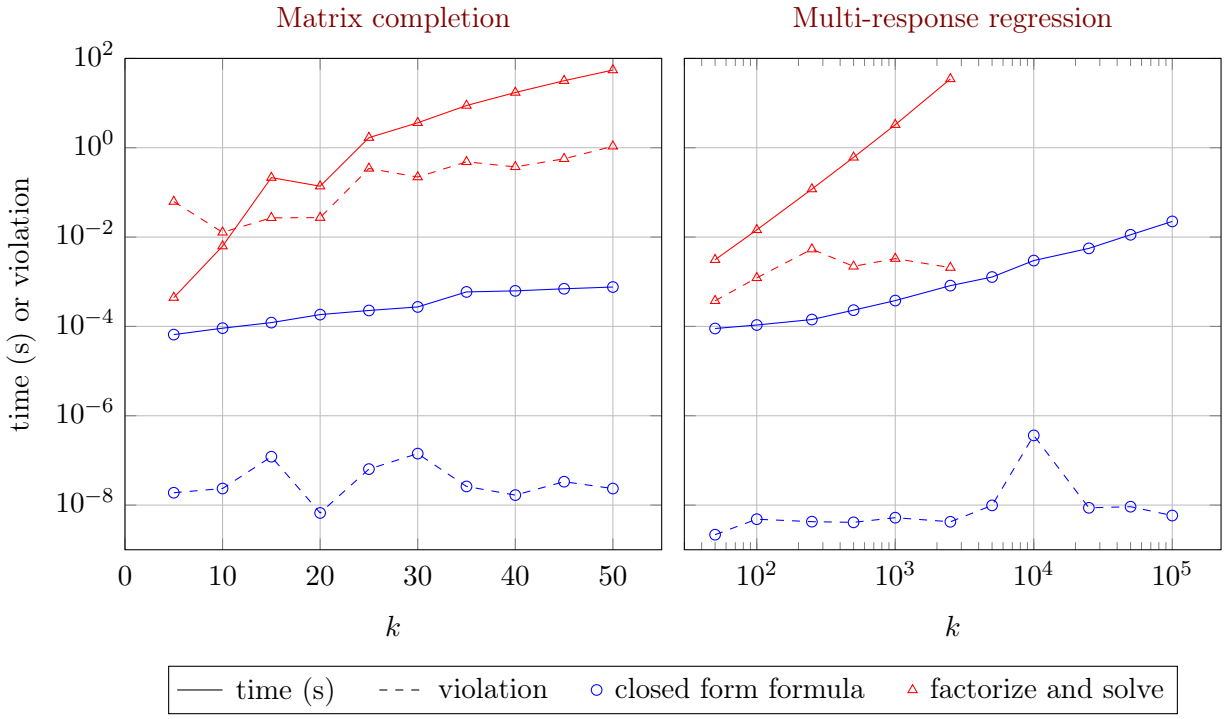
$$\bar{H}_W = U\operatorname{Diag}(\zeta)(uU'R - S\operatorname{Diag}(\sigma)V'). \tag{2.82}$$

The inverse Hessian product oracle $\bar{H}$ given by (2.80) and (2.82) is essentially as easy to compute as the Hessian product oracle H in (2.77). We compare the time and memory complexity of this procedure with that of a naive approach that computes the explicit Hessian matrix, performs a Cholesky factorization, and uses a direct linear solve. We exclude the cost of the SVD, which is computed only once during the primal cone feasibility check and reused. The closed form $\bar{H}$ only uses $\mathcal{O}(ds)$ memory, but the naive approach requires an explicit Hessian, so it uses $\mathcal{O}(d^2s^2)$ memory. The closed form $\bar{H}$, like the closed form Hessian product H, requires only a few matrix multiplications of size at most $d \times d$ by $d \times s$, so it uses $\mathcal{O}(d^2s)$ time. The naive approach requires a Cholesky factorization of the Hessian, which takes $\mathcal{O}(d^3s^3)$ time.

Now we compare the practical performance of these two alternative procedures. Using Hypatia, we first solve NF instances of a range of sizes k for the examples from Section 2.3.2 (with $m = 10$) and Section 2.3.3 (with $l = m = 15$). For each instance, at Hypatia's final IPM iterate, we take the direction $\tilde{p} = g$ (i.e. the gradient oracle at the iterate) and compute $\bar{H}$ for this direction using each procedure. To measure the numerical accuracy of each procedure, we compute $\epsilon \coloneqq |1 - \nu^{-1}\langle \bar{H}, g\rangle|$, which is the violation on a particular identity (Nesterov, Todd, and Ye, 1997, Equation 2.5) satisfied by a logarithmically homogeneous function such as the LHSCB f. We also time each procedure, excluding Hessian memory allocation time for the naive procedure.

Our results are displayed in Figure 2.2. We note that the Cholesky factorization fails numerically (i.e. LAPACK errors) for some sizes; when this occurs, Hypatia uses a Bunch-Kaufman (symmetric LDLT-like) factorization as a fallback. These comparisons demonstrate that our closed form formula allows computing $\bar{H}$ faster and with greater numerical accuracy.

Figure 2.2: For instances of two examples, the speed and logarithmic homogeneity condition violation (at the final iterate) for the two inverse Hessian product procedures.



Chapter 3

Conic optimization with spectral functions on Euclidean Jordan algebras

Abstract

Spectral functions on Euclidean Jordan algebras arise frequently in convex models. Despite the success of primal-dual conic interior point solvers, there has been little work on enabling direct support for *spectral function cones*, i.e. proper nonsymmetric cones defined from epigraphs and perspectives of spectral functions. We propose simple logarithmically homogeneous barriers for spectral function cones and we derive efficient, numerically stable procedures for evaluating barrier oracles such as inverse Hessian operators. For two useful classes of spectral function cones - the *root-determinant cones* and the *matrix monotone derivative cones* - we show that the barriers are self-concordant, with nearly optimal parameters. We implement these cones and oracles in Hypatia, and we write simple, natural conic formulations for four applied examples. In our computational experiments, Hypatia solves these natural formulations more efficiently than specialized conic solvers such as MOSEK 9 solve equivalent standard conic extended formulations.

3.1 Introduction

In convex optimization applications, we frequently encounter spectral functions on Euclidean Jordan algebras such as the real vectors and real symmetric or complex Hermitian matrices. In this context, a spectral function is a real-valued symmetric function of the (real) eigenvalues. Examples include the geometric mean (or root-determinant), the entropy (e.g. von Neumann entropy), and the trace of the inverse (e.g. the A-optimal design criterion). Indeed, many disciplined convex programming (DCP) functions are spectral functions (Grant, Boyd, and Ye, 2006; Grant and Boyd, 2014). The *spectral function cones* are proper cones defined from epigraphs or hypographs of homogeneous spectral functions or perspective functions of nonhomogeneous spectral functions. These cones allow simple, natural conic reformulations of a wide range of convex optimization problems. However, there has been little prior work enabling direct support for various spectral function cones in primal-dual conic solvers.

As we discuss in Chapter 1, conic interior point methods (IPMs) require tractable oracles for logarithmically homogeneous self-concordant barrier (LHSCB) functions for proper cones. Complexity analysis of idealized IPMs shows that they converge to ε tolerance in $\mathcal{O}(\sqrt{\nu}\log(1/\varepsilon))$ iterations, where ν is the barrier parameter of the LHSCB. Currently, most IPMs are specialized for the nonnegative, second order, and positive semidefinite (PSD) cones, which are cones of squares of Euclidean Jordan algebras. However these symmetric cones limit modeling generality and often require the construction of large extended formulations (EFs). For many spectral function cones (e.g. the epigraph of the perspective of negative entropy), equivalent EFs using only symmetric cones do not exist, and when they do, they can be impractically large. Nonsymmetric conic IPMs (e.g. by Nesterov, Todd, and Ye (1996), Nesterov (2012), and Skajaa and Ye (2015)) can handle a much broader class of cones.

In Chapter 1, we generalize and enhance the practical performance of the IPM by Skajaa and Ye (2015) and test our implementations in Hypatia. Recall that Hypatia's generic cone interface allows specifying a proper cone $\mathcal{K}$ by implementing a small list of oracles, and once $\mathcal{K}$ is defined, both $\mathcal{K}$ and its dual cone $\mathcal{K}^*$ may be used in any combination with other recognized cones to construct conic models. The oracles Hypatia uses for ideal performance include an initial interior point $t \in \text{int}(\mathcal{K})$, a feasibility test for the cone interior $\text{int}(\mathcal{K})$ and for the dual cone interior $\text{int}(\mathcal{K}^*)$, and several LHSCB oracles - in particular, the gradient, (inverse) Hessian product, and scaled third order directional derivative. Fast and numerically stable procedures for evaluating cone oracles are crucial for practical performance in conic IPM solvers such as Hypatia.

Our first main contribution is to define simple logarithmically homogeneous barriers for spectral function cones and derive efficient and numerically stable barrier oracle procedures. For example, for the case where the spectral function is separable, we show how to apply the inverse Hessian operator of the barrier function very cheaply using a closed form formula, without the need to compute or factorize an explicit Hessian matrix (which can be expensive and prone to numerical issues). Similarly, for the negative log-determinant and root-determinant spectral function cones, we derive highly-efficient specialized oracle procedures.

Our second main contribution is to show that for two important subclasses of spectral function cones - the root-determinant cones and the matrix monotone derivative (MMD) cones - the barriers we propose are LHSCBs. These LHSCBs have parameters that are only a small additive increment of one larger than the parameter of the LHSCB for the cone of squares domain of the cone, hence the parameters are near-optimal. MMD cones allow modeling epigraphs of a variety of useful convex separable spectral functions, e.g. the trace of the negative logarithm, negative entropy, and negative squareroot. Furthermore, the dual cones of the MMD cones allow modeling epigraphs of even more separable spectral functions, such as the trace of the inverse and exponential functions.

These developments enable efficient and numerically stable implementations of the MMD cone and the log-determinant and root-determinant cones in nonsymmetric conic IPMs. We define these cones through Hypatia's cone interface. Our MMD cone implementation is parametrized by both a Jordan algebra domain and an MMD function, allowing the user to define new domains and

MMD functions. An MMD function is easily specified by implementing a small set of oracles for its univariate form: the function itself, its first three derivatives, and its convex conjugate, as well as an interior point for the corresponding MMD cone. We predefine five common MMD functions and three typical Jordan algebra domains: the real vectors, real symmetric matrices, and complex Hermitian matrices. In Section 2.2.3, we describe techniques for constructing EFs using standard cones (those recognized by MOSEK 9; see Section 2.2) for constraints over these spectral function cones.

We formulate example problems from distribution estimation, experiment design, quantum information science, and polynomial optimization. The natural formulations (NFs) using these cones are simpler and smaller than the equivalent EFs. Our computational experiments demonstrate that, across a wide range of sizes and spectral functions, Hypatia can solve the NFs faster than Hypatia, MOSEK, or ECOS can solve the equivalent EFs. Furthermore, to illustrate the practical impact of our efficient oracle procedures, we show that our analytic inverse Hessian product for the MMD cone is faster and more numerically reliable than a naive direct solve using an explicit Hessian factorization.

3.1.1 Overview

We describe relevant aspects of Euclidean Jordan algebras, cones of squares, and spectral decompositions in Section 3.2. In Section 3.3, we define spectral functions on Euclidean Jordan algebras and give expressions for gradients and second and third order directional derivatives of spectral functions. We also specialize these formulae for separable spectral functions and the log-determinant case.

In Section 3.4, we define spectral function cones (and their dual cones) from epigraphs of homogenized convex spectral functions on cones of squares. We propose simple logarithmically homogeneous barriers for these cones and describe the additional properties that must be satisfied by an LHSCB. We also define several barrier oracles needed by Hypatia's IPM. Then in Section 3.5, we describe fast and numerically stable procedures for these barrier oracles, using the derivative results from Section 3.3. We specialize the oracle procedures for cones defined from separable spectral functions and the log-determinant function.

In Section 3.6, we define the MMD cone and its dual cone, and we give useful examples of MMD functions. We show that for the MMD cone, our barrier function is an LHSCB. In Section 3.7, we define the root-determinant cone and its dual cone, prove that our barrier is an LHSCB, and derive efficient oracle procedures. Finally, in Section 3.8, we describe a series of applied examples over the new root-determinant, log-determinant, and MMD cones and their dual cones. We perform computational testing to demonstrate the advantages of solving these NFs with Hypatia and to exemplify the impact of efficient oracle procedures.

3.2 Jordan algebras

Jordan algebraic concepts provide a useful and straightforward abstraction for spectral functions, cones, and our barrier results in later sections. We follow the notation of Faraut and Koranyi (1998, Chapter 2) where possible.

An algebra over the real or complex numbers is a vector space V equipped with a bilinear product $\circ : V \times V \to V$. For $w \in V$, $w^2 \coloneqq w \circ w$. We refer to V as a *Jordan algebra* if for all $w_a, w_b \in V$:

$$w_a \circ w_b = w_b \circ w_a, \tag{3.1a}$$

$$w_a \circ (w_a^2 \circ w_b) = w_a^2 \circ (w_a \circ w_b). \tag{3.1b}$$

For example, for $V = \mathbb{R}^d$, we can define $\circ$ as an elementwise multiplication, or for $V = \mathbb{S}^d$ and $V = \mathbb{H}^d$, we can let $w_a \circ w_b = \frac{1}{2}(w_a w_b + w_b w_a)$.

Given $w_a \in V$, we define the linear map $L(w_a) : V \to V$ satisfying:

$$L(w_a)w_b = w_a \circ w_b \qquad \forall w_b \in V. \tag{3.2}$$

Given $w \in V$, we define the linear map $P(w) : V \to V$ satisfying:

$$P(w) = 2L(w)^2 - L(w^2). \tag{3.3}$$

P is called the *quadratic representation* of V. In general, $P(w) \neq L(w)^2 \neq L(w^2)$ because $\circ$ need not be associative. For example, for $V = \mathbb{S}^d$, we have $L(w_a^2)w_b = \frac{1}{2}(w_a^2 w_b + w_b w_a^2)$, $L(w_a)^2 w_b = \frac{1}{2}(L(w_a^2)w_b + w_a w_b w_a)$, and $P(w_a)w_b = w_a w_b w_a$.

For any positive integer k, we have (Manuel V.C. Vieira, 2007, Corollary 2.3.9):

$$P(w)^k = P(w^k). \tag{3.4}$$

It is standard to assume the existence of a multiplicative identity e. Note that $P(e)w = w$. A point $w \in V$ is *invertible* if and only if $L(w)$ is invertible, and the inverse of w is the element $w^{-1} \in V$ such that $w^{-1} = L(w)^{-1}e$ (Faraut and Koranyi, 1998, Proposition II.2.2). (3.4) also holds for $k = -1$ if w is invertible (Faraut and Koranyi, 1998, Proposition II.3.1).

Henceforth we consider only the finite dimensional *Euclidean* Jordan algebras. A Jordan algebra V is Euclidean if $\langle w_a \circ w_b, w_c \rangle = \langle w_b, w_a \circ w_c \rangle$ for all $w_a, w_b, w_c \in V$.

We call $\mathcal{Q}$ a *cone of squares* on V if $\mathcal{Q} = \{w \circ w : w \in V\}$. The cone $\mathcal{Q}$ is proper (closed, convex, pointed, and solid) because V is Euclidean (and therefore formally real); see Papp and Alizadeh (2013, Theorem 3.3) and Faraut and Koranyi (1998, Section III.1 and Proposition VIII.4.2). In addition, $\mathcal{Q}$ is *self-dual* and *homogeneous*; see Manuel V.C. Vieira (2007, Proposition 2.5.8) and Faraut and Koranyi (1998, Theorem III.2.1). For example, for $V = \mathbb{S}^d$, the cone of squares is $\mathcal{Q} = \mathbb{S}^d_+$.

For convenience, we often write $a \succeq b$ instead of $a - b \in \mathcal{Q}$, or $a \succ b$ instead of $a - b \in \operatorname{int}(\mathcal{Q})$, where $\mathcal{Q}$ is clear from context. If $w \succ 0$, then w is invertible (Faraut and Koranyi, 1998, Theorem III.2.1). Furthermore $w \succ 0$ implies that $w^{1/2}$ is well-defined and invertible, and $P(w^{1/2}) = P(w)^{1/2}$

Manuel V.C. Vieira (2007, Proposition 2.5.11). This also implies by (3.4) (with $k = -1$) that $P(w^{-1/2}) = P(w)^{-1/2}$.

3.2.1 Spectral decomposition

In a Euclidean Jordan algebra V, an *idempotent* is an element $c \in V$ such that $c^2 = c$. Two idempotents c_1, c_2 are *orthogonal* if $c_1 \circ c_2 = 0$. Let d be the rank of V. $c_1, \ldots, c_d$ is a *complete* system of orthogonal idempotents if $c_1, \ldots, c_d$ are all idempotents, pairwise orthogonal, and $\sum_{i \in [\![d]\!]} c_i = e$. An idempotent is *primitive* if it is non-zero and cannot be written as the sum of two orthogonal non-zero idempotents. A *Jordan frame* is a complete system of orthogonal idempotents, where each idempotent is primitive. The number of elements in any Jordan frame is called the *rank* of V. For example, the rank of $\mathbb{R}^d$, $\mathbb{S}^d$, or $\mathbb{H}^d$ is d.

For any $w \in V$, there exist unique real numbers (not necessarily distinct) $w_1, \ldots, w_d$ and a unique Jordan frame $c_1, \ldots, c_d$ such that w has the *spectral decomposition* (Faraut and Koranyi, 1998, Theorem III.1.2):

$$w = \sum_{i \in [\![d]\!]} w_i c_i. \tag{3.5}$$

We call $w_1, \ldots, w_d$ the *eigenvalues* of w. The *determinant* is $\det(w) = \prod_{i \in [\![d]\!]} w_i$ and the *trace* is $\operatorname{tr}(w) = \sum_{i \in [\![d]\!]} w_i$ (Faraut and Koranyi, 1998, Section II.2, Page 29). For example, for $V = \mathbb{R}^d$, the Jordan frame is the standard unit vectors and w is its own vector of eigenvalues. For $V = \mathbb{S}^d$, we can think of the Jordan frame as the rank one PSD matrices from a full symmetric eigendecomposition.

Henceforth, we define the inner product on V as $\langle w_a, w_b \rangle = \operatorname{tr}(w_a \circ w_b)$. Under this inner product, $P(w)$ is self-adjoint (Manuel V.C. Vieira, 2007, Page 27). Thus for $w \in \operatorname{int}(\mathcal{Q})$ and $r_1, r_2 \in V$, we have:

$$\langle P(w) r_1, r_2 \rangle = \langle P(w^{1/2}) r_1, P(w^{1/2}) r_2 \rangle = \langle r_1, P(w) r_2 \rangle. \tag{3.6}$$

3.2.2 Peirce decomposition

We let $c_1, \ldots, c_d$ be a Jordan frame for V, and define for $i, j \in [\![d]\!]$:

$$V(c_i, \lambda) \coloneqq \{w : c_i \circ w = \lambda w\}, \tag{3.7a}$$

$$V_{i,i} \coloneqq V(c_i, 1) = \{t c_i : t \in \mathbb{R}\}, \tag{3.7b}$$

$$V_{i,j} \coloneqq V(c_i, \tfrac{1}{2}) \cap V(c_j, \tfrac{1}{2}). \tag{3.7c}$$

V has the direct sum decomposition $V = \oplus_{i,j \in [\![d]\!] : i \le j} V_{i,j}$ (Faraut and Koranyi, 1998, Theorem IV.1.3). For example, for $V = \mathbb{S}^d$, let $E_{i,j}$ be a matrix of zeros except in the (i,j)th position, and let $c_i = E_{i,i}$; then $V_{i,i} = \{t E_{i,i} : t \in \mathbb{R}\}$ and $V_{i,j} = \{t(E_{i,j} + E_{j,i}) : t \in \mathbb{R}\}$.

The *Peirce decomposition* allows us to write any $r \in V$ as:

$$r = \sum_{i,j \in [\![d]\!] : i \le j} r_{i,j} = \sum_{i,j \in [\![d]\!] : i < j} r_{i,j} + \sum_{i \in [\![d]\!]} r_i c_i, \tag{3.8}$$

where $r_i = \langle r, c_i \rangle$ and $r_{i,j} \in V_{i,j}$, $\forall i, j \in [\![d]\!]$. Each $r_{i,j}$ is a projection of r onto $V_{i,j}$, where:

$$r_{i,i} = r_i c_i = P(c_i) r \qquad \forall i \in [\![d]\!], \tag{3.9a}$$

$$r_{i,j} = 4L(c_i)L(c_j)r = 4c_i \circ (c_j \circ r) \qquad \forall i, j \in [\![d]\!] : j \neq i. \tag{3.9b}$$

Note that $r_{i,j} = r_{j,i}$, since $L(c_i)$ and $L(c_j)$ commute (Faraut and Koranyi, 1998, Lemma IV.1.3). For example, let $c_1, \ldots, c_d$ be a Jordan frame for $V = \mathbb{S}^d$ and let $r \in V$; then $r_i = c_i r c_i$ and $r_{i,j} = c_i r c_j + c_j r c_i$, for $i, j \in [\![d]\!]$.

We list some useful facts relating to compositions of projection operators (see Faraut and Koranyi (1998, Theorem IV.2.2) and D. Sun and J. Sun (2008, Page 430)):

$$L(c_i)L(c_j)L(c_k)L(c_l) = 0 \qquad \forall i, j, k, l \in [\![d]\!] : i \neq j, k \neq l, (i, j) \neq (k, l), \tag{3.10a}$$

$$L(c_i)L(c_j)P(c_k) = 0 \qquad \forall i, j, k \in [\![d]\!] : i \neq j, \tag{3.10b}$$

$$P(c_k)L(c_i)L(c_j) = 0 \qquad \forall i, j, k \in [\![d]\!] : i \neq j, \tag{3.10c}$$

$$(4L(c_i)L(c_j))^2 = 4L(c_i)L(c_j) \qquad \forall i, j \in [\![d]\!], \tag{3.10d}$$

$$P(c_i)^2 = P(c_i) \qquad \forall i \in [\![d]\!], \tag{3.10e}$$

and furthermore:

$$L(e) = \sum_{i,j \in [\![d]\!] : i < j} 4L(c_i)L(c_j) + \sum_{i \in [\![d]\!]} P(c_i). \tag{3.11}$$

Given $\lambda_{i,j} \neq 0$ for $i, j \in [\![d]\!]$, consider an operator $\Lambda : V \to V$ of the form:

$$\Lambda \coloneqq \sum_{i,j \in [\![d]\!] : i < j} 4\lambda_{i,j} L(c_i)L(c_j) + \sum_{i \in [\![d]\!]} \lambda_{i,i} P(c_i). \tag{3.12}$$

The inverse operator is given by:

$$\Lambda^{-1} = \sum_{i,j \in [\![d]\!] : i < j} 4\lambda_{i,j}^{-1} L(c_i)L(c_j) + \sum_{i \in [\![d]\!]} \lambda_{i,i}^{-1} P(c_i). \tag{3.13}$$

It can be verified using (3.10) and (3.11) that for any $r \in V$, $\Lambda\Lambda^{-1}r = \Lambda^{-1}\Lambda r = r$. For example, let $w = \sum_{i \in [\![d]\!]} w_i c_i \in V$ be invertible and suppose that $\lambda_{i,j} = w_i w_j$ for $i, j \in [\![d]\!]$; then:

$$\Lambda = \sum_{i,j \in [\![d]\!] : i < j} 4w_i w_j L(c_i)L(c_j) + \sum_{i \in [\![d]\!]} w_i^2 P(c_i) = P(w), \tag{3.14a}$$

$$\Lambda^{-1} = \sum_{i,j \in [\![d]\!] : i < j} 4w_i^{-1} w_j^{-1} L(c_i)L(c_j) + \sum_{i \in [\![d]\!]} w_i^{-2} P(c_i) = P(w^{-1}). \tag{3.14b}$$

3.3 Spectral functions and derivatives

Let V be a Jordan algebra of rank d. A real-valued function $f : \mathbb{R}^d \to \mathbb{R}$ is *symmetric* if it is invariant to the order of its inputs. A symmetric function f composed with an eigenvalue map $\lambda : V \to \mathbb{R}^d$ induces a *spectral function* $\varphi : V \to \mathbb{R}$ such that $\varphi(w) = f(\lambda(w))$, where $\lambda(w) = (w_1, \ldots, w_d)$ is the

eigenvalue vector of w (Baes, 2007, Definition 8). Note that φ is convex if and only if f is convex (Davis, 1957).

In this section, we give expressions for certain derivatives and directional derivatives of φ that are useful for the barrier oracles we derive in Section 3.5. We express these derivatives at a point $w \in V$ (satisfying certain assumptions as necessary below) with spectral decomposition (3.5), and we let the direction be $r \in V$ with Peirce decomposition (3.8). The gradient is $\nabla\varphi(w) \in V$ and the second and third order directional derivatives are $\nabla^2\varphi(w)[r] \in V$ and $\nabla^3\varphi(w)[r, r] \in V$. We begin with the general nonseparable case in Section 3.3.1 before specializing for separable spectral functions in Section 3.3.2 and finally for the important case of the negative log-determinant function in Section 3.3.3.

3.3.1 The nonseparable case

Let ∇f, $\nabla^2 f$, and $\nabla^3 f$ denote the derivatives of f evaluated at $\lambda(w)$. We use subindices to denote particular components of these derivatives. According to Baes (2007, Theorem 38) and D. Sun and J. Sun (2008, Theorem 4.1), the gradient of φ at w is:

$$\nabla\varphi(w) = \sum_{i \in [\![d]\!]} (\nabla f)_i c_i. \tag{3.15}$$

Henceforth we assume the eigenvalues of w are all distinct for simplicity. The second order directional derivative of φ in direction r is (D. Sun and J. Sun, 2008, Theorem 4.2):

$$\nabla^2\varphi(w)[r] = \sum_{i,j \in [\![d]\!] : i<j} \frac{(\nabla f)_i - (\nabla f)_j}{w_i - w_j} r_{i,j} + \sum_{i,j \in [\![d]\!]} (\nabla^2 f)_{i,j} r_i c_j. \tag{3.16}$$

D. Sun and J. Sun (2008, Theorem 4.2) also generalize this expression to allow for non-distinct eigenvalues.

To derive an expression for the third order directional derivative $\nabla^3\varphi(w)[r, r]$, we let:

$$w(t) \coloneqq w + tr = \sum_{i \in [\![d]\!]} w_i(t) c_i(t), \tag{3.17}$$

where $w_i(t)$ is the ith eigenvalue of $w(t)$. Note that $\nabla^2\varphi(w)[r] = \frac{\mathrm{d}}{\mathrm{d}t}\nabla\varphi(w(t))|_{t=0}$ and $\nabla^3\varphi(w)[r, r] = \frac{\mathrm{d}^2}{\mathrm{d}t^2}\nabla\varphi(w(t))|_{t=0}$. We let $\nabla f(t)$, $\nabla^2 f(t)$, and $\nabla^3 f(t)$ denote the derivatives of f evaluated at $\lambda(w(t))$. Due to the chain rule and (3.16):

$$\frac{\mathrm{d}}{\mathrm{d}t}\nabla\varphi(w(t)) = \nabla^2\varphi(w(t))[r] \tag{3.18a}$$

$$= \sum_{i,j \in [\![d]\!] : i<j} \frac{(\nabla f(t))_i - (\nabla f(t))_j}{w_i(t) - w_j(t)} r_{i,j}(t) + \sum_{i,j \in [\![d]\!]} (\nabla^2 f(t))_{i,j} r_i(t) c_j(t). \tag{3.18b}$$

We differentiate (3.18) once more. From Manuel VC Vieira (2016, Corollary 1 and Theorem 3.3)

and D. Sun and J. Sun (2008, Equation 37), for $i \in [\![d]\!]$ we have:

$$\frac{\mathrm{d}}{\mathrm{d}t} w_i(t) = r_i(t), \tag{3.19a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} c_i(t) = s_i(t) \coloneqq \sum_{j \in [\![d]\!] : j \neq i} \frac{r_{i,j}(t)}{w_i(t) - w_j(t)}. \tag{3.19b}$$

Using the chain rule, (3.19a) implies:

$$\frac{\mathrm{d}}{\mathrm{d}t} (\nabla f(t))_i = \sum_{k \in [\![d]\!]} (\nabla^2 f(t))_{i,k} r_k(t) \qquad \forall i \in [\![d]\!], \tag{3.20a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} (\nabla^2 f(t))_{i,j} = \sum_{k \in [\![d]\!]} (\nabla^3 f(t))_{i,j,k} r_k(t) \qquad \forall i, j \in [\![d]\!]. \tag{3.20b}$$

Applying the chain and product rules, we have:

$$\frac{\mathrm{d}}{\mathrm{d}t} \frac{1}{w_i(t) - w_j(t)} = \frac{r_j(t) - r_i(t)}{(w_i(t) - w_j(t))^2} \qquad \forall i, j \in [\![d]\!] : i \neq j, \tag{3.21a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} r_{i,j}(t) = \frac{\mathrm{d}}{\mathrm{d}t} (4 c_i(t) \circ (c_j(t) \circ r)) \tag{3.21b}$$

$$= 4 c_i(t) \circ (s_j(t) \circ r) + 4 s_i(t) \circ (c_j(t) \circ r) \qquad \forall i, j \in [\![d]\!] : i \neq j, \tag{3.21c}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \langle c_i(t), r \rangle c_j(t) = \langle s_i(t), r \rangle c_j(t) + r_i(t) s_j(t) \qquad \forall i, j \in [\![d]\!]. \tag{3.21d}$$

Finally, letting $s_i \coloneqq s_i(0)$ for all $i \in [\![d]\!]$, these results imply that:

$$\nabla^3 \varphi[r, r] = \frac{\mathrm{d}^2}{\mathrm{d}t^2} \nabla \varphi(w(t))\big|_{t=0} \tag{3.22a}$$

$$= \sum_{i,j \in [\![d]\!] : i < j} \frac{(\nabla f)_i - (\nabla f)_j}{w_i - w_j} \Big(4 c_i \circ (s_j \circ r) + 4 s_i \circ (c_j \circ r) - \frac{r_i - r_j}{w_i - w_j} r_{i,j} \Big) + \sum_{i,j,k \in [\![d]\!] : i < j} \frac{(\nabla^2 f)_{i,k} - (\nabla^2 f)_{j,k}}{w_i - w_j} r_k r_{i,j} + \sum_{i,j \in [\![d]\!]} (\nabla^2 f)_{i,j} (\langle s_i, r \rangle c_j + r_i s_j) + \sum_{i,j,k \in [\![d]\!]} (\nabla^3 f)_{i,j,k} r_i r_k c_j \tag{3.22b}$$

$$= \sum_{i,j \in [\![d]\!] : i < j} \frac{(\nabla f)_i - (\nabla f)_j}{w_i - w_j} \Big(4 c_i \circ (s_j \circ r) + 4 s_i \circ (c_j \circ r) - \frac{r_i - r_j}{w_i - w_j} r_{i,j} \Big) + \sum_{i,j \in [\![d]\!]} (\nabla^2 f)_{i,j} (2 r_j s_i + \langle s_i, r \rangle c_j) + \sum_{i,j,k \in [\![d]\!]} (\nabla^3 f)_{i,j,k} r_i r_k c_j. \tag{3.22c}$$

The derivative expressions simplify significantly for $V = \mathbb{R}^d$. For $V = \mathbb{S}^d$, the form of (3.16) is well-known (Faybusovich and Zhou, 2021) and the form of (3.22c) appears in Sendov (2007).

3.3.2 The separable case

The spectral function φ induced by f is *separable* if f is a separable function, i.e. $f(\lambda) = \sum_{i\in[\![d]\!]} h(\lambda_i)$ for $\lambda \in \mathbb{R}^d$ and some function $h : \mathbb{R} \to \mathbb{R}$. For convenience, if $w \in V$, we also define $h : V \to V$ as $h(w) \coloneqq \sum_{i\in[\![d]\!]} h(w_i)c_i$. This allows us to write $\varphi(w) = \operatorname{tr}(h(w)) = \sum_{i\in[\![d]\!]} h(\lambda_i)$. Note that φ is convex if and only if h is convex. For example, if $h(w) = -\log(w)$ then $\varphi(w) = \operatorname{tr}(-\log(w)) = -\operatorname{logdet}(w)$; we consider this special case in Section 3.3.3.

We specialize the derivatives from (3.15), (3.16) and (3.22c), maintaining the assumption of distinct eigenvalues. Since $(\nabla^2 f)_{i,j} = (\nabla^3 f)_{i,j,k} = 0$ unless $i = j = k$, we have:

$$\nabla\varphi(w) = \sum_{i\in[\![d]\!]} \nabla h(w_i)c_i, \tag{3.23a}$$

$$\nabla^2\varphi(w)[r] = \sum_{i,j\in[\![d]\!]:i<j} \frac{\nabla h(w_i) - \nabla h(w_j)}{w_i - w_j} 4c_i \circ (c_j \circ r) + \sum_{i\in[\![d]\!]} \nabla^2 h(w_i)P(c_i)r, \tag{3.23b}$$

$$\begin{aligned}\nabla^3\varphi(w)[r,r] = \sum_{i,j\in[\![d]\!]:i<j} \frac{\nabla h(w_i) - \nabla h(w_j)}{w_i - w_j}\Big(&4c_i \circ (s_j \circ r) + 4s_i \circ (c_j \circ r) - \\ &\frac{r_i - r_j}{w_i - w_j} r_{i,j}\Big) + \sum_{i\in[\![d]\!]} \nabla^2 h(w_i)(2r_i s_i + \langle s_i, r\rangle c_i) + \sum_{i\in[\![d]\!]} \nabla^3 h(w_i) r_i^2 c_i.\end{aligned} \tag{3.23c}$$

3.3.3 The negative log-determinant case

The negative log-determinant function $\varphi(w) = -\operatorname{logdet}(w)$ is a separable spectral function. We let $w \succ 0$ and drop the assumption of distinct eigenvalues. For convenience, we let $\hat{r} \coloneqq P(w^{-1/2})r \in V$. First, note that (similar to Manuel V.C. Vieira (2007, Lemma 3.3.4)):

$$\langle w^{-1}, r\rangle = \langle P(w^{-1/2})e, r\rangle = \langle e, P(w^{-1/2})r\rangle = \operatorname{tr}(\hat{r}), \tag{3.24}$$

by (3.6). Due to Faraut and Koranyi (1998, Proposition II.2.3):

$$\nabla_w(\operatorname{tr}(\hat{r})) = \nabla_w(w^{-1})[r] = -P(w^{-1})r. \tag{3.25}$$

Adapting the result in Faybusovich and Tsuchiya (2017, Lemma 3.4):

$$\nabla_w(P(w^{-1})r)[r] = -2P(w^{-1/2})\hat{r}^2. \tag{3.26}$$

Now, the gradient of φ is (Faraut and Koranyi, 1998, Propositions III.4.2(ii)):

$$\nabla\varphi(w) = -w^{-1}, \tag{3.27}$$

so from (3.25) and (3.26), the second and third order directional derivatives are:

$$\nabla^2\varphi(w)[r] = P(w^{-1})r, \tag{3.28a}$$

$$\nabla^3\varphi(w)[r,r] = -2P(w^{-1/2})(P(w^{-1/2})r)^2. \tag{3.28b}$$

Note that unlike the separable spectral function case in Section 3.3.2, here we do not need the explicit eigenvalues of w.

3.4 Cones and barrier functions

In this chapter we are concerned with a class of proper cones that can be characterized as follows:

$$\mathcal{K} \coloneqq \operatorname{cl}\{\tilde{u} \in \mathcal{E} : \zeta(\tilde{u}) \geq 0\} \subset \tilde{V}, \tag{3.29}$$

where $\zeta : \mathcal{E} \to \mathbb{R}$ is a concave, (degree one) homogeneous function and $\mathcal{E}$ is some convex cone in the space $\tilde{V}$. In particular, we define ζ in terms of a C^3-smooth spectral function φ that is defined on the interior of a cone of squares $\mathcal{Q}$ of a Jordan algebra V of rank d.

3.4.1 The homogeneous case

First, we suppose that φ is convex and homogeneous. Then $\zeta(u, w) \coloneqq u - \varphi(w)$ is concave and homogeneous, and we let $\mathcal{E} \coloneqq \mathbb{R} \times \operatorname{int}(\mathcal{Q})$ and $\tilde{V} \coloneqq \mathbb{R} \times V$. This defines a convex cone that is the closure of the epigraph set of φ:

$$\mathcal{K}_h \coloneqq \operatorname{cl}\{(u, w) \in \mathbb{R} \times \operatorname{int}(\mathcal{Q}) : u \geq \varphi(w)\}. \tag{3.30}$$

Note that if φ is concave, we can analogously define a cone from the hypograph set of φ. In Section 3.7, we consider the root-determinant cone, which is the hypograph of the concave root-determinant function. To check membership in $\operatorname{int}(\mathcal{K}_h)$, we first determine whether $w \in \operatorname{int}(\mathcal{Q})$ (which is equivalent to positivity of the eigenvalues), and if so, whether $\zeta(\tilde{u}) > 0$.

3.4.2 The non-homogeneous case

Now we suppose that φ is convex and non-homogeneous. We define the perspective function of φ, $\operatorname{per}\varphi : \mathbb{R}_> \times \operatorname{int}(\mathcal{Q}) \to \mathbb{R}$, as $(\operatorname{per}\varphi)(v, w) \coloneqq v\varphi(v^{-1}w)$. This is a homogeneous and convex function (Boyd and Vandenberghe, 2004, Section 3.2.6). We let $\zeta(u, v, w) \coloneqq u - (\operatorname{per}\varphi)(v, w)$, with $\mathcal{E} \coloneqq \mathbb{R} \times \mathbb{R}_> \times \operatorname{int}(\mathcal{Q})$ and $\tilde{V} \coloneqq \mathbb{R} \times \mathbb{R} \times V$. This defines a convex cone that is the closure of the epigraph set of the perspective function of φ:

$$\mathcal{K}_p \coloneqq \operatorname{cl}\{(u, v, w) \in \mathbb{R} \times \mathbb{R}_> \times \operatorname{int}(\mathcal{Q}) : u \geq v\varphi(v^{-1}w)\}. \tag{3.31}$$

Equivalently, we can view $\mathcal{K}_p$ as the closed conic hull of the epigraph set of φ (Nesterov and Nemirovski, 1994, Chapter 5). In Section 3.6, we consider the special case where φ is a separable spectral function with matrix monotone first derivative. The membership check for $\operatorname{int}(\mathcal{K}_p)$ is similar to that of $\operatorname{int}(\mathcal{K}_h)$ except we first check whether $v > 0$.

3.4.3 Dual cones

A proper cone is primitive if it cannot be written as a Cartesian product of two or more lower dimensional proper cones. The dual cone of a primitive, proper cone $\mathcal{K}$ is another primitive, proper cone:

$$\mathcal{K}^* := \{z : \langle s, z\rangle \geq 0, \forall s \in \mathcal{K}\}. \tag{3.32}$$

Recall that when $\mathcal{K}$ is defined through Hypatia's generic cone interface, both $\mathcal{K}$ and $\mathcal{K}^*$ become available for constructing conic models.

We assume that φ is convex, and we derive the dual cones of the epigraph cones $\mathcal{K}_h$ and $\mathcal{K}_p$ (these steps can be adapted for analogous hypograph cones if φ is concave). We define the convex conjugate function $\varphi^* : V \to \mathbb{R} \cup \infty$ of φ as the modified Legendre-Fenchel transformation (similar to Zhang (2004, Page 483)):

$$\varphi^*(r) = \sup_{w \in \mathrm{dom}(\varphi)}\{-\langle w, r\rangle - \varphi(w)\}, \tag{3.33}$$

which is a convex function. The conjugate of a symmetric function is also a symmetric function (Baes, 2007, Lemma 29), and the conjugate of a spectral function induced by a symmetric function f is the spectral function induced by f^* (Baes, 2007, Theorem 30). Thus for $\varphi(w) = f(\lambda(w))$ we have the conjugate function $\varphi^*(w) = f^*(\lambda(w))$.

For the epigraph-perspective cone $\mathcal{K}_p$ in (3.31), Zhang (2004, Theorem 3.2) and Rockafellar (2015, Theorem 14.4) derive the dual cone $\mathcal{K}_p^*$:

$$\mathcal{K}_p^* = \mathrm{cl}\{(u, v, w) \in \mathbb{R}_> \times \mathbb{R} \times V : v \geq u\varphi^*(u^{-1}w)\}. \tag{3.34}$$

We can view $\mathcal{K}_p^*$ as the epigraph set of the perspective function of the conjugate of φ, but with the epigraph and perspective components swapped (compare to (3.31)). Depending on the natural domain of φ^*, the w component of $\mathcal{K}_p^*$ is not necessarily restricted to lie in $\mathcal{Q}$; in Section 3.6.2 we discuss several example spectral functions, some of which have conjugates defined on all V and others only on $\mathrm{int}(\mathcal{Q})$.

For $\mathcal{K}_h$ in (3.30), we derive the dual cone $\mathcal{K}_h^*$ as follows. Since φ is homogeneous in this case, $(\mathrm{per}\,\varphi)(v, w) = v\varphi(v^{-1}w) = \varphi(w)$. Therefore the corresponding perspective cone $\mathcal{K}_p$ for φ is not a primitive cone, as it can be written as a (permuted) Cartesian product of $\mathbb{R}_\geq$ and $\mathcal{K}_h$:

$$\mathcal{K}_p = \mathrm{cl}\{(u, v, w) \in \tilde{V} : v \in \mathbb{R}_\geq, (u, w) \in \mathcal{K}_h\}. \tag{3.35}$$

Since the dual cone of a Cartesian product of cones is the Cartesian product of their dual cones, we have (since $\mathbb{R}_\geq^* = \mathbb{R}_\geq$):

$$\mathcal{K}_p^* = \mathrm{cl}\{(u, v, w) \in \tilde{V} : v \in \mathbb{R}_\geq, (u, w) \in \mathcal{K}_h^*\}. \tag{3.36}$$

By (Lasserre, 1998, Theorem 2.1), the homogeneity of φ implies that φ^* can only take the values

zero or infinity. Hence by (3.34), we know:

$$\mathcal{K}_p^* = \mathrm{cl}\{(u, v, w) \in \mathbb{R}_> \times \mathbb{R} \times V : v \geq 0, u\varphi^*(u^{-1}w) < \infty\}. \tag{3.37}$$

Since (3.36) and (3.37) describe the same cone, we can conclude that the dual cone of $\mathcal{K}_h$ is:

$$\mathcal{K}_h^* = \mathrm{cl}\{(u, w) \in \mathbb{R}_> \times V : \varphi^*(u^{-1}w) < \infty\}. \tag{3.38}$$

3.4.4 Barrier functions and oracles

A logarithmically homogeneous barrier (LHB) function Γ for a proper cone $\mathcal{K} \subset \tilde{V}$ is C^2-smooth and satisfies $\Gamma(\tilde{u}_i) \to \infty$ along every sequence $\tilde{u}_i \in \mathrm{int}(\mathcal{K})$ converging to the boundary of $\mathcal{K}$, and:

$$\Gamma(\theta\tilde{u}) = \Gamma(\tilde{u}) - \nu \log(\theta) \qquad \forall \tilde{u} \in \mathrm{int}(\mathcal{K}), \theta \in \mathbb{R}_>, \tag{3.39}$$

for some $\nu \geq 0$ (Nesterov and Nemirovski, 1994, Definition 2.3.2). If Γ is also self-concordant, then it is an LHSCB with parameter $\nu \geq 1$ (or a ν-LHSCB) for $\mathcal{K}$. For self-concordance, Γ must be C^3-smooth and satisfy (Nesterov and Nemirovski, 1994, Definition 2.1.1):

$$|\nabla^3\Gamma(\tilde{u})[\tilde{p}, \tilde{p}, \tilde{p}]| \leq 2(\nabla^2\Gamma(\tilde{u})[\tilde{p}, \tilde{p}])^{3/2} \qquad \forall \tilde{u} \in \mathrm{int}(\mathcal{K}), \tilde{p} \in \tilde{V}. \tag{3.40}$$

The best known interior point algorithms need at most $\mathcal{O}(\sqrt{\nu}\log(1/\varepsilon))$ iterations to converge to a solution within ε tolerance (Nesterov, Todd, and Ye, 1997).

The LHB we consider for a cone of the form (3.29) is:

$$\Gamma(\tilde{u}) \coloneqq -\log(\zeta(\tilde{u})) + \Psi(\tilde{u}), \tag{3.41}$$

where Ψ can be thought of as an LHSCB for the domain of ζ or $\mathrm{cl}(\mathcal{E})$. The negative logarithm function $-\log$ is the standard LHSCB for $\mathbb{R}_\geq$, with parameter $\nu = 1$. Similarly, the spectral function $-\mathrm{logdet}$ (see Section 3.3.3) is the standard LHSCB for a cone of squares $\mathcal{Q}$ of V, with $\nu = d$ (the rank of V). For $\mathcal{K}_h$ we let $\Psi(\tilde{u}) = -\mathrm{logdet}(w)$, hence Γ has parameter $\nu = 1 + d$. Since an LHSCB for a Cartesian product of cones is the sum of LHSCBs for the primitive cones, for $\mathcal{K}_p$ we let $\Psi(\tilde{u}) = -\log(v) - \mathrm{logdet}(w)$, hence Γ has parameter $\nu = 2 + d$. Note that although Γ is an LHB, it is not necessarily self-concordant; in Sections 3.6.4 and 3.7.3 we prove that Γ is an LHSCB for some useful special cases.

We now define four barrier oracles that Hypatia's IPM uses; for ideal performance, these oracle implementations should be efficient and numerically stable. For an interior point $\tilde{u} \in \mathrm{int}(\mathcal{K})$ and a direction $\tilde{p} \in \tilde{V}$, the gradient g, the Hessian product H, the inverse Hessian product $\bar{H}$, and the third order directional derivative T are:

$$g \coloneqq \nabla\Gamma(\tilde{u}), \tag{3.42a}$$

$$H \coloneqq \nabla^2\Gamma(\tilde{u})[\tilde{p}], \tag{3.42b}$$

$$\bar{H} \coloneqq (\nabla^2\Gamma(\tilde{u}))^{-1}[\tilde{p}], \tag{3.42c}$$

$$T \coloneqq \nabla^3\Gamma(\tilde{u})[\tilde{p}, \tilde{p}]. \tag{3.42d}$$

Note $g, H, \bar{H}, T \in \tilde{V}$. In later sections, we use subscripts to refer to subcomponents of these oracles, for example the w component of the gradient oracle is $g_w \coloneqq \nabla_w\Gamma(\tilde{u}) \in V$. Ideally, H applies the positive definite linear operator $\nabla^2\Gamma(\tilde{u}) : \tilde{V} \to \tilde{V}$ without constructing an explicit Hessian, and similarly, $\bar{H}$ applies the (unique) inverse operator $(\nabla^2\Gamma(\tilde{u}))^{-1} : \tilde{V} \to \tilde{V}$ without constructing or factorizing an explicit Hessian.

We note that for the standard LHSCB Ψ for a cone of squares, efficient and numerically stable procedures for these four oracles are well-known. The same cannot be said for the LHB Γ currently. In Section 3.5, we derive these oracles for $\mathcal{K}_p$ (noting that they can be adapted easily for $\mathcal{K}_h$). In the special cases for which we show Γ is an LHSCB, the oracles can be computed particularly efficiently.

3.5 Barrier oracles for epigraph-perspective cones

We consider the epigraph-perspective cone $\mathcal{K}_p$ defined in (3.31). Recall that we let $\tilde{p} = (p, q, r) \in \mathbb{R} \times \mathbb{R} \times V$ and $\tilde{u} = (u, v, w) \in \operatorname{int}(\mathcal{K}_p)$, and we define ζ and $\Gamma : \operatorname{int}(\mathcal{K}_p) \to \mathbb{R}$ from (3.41) as:

$$\zeta(\tilde{u}) \coloneqq u - v\varphi(v^{-1}w), \tag{3.43a}$$
$$\Gamma(\tilde{u}) \coloneqq -\log(\zeta(\tilde{u})) - \log(v) - \operatorname{logdet}(w). \tag{3.43b}$$

In this section, we derive expressions and evaluation procedures for the g, H, T, and $\bar{H}$ oracles (defined in Section 3.4.4) corresponding to the LHB Γ for $\mathcal{K}_p$. We note that the oracles for $\mathcal{K}_h$ in (3.30) are simpler because no perspective operation is needed for a homogeneous φ; they can be obtained by fixing $v = 1$ and $q = 0$ and ignoring the v components in the oracle expressions in this section.

Without assuming any particular form for φ, we write g, H, and T in Section 3.5.1 and $\bar{H}$ in Section 3.5.2 in terms of the derivatives of φ. If φ is a spectral function, these derivatives can be computed using the expressions from Section 3.3. In the case that φ is a separable spectral function (see Section 3.3.2), we derive a more specialized procedure for $\bar{H}$ in Section 3.5.3, which is no more expensive than H. Finally, in Section 3.5.4, we specialize the four oracles for the negative log-determinant function (i.e. $\varphi(w) = -\operatorname{logdet}(w)$; see Section 3.3.3) and we discuss implementations.

3.5.1 Derivatives

First, we express the derivatives of ζ in terms of those of φ. We define the function $\mu : \mathbb{R}_> \times \operatorname{int}(\mathcal{Q}) \to \operatorname{int}(\mathcal{Q})$ and its first directional derivative $\xi \in V$ in the direction (q, r) as:

$$\mu(v, w) \coloneqq v^{-1}w, \tag{3.44a}$$
$$\xi \coloneqq \nabla\mu(v, w)[(q, r)] = \nabla_v\mu(v, w)q + \nabla_w\mu(v, w)[r] = v^{-1}(r - q\mu(v, w)). \tag{3.44b}$$

For convenience, we fix the constants $\mu \coloneqq \mu(v,w)$, $\varphi \coloneqq \varphi(\mu)$, and $\zeta \coloneqq \zeta(\tilde{u})$. Let $\nabla\varphi$, $\nabla^2\varphi$, and $\nabla^3\varphi$ be the derivatives of φ evaluated at μ, and let $\nabla\zeta$, $\nabla^2\zeta$, and $\nabla^3\zeta$ be the derivatives of ζ evaluated at $\tilde{u}$. Using (3.44), the directional derivatives of ζ can be written compactly as:

$$\nabla_u \zeta = 1, \tag{3.45a}$$
$$\nabla_v \zeta = -\varphi + \nabla\varphi[\mu], \tag{3.45b}$$
$$\nabla_w \zeta = -\nabla\varphi, \tag{3.45c}$$
$$\nabla\zeta[\tilde{p}] = p - q\varphi - v\nabla\varphi[\xi], \tag{3.45d}$$
$$(\nabla^2\zeta[\tilde{p}])_v = \nabla^2\varphi[\xi,\mu], \tag{3.45e}$$
$$(\nabla^2\zeta[\tilde{p}])_w = -\nabla^2\varphi[\xi], \tag{3.45f}$$
$$\nabla^2\zeta[\tilde{p},\tilde{p}] = -v\nabla^2\varphi[\xi,\xi], \tag{3.45g}$$
$$(\nabla^3\zeta[\tilde{p},\tilde{p}])_v = \nabla^3\varphi[\xi,\xi,\mu] + \nabla^2\varphi[\xi,\xi] - 2v^{-1}q\nabla^2\varphi[\xi,\mu], \tag{3.45h}$$
$$(\nabla^3\zeta[\tilde{p},\tilde{p}])_w = 2v^{-1}q\nabla^2\varphi[\xi] - \nabla^3\varphi[\xi,\xi], \tag{3.45i}$$
$$\nabla^3\zeta[\tilde{p},\tilde{p},\tilde{p}] = -v\nabla^3\varphi[\xi,\xi,\xi] + 3q\nabla^2\varphi[\xi,\xi]. \tag{3.45j}$$

Using (3.45), we now derive the directional derivatives of Γ. For convenience, we let $\nabla\Gamma$, $\nabla^2\Gamma$, $\nabla^3\Gamma$ be the derivatives of Γ evaluated at $\tilde{u}$. We define:

$$\sigma \coloneqq -\nabla_v\zeta = \varphi - \nabla\varphi[\mu] \in \mathbb{R}. \tag{3.46}$$

The components of the gradient g of Γ are:

$$g_u = -\zeta^{-1}, \tag{3.47a}$$
$$g_v = \zeta^{-1}\sigma - v^{-1}, \tag{3.47b}$$
$$g_w = \zeta^{-1}\nabla\varphi - w^{-1}. \tag{3.47c}$$

Note (3.47c) follows from (3.27). Differentiating (3.47), the Hessian components are:

$$\nabla^2_{u,u}\Gamma = \zeta^{-2} > 0, \tag{3.48a}$$
$$\nabla^2_{v,u}\Gamma = -\zeta^{-2}\sigma \in \mathbb{R}, \tag{3.48b}$$
$$\nabla^2_{w,u}\Gamma = -\zeta^{-2}\nabla\varphi \in V, \tag{3.48c}$$
$$\nabla^2_{v,v}\Gamma = v^{-2} + \zeta^{-2}\sigma^2 + v^{-1}\zeta^{-1}\nabla^2\varphi[\mu,\mu] > 0, \tag{3.48d}$$
$$\nabla^2_{w,v}\Gamma = \zeta^{-2}\sigma\nabla\varphi - v^{-1}\zeta^{-1}\nabla^2\varphi[\mu] \in V. \tag{3.48e}$$

Differentiating (3.47c) in the direction r and using (3.25):

$$\nabla^2_{w,w}\Gamma[r] = \zeta^{-2}\nabla\varphi[r]\nabla\varphi + v^{-1}\zeta^{-1}\nabla^2\varphi[r] + P(w^{-1})r \in V. \tag{3.49}$$

Let:

$$\chi \coloneqq \zeta^{-1}(p - q\sigma - \nabla\varphi[r]) \in \mathbb{R}. \tag{3.50}$$

The components of the Hessian product H are:

$$H_u = \zeta^{-1}\chi, \tag{3.51a}$$
$$H_v = -\zeta^{-1}\sigma\chi - \zeta^{-1}\nabla^2\varphi[\xi,\mu] + v^{-2}q, \tag{3.51b}$$
$$H_w = -\zeta^{-1}\chi\nabla\varphi + \zeta^{-1}\nabla^2\varphi[\xi] + P(w^{-1})r. \tag{3.51c}$$

Let:

$$\kappa \coloneqq 2\zeta^{-1}(\chi + v^{-1}q)\nabla^2\varphi[\xi] - \zeta^{-1}\nabla^3\varphi[\xi,\xi] \in V. \tag{3.52}$$

The components of the third order directional derivative T are:

$$T_u = -2\zeta^{-1}\chi^2 - v\zeta^{-2}\nabla^2\varphi[\xi,\xi], \tag{3.53a}$$
$$T_v = -T_u\sigma + \langle\kappa,\mu\rangle - \zeta^{-1}\nabla^2\varphi[\xi,\xi] - 2q^2v^{-3}, \tag{3.53b}$$
$$T_w = -T_u\nabla\varphi - \kappa - 2P(w^{-1/2})(P(w^{-1/2})r)^2. \tag{3.53c}$$

Note (3.53c) follows from (3.26).

3.5.2 Inverse Hessian operator

The Hessian of Γ at any point $\tilde{u} \in \operatorname{int}(\mathcal{K}_p)$ is a positive definite linear operator and hence invertible. By treating the components of the Hessian in (3.48) and (3.49) analogously to blocks of a positive definite matrix, we derive the inverse operator. For convenience, we let:

$$Y_u \coloneqq (\nabla^2_{w,w}\Gamma)^{-1}\nabla^2_{w,u}\Gamma, \tag{3.54a}$$
$$Y_v \coloneqq (\nabla^2_{w,w}\Gamma)^{-1}\nabla^2_{w,v}\Gamma, \tag{3.54b}$$
$$Z_{u,u} \coloneqq \nabla^2_{u,u}\Gamma - \langle\nabla^2_{w,u}\Gamma, Y_u\rangle, \tag{3.54c}$$
$$Z_{v,u} \coloneqq \nabla^2_{v,u}\Gamma - \langle\nabla^2_{w,u}\Gamma, Y_v\rangle, \tag{3.54d}$$
$$Z_{v,v} \coloneqq \nabla^2_{v,v}\Gamma - \langle\nabla^2_{w,v}\Gamma, Y_v\rangle. \tag{3.54e}$$

Note $Y_u, Y_v \in V$. We let Z be:

$$Z \coloneqq \begin{bmatrix} Z_{u,u} & Z_{v,u} \\ Z_{v,u} & Z_{v,v} \end{bmatrix} \in \mathbb{S}^2_{\succ}, \tag{3.55}$$

and its inverse is:

$$\bar{Z} \coloneqq Z^{-1} = \frac{1}{Z_{u,u}Z_{v,v} - Z^2_{v,u}} \begin{bmatrix} Z_{v,v} & -Z_{v,u} \\ -Z_{v,u} & Z_{u,u} \end{bmatrix} \in \mathbb{S}^2_{\succ}. \tag{3.56}$$

It can be verified (for example, by analogy to the block symmetric matrix inverse formula) that the inverse Hessian product oracle $\bar{H}$ in (3.42c) is:

$$\bar{H}_u = \bar{Z}_{u,u}(p - \langle Y_u, r\rangle) + \bar{Z}_{v,u}(q - \langle Y_v, r\rangle), \tag{3.57a}$$

$$\bar{H}_v = \bar{Z}_{v,u}(p - \langle Y_u, r\rangle) + \bar{Z}_{v,v}(q - \langle Y_v, r\rangle), \tag{3.57b}$$
$$\bar{H}_w = -\bar{H}_u Y_u - \bar{H}_v Y_v + (\nabla^2_{w,w}\Gamma)^{-1} r. \tag{3.57c}$$

Hence computing $\bar{H}$ is essentially only as difficult as applying the positive definite linear operator $(\nabla^2_{w,w}\Gamma)^{-1}$. We are not aware of a simple expression for $(\nabla^2_{w,w}\Gamma)^{-1}$ in general, but we explore the special cases of separable spectral functions in Section 3.5.3, the negative log-determinant function in Section 3.5.4, and the root-determinant function in Section 3.7.4.

3.5.3 Inverse Hessian operator for the separable spectral case

Suppose $w \succ 0$ has the spectral decomposition (3.5), i.e. w has the eigenvalues $w_1, \ldots, w_d > 0$ and the Jordan frame $c_1, \ldots, c_d$. As in Section 3.3.2, we assume distinct eigenvalues for simplicity. In the special case where φ is a convex separable spectral function, i.e. $\varphi(w) = \sum_{i\in[\![d]\!]} h(w_i)$ for some convex $h : \mathbb{R}_> \to \mathbb{R}$, we show how to compute $\bar{H}$ as efficiently as Hessian product oracle H. For all $i \in [\![d]\!]$, we let h_i, $(\nabla h)_i$, $(\nabla^2 h)_i$, and $(\nabla^3 h)_i$ denote the value and derivatives of h evaluated at μ. We define $m_{i,j} \in \mathbb{R}$ for $i, j \in [\![d]\!]$ as:

$$m_{i,j} := \begin{cases} \zeta^{-1}\frac{(\nabla h)_i - (\nabla h)_j}{w_i - w_j} + w_i^{-1} w_j^{-1} & i \neq j, \\ \zeta^{-1} v^{-1} (\nabla^2 h)_i + w_i^{-2} & i = j. \end{cases} \tag{3.58}$$

Since h is convex, $m_{i,j} > 0, \forall i, j \in [\![d]\!]$. Let $M : V \to V$ be the self-adjoint linear operator:

$$M := v^{-1}\zeta^{-1}\nabla^2\varphi + P(w^{-1}) = \sum_{i,j\in[\![d]\!]:i<j} 4m_{i,j} L(c_i)L(c_j) + \sum_{i\in[\![d]\!]} m_{i,i} P(c_i). \tag{3.59}$$

Using (3.13), we have the self-adjoint inverse operator of M:

$$M^{-1} = \sum_{i,j\in[\![d]\!]:i<j} 4m_{i,j}^{-1} L(c_i)L(c_j) + \sum_{i\in[\![d]\!]} m_{i,i}^{-1} P(c_i). \tag{3.60}$$

Substituting (3.59) into (3.49), we have:

$$\nabla^2_{w,w}\Gamma[r] = \zeta^{-2}\nabla\varphi[r]\nabla\varphi + Mr. \tag{3.61}$$

Note that the first term in (3.61) is analogous to the application (to r) of a low-rank update to M, and that M^{-1} in (3.60) is easy to apply. By analogy to the Sherman-Morrison-Woodbury formula (Deng, 2011, Theorem 1.1), we can derive a simple expression for the inverse operator $(\nabla^2_{w,w}\Gamma)^{-1} r$.

We let:

$$\alpha := M^{-1}\nabla\varphi = \sum_{i\in[\![d]\!]} m_{i,i}^{-1}(\nabla h)_i c_i \in V, \tag{3.62a}$$
$$\gamma := v^{-2}\zeta^{-1} M^{-1}\nabla^2\varphi[w] = v^{-2}\zeta^{-1}\sum_{i\in[\![d]\!]} m_{i,i}^{-1}(\nabla^2 h)_i w_i c_i \succ 0. \tag{3.62b}$$

Noting that $\gamma, w^{-1} \succ 0$ implies $\langle \gamma, w^{-1} \rangle > 0$, we define the scalar constants:

$$k_1 \coloneqq \zeta^2 + \langle \nabla\varphi, \alpha \rangle > 0, \tag{3.63a}$$
$$k_2 \coloneqq \sigma + \langle \nabla\varphi, \gamma \rangle = \sigma + v^{-2}\zeta^{-1}\langle \nabla^2\varphi[w], \alpha \rangle, \tag{3.63b}$$
$$k_3 \coloneqq v^{-2} + v^{-2}\zeta^{-1}\langle \nabla^2\varphi[w], \mu - \gamma \rangle = v^{-2} + v^{-1}\langle \gamma, w^{-1} \rangle > 0. \tag{3.63c}$$

Now using the Sherman-Morrison-Woodbury formula:

$$(\nabla^2_{w,w}\Gamma)^{-1} r = M^{-1}r - \frac{\zeta^{-2}\langle M^{-1}\nabla\varphi, r\rangle}{1 + \zeta^{-2}\langle M^{-1}\nabla\varphi, \nabla\varphi\rangle} M^{-1}\nabla\varphi \tag{3.64a}$$
$$= M^{-1}r - k_1^{-1}\langle \alpha, r\rangle\alpha. \tag{3.64b}$$

Substituting (3.48) and (3.64) into (3.54), we have:

$$Y_u = (\nabla^2_{w,w}\Gamma)^{-1}(-\zeta^{-2}\nabla\varphi) \tag{3.65a}$$
$$= -\zeta^{-2}\alpha + \zeta^{-2}k_1^{-1}\langle \alpha, \nabla\varphi\rangle\alpha \tag{3.65b}$$
$$= -k_1^{-1}\alpha, \tag{3.65c}$$
$$Y_v = (\nabla^2_{w,w}\Gamma)^{-1}(\zeta^{-2}\sigma\nabla\varphi - v^{-1}\zeta^{-1}\nabla^2\varphi[\mu]) \tag{3.65d}$$
$$= -\sigma Y_u - v^{-2}\zeta^{-1}(\nabla^2_{w,w}\Gamma)^{-1}\nabla^2\varphi[w] \tag{3.65e}$$
$$= \sigma k_1^{-1}\alpha - \gamma + v^{-2}\zeta^{-1}k_1^{-1}\langle \alpha, \nabla^2\varphi[w]\rangle\alpha \tag{3.65f}$$
$$= k_1^{-1}k_2\alpha - \gamma, \tag{3.65g}$$
$$Z_{u,u} = \zeta^{-2} - \langle \nabla^2_{w,u}\Gamma, Y_u\rangle \tag{3.65h}$$
$$= \zeta^{-2} - \zeta^{-2}k_1^{-1}\langle \nabla\varphi, \alpha\rangle \tag{3.65i}$$
$$= k_1^{-1}, \tag{3.65j}$$
$$Z_{v,u} = -\zeta^{-2}\sigma - \langle \nabla^2_{w,u}\Gamma, Y_v\rangle \tag{3.65k}$$
$$= -\zeta^{-2}(\sigma - \langle \nabla\varphi, k_1^{-1}k_2\alpha - \gamma\rangle) \tag{3.65l}$$
$$= -\zeta^{-2}(\sigma - k_1^{-1}k_2(k_1 - \zeta^2) + k_2 - \sigma) \tag{3.65m}$$
$$= -k_1^{-1}k_2, \tag{3.65n}$$

and:

$$Z_{v,v} = \nabla^2_{v,v}\Gamma - \langle \nabla^2_{w,v}\Gamma, Y_v\rangle \tag{3.66a}$$
$$= \nabla^2_{v,v}\Gamma + \sigma\langle \nabla^2_{w,u}\Gamma, Y_v\rangle + v^{-2}\zeta^{-1}\langle \nabla^2\varphi[w], Y_v\rangle \tag{3.66b}$$
$$= \nabla^2_{v,v}\Gamma + \sigma(k_1^{-1}k_2 - \zeta^{-2}\sigma) + v^{-2}\zeta^{-1}\langle \nabla^2\varphi[w], Y_v\rangle \tag{3.66c}$$
$$= v^{-2} + v^{-3}\zeta^{-1}\nabla^2\varphi[w, w] + \sigma k_1^{-1}k_2 + v^{-2}\zeta^{-1}\langle \nabla^2\varphi[w], Y_v\rangle \tag{3.66d}$$
$$= v^{-2} + v^{-3}\zeta^{-1}\nabla^2\varphi[w, w] + \sigma k_1^{-1}k_2 + k_1^{-1}k_2(k_2 - \sigma) - v^{-2}\zeta^{-1}\langle \nabla^2\varphi[w], \gamma\rangle \tag{3.66e}$$
$$= v^{-2} + k_1^{-1}k_2^2 + v^{-2}\zeta^{-1}\langle \nabla^2\varphi[w], \mu - \gamma\rangle \tag{3.66f}$$
$$= k_3 + k_1^{-1}k_2^2. \tag{3.66g}$$

For Z in (3.55), we have $\det(Z) = k_1^{-1}k_3$, so its inverse $\bar{Z}$ in (3.56) is:

$$\bar{Z}_{u,u} = k_1(k_3 + k_1^{-1}k_2^2)k_3^{-1} = k_1 + k_2^2 k_3^{-1}, \tag{3.67a}$$
$$\bar{Z}_{u,v} = k_2 k_3^{-1}, \tag{3.67b}$$
$$\bar{Z}_{v,v} = k_3^{-1}. \tag{3.67c}$$

Finally, we substitute (3.65) to (3.67) into (3.57) to derive the inverse Hessian product $\bar{H}$. We let:

$$c_1 \coloneqq p - \langle Y_u, r\rangle = p + k_1^{-1}\langle \alpha, r\rangle, \tag{3.68a}$$
$$c_2 \coloneqq q - \langle Y_v, r\rangle = q - k_1^{-1}k_2\langle \alpha, r\rangle + \langle \gamma, r\rangle. \tag{3.68b}$$

For convenience, we derive $\bar{H}_v$ before $\bar{H}_u$ and $\bar{H}_w$:

$$\bar{H}_v = \bar{Z}_{u,v}c_1 + \bar{Z}_{v,v}c_2 \tag{3.69a}$$
$$= k_3^{-1}(k_2 c_1 + c_2) \tag{3.69b}$$
$$= k_3^{-1}(k_2 p + q + \langle \gamma, r\rangle), \tag{3.69c}$$
$$\bar{H}_u = \bar{Z}_{u,u}c_1 + \bar{Z}_{u,v}c_2 \tag{3.69d}$$
$$= (\bar{Z}_{u,u} - \bar{Z}_{u,v}k_2)c_1 + \bar{Z}_{u,v}k_3\bar{H}_v \tag{3.69e}$$
$$= k_1 p + k_2\bar{H}_v + \langle \alpha, r\rangle, \tag{3.69f}$$
$$\bar{H}_w = -\bar{H}_u Y_u - \bar{H}_v Y_v + (\nabla^2_{w,w})^{-1}r \tag{3.69g}$$
$$= \bar{H}_u k_1^{-1}\alpha - \bar{H}_v(k_1^{-1}k_2\alpha - \gamma) + M^{-1}r - k_1^{-1}\langle \alpha, r\rangle\alpha \tag{3.69h}$$
$$= p\alpha + \bar{H}_v\gamma + M^{-1}r. \tag{3.69i}$$

In Section 3.8.5, we compare the efficiency and numerical performance of the closed form formula for $\bar{H}$ in (3.69) against a naive approach to computing $\bar{H}$ that performs a Cholesky factorization of an explicit Hessian matrix and uses a direct linear solve. The closed form formula is faster and more scalable, more memory-efficient, more reliable to compute (as the Cholesky decomposition can fail), and more numerically accurate.

3.5.4 Oracles for the log-determinant case

We now specialize the oracles derived in Sections 3.5.1 and 3.5.3 for the separable spectral function $\varphi(w) = -\operatorname{logdet}(w) = -\sum_{i\in[\![d]\!]}\log(w_i)$. In Section 3.6, we show that Γ is an LHSCB in this case. We let:

$$\hat{\xi} \coloneqq P(w^{-1/2})\xi = v^{-1}P(w^{-1/2})(-v^{-1}qw + r) = v^{-1}(-v^{-1}qe + \hat{r}) \in V. \tag{3.70}$$

Using (3.27), (3.28a) and (3.28b), we have:

$$\nabla\varphi = -\mu^{-1} = -vP(w^{-1/2})e, \tag{3.71a}$$
$$\nabla^2\varphi[\xi] = v^2P(w^{-1})\xi = v^2P(w^{-1/2})\hat{\xi}, \tag{3.71b}$$

$$\nabla^3\varphi[\xi,\xi] = -2v^3P(w^{-1/2})\hat{\xi}^2. \tag{3.71c}$$

The constants from (3.46) and (3.50) have the form:

$$\sigma = \varphi + d, \tag{3.72a}$$
$$\chi = \zeta^{-1}(p - q\sigma + v\,\mathrm{tr}(\hat{r})). \tag{3.72b}$$

From (3.47), the w component of the gradient is:

$$g_w = -(1 + v\zeta^{-1})w^{-1}. \tag{3.73}$$

From (3.51), the v and w components of the Hessian product are:

$$H_v = -\zeta^{-1}\sigma\chi - v\zeta^{-1}\,\mathrm{tr}(\hat{\xi}) + v^{-2}q, \tag{3.74a}$$
$$H_w = P(w^{-1/2})(v\zeta^{-1}\chi e + v^2\zeta^{-1}\hat{\xi} + \hat{r}). \tag{3.74b}$$

From (3.53), the third order directional derivative is:

$$T_u = -2\zeta^{-1}\chi^2 - v^3\zeta^{-2}\,\mathrm{tr}(\hat{\xi}^2), \tag{3.75a}$$
$$T_v = -T_u\sigma + v\zeta^{-1}(2(\chi + v^{-1}q)\,\mathrm{tr}(\hat{\xi}) + v\,\mathrm{tr}(\hat{\xi}^2)) - 2v^{-3}q^2, \tag{3.75b}$$
$$T_w = P(w^{-1/2})(T_u ve - 2\zeta^{-1}v^2((\chi + v^{-1}q)\hat{\xi} + v\hat{\xi}^2) - 2\hat{r}^2). \tag{3.75c}$$

We derive the inverse Hessian product $\bar{H}$ by specializing the separable case in (3.69). We let:

$$\check{r} \coloneqq P(w^{1/2})r \in V, \tag{3.76a}$$
$$\theta \coloneqq v^2(\zeta + (1+d)v)^{-1}. \tag{3.76b}$$

From (3.62) and (3.63), we have:

$$M^{-1} = \zeta(\zeta + v)^{-1}P(w), \tag{3.77a}$$
$$\alpha = -v\zeta(\zeta + v)^{-1}w, \tag{3.77b}$$
$$\gamma = (\zeta + v)^{-1}w, \tag{3.77c}$$
$$k_1 = \zeta^2 + dv^2\zeta(\zeta + v)^{-1}, \tag{3.77d}$$
$$k_2 = \varphi + d\zeta(\zeta + v)^{-1}, \tag{3.77e}$$
$$k_3^{-1} = (\zeta + v)\theta. \tag{3.77f}$$

For convenience, we derive $\bar{H}_v$ before $\bar{H}_u$ and $\bar{H}_w$ as in (3.69):

$$\bar{H}_v = k_3^{-1}(k_2p + q + \langle\gamma, r\rangle) \tag{3.78a}$$
$$= (\zeta + v)\theta((\varphi + d\zeta(\zeta + v)^{-1})p + q + \langle r, w\rangle(\zeta + v)^{-1}) \tag{3.78b}$$
$$= \theta((\zeta + v)(\varphi p + q) + d\zeta p + \mathrm{tr}(\check{r})), \tag{3.78c}$$
$$\bar{H}_u = k_1p + (\varphi + d\zeta(\zeta + v)^{-1})\bar{H}_v + \langle\alpha, r\rangle \tag{3.78d}$$

$$= (\zeta^2 + dv^2\zeta(\zeta+v)^{-1})p + (\varphi + d\zeta(\zeta+v)^{-1})\bar{H}_v + \langle -v\zeta(\zeta+v)^{-1}w, r\rangle \tag{3.78e}$$

$$= \zeta(\zeta+v)^{-1}(dv^2 p + d\bar{H}_v - v\langle w, r\rangle) + \zeta^2 p + \varphi\bar{H}_v, \tag{3.78f}$$

$$\bar{H}_w = p\alpha + \bar{H}_v\gamma + M^{-1}r \tag{3.78g}$$

$$= -pv\zeta(\zeta+v)^{-1}w + \bar{H}_v(\zeta+v)^{-1}w + \zeta(\zeta+v)^{-1}P(w)r \tag{3.78h}$$

$$= (\zeta+v)^{-1}P(w^{1/2})((-\zeta pv + \bar{H}_v)e + \zeta\check{r}). \tag{3.78i}$$

Recall that in Section 3.5.3, we used the simplifying assumption of distinct eigenvalues, but for the negative log-determinant case this is not necessary. Note that if it is possible to apply $P(w^{1/2})$ and $P(w^{-1/2})$ without accessing the eigenvalues of w, then all four oracles can be computed without an explicit eigendecomposition. For example, in our implementation for $V = \mathbb{S}^d$ and $V = \mathbb{H}^d$, only a Cholesky factorization of w is needed. This is unlike the more general separable spectral function case, where the explicit eigenvalues of w are needed.

3.6 Matrix monotone derivative cones

After defining the matrix monotone property of a function in Section 3.6.1, we introduce the matrix monotone derivative cone $\mathcal{K}_{\text{MMD}}$ in Section 3.6.2. $\mathcal{K}_{\text{MMD}}$ is a special case of the epigraph-perspective cone $\mathcal{K}_p$ with a separable spectral function φ. In Section 3.6.4, we prove that our barrier function Γ for $\mathcal{K}_{\text{MMD}}$ is an LHSCB.

3.6.1 Matrix monotonicity

A function f is *matrix monotone* (or operator monotone) if $w_a \succeq w_b \succeq 0$ implies $f(w_a) \succeq f(w_b)$ for all $w_a, w_b \in \mathbb{S}^d$ for all integers d. The following integral representation result is attributed to Löwner (1934) (see e.g. Kwong (1989, Theorem 1) and Furuta (2008, Theorem L)). A function $f : \mathbb{R}_> \to \mathbb{R}$ is matrix monotone in $\mathbb{R}_>$ if and only if it has the representation:

$$f(x) = \alpha + \beta x + \int_0^\infty \frac{x}{x+t}\,\mathrm{d}\rho(t) = \alpha + \beta x + \int_0^\infty \left(1 - \frac{t}{x+t}\right)\mathrm{d}\rho(t), \tag{3.79}$$

where $\alpha \in \mathbb{R}, \beta \in \mathbb{R}_\geq$ and ρ is a positive measure on $\mathbb{R}_>$ such that $\int_0^\infty (1+t)^{-1}\,\mathrm{d}\rho(t) < \infty$.

This result implies that, for a cone of squares $\mathcal{Q}$ of a Jordan algebra, and for $w \in \operatorname{int}(\mathcal{Q})$ with the spectral decomposition $w = \sum_{i\in[\![d]\!]} w_i c_i$, we have:

$$f(w) = \sum_{i\in[\![d]\!]} f(w_i)c_i \tag{3.80a}$$

$$= \alpha e + \beta w + \int_0^\infty \sum_{i\in[\![d]\!]} \left(1 - \frac{t}{w_i + t}\right)c_i\,\mathrm{d}\rho(t) \tag{3.80b}$$

$$= \alpha e + \beta w + \int_0^\infty (e - t(w+te)^{-1})\,\mathrm{d}\rho(t). \tag{3.80c}$$

This is similar to the representation in Faybusovich and Tsuchiya (2017, Page 1520).

3.6.2 Cone definition

Let $h : \mathbb{R}_> \to \mathbb{R}$ be a convex C^3-smooth function. We assume that that the first derivative of h, ∇h, is a matrix monotone function. This also implies that h is convex. We call such functions *matrix monotone derivative* (MMD) functions.

In Table 3.1, we give some common examples of MMD functions, with abbreviated names in the first column. We also give ∇h, the domain of the convex conjugate h^* (defined in (3.33)), and a closed form formula for h^*. Due to Carlen (2010, Theorem 2.6 (Löwner-Heinz Theorem)), the functions $x \to \log(x)$, $x \to -x^p$ for $p \in [-1, 0]$, and $x \to x^p$ for $p \in [0, 1]$ are matrix monotone. This implies that in Table 3.1, each function in the ∇h column is matrix monotone. Note that NegSqrt is equivalent to NegPower for $p = 1/2$; we highlight NegSqrt as an interesting case, for which the conjugate h^* is a positive rescaling of the inverse function. Note we exclude the case $p = 1$ in NegPower and Power because it is homogeneous (h is linear). More examples of matrix monotone functions can be found in Kwong (1989) and Furuta (2008).

Table 3.1: Examples of MMD functions. We let $q := p/(p-1)$, which gives $q \in (-\infty, 0)$ for $p \in (0, 1)$ in the NegPower case, or $q \in [2, \infty)$ for $p \in (1, 2]$ in the Power case. We also let $x_- := \max(-x, 0)$ in the Power case.

MMD function	h	∇h	$\operatorname{dom}(h^*)$	h^*
NegLog	$-\log(x)$	$-x^{-1}$	$\mathbb{R}_>$	$-1 - \log(x)$
NegEntropy	$x \log(x)$	$1 + \log(x)$	$\mathbb{R}$	$\exp(-1 - x)$
NegSqrt	$-\sqrt{x}$	$-\frac{1}{2}x^{-1/2}$	$\mathbb{R}_>$	$\frac{1}{4}x^{-1}$
NegPower, $p \in (0, 1)$	$-x^p$	$-px^{p-1}$	$\mathbb{R}_\geq$	$-(p-1)(x/p)^q$
Power, $p \in (1, 2]$	x^p	px^{p-1}	$\mathbb{R}$	$(p-1)(x_-/p)^q$

Suppose $\mathcal{Q}$ is the cone of squares of a Jordan algebra V with rank d. Let $\varphi : \operatorname{int}(\mathcal{Q}) \to \mathbb{R}$ be the C^3-smooth function $\varphi(w) = \operatorname{tr}(h(w)) = \sum_{i \in [\![d]\!]} h(w_i)$, which is a convex separable spectral function (see Section 3.3.2). As in Section 3.4.2, we let $\tilde{u} = (u, v, w) \in \mathcal{E} = \mathbb{R} \times \mathbb{R}_> \times \operatorname{int}(\mathcal{Q})$. The function $\zeta : \mathcal{E} \to \mathbb{R}$ has the form:

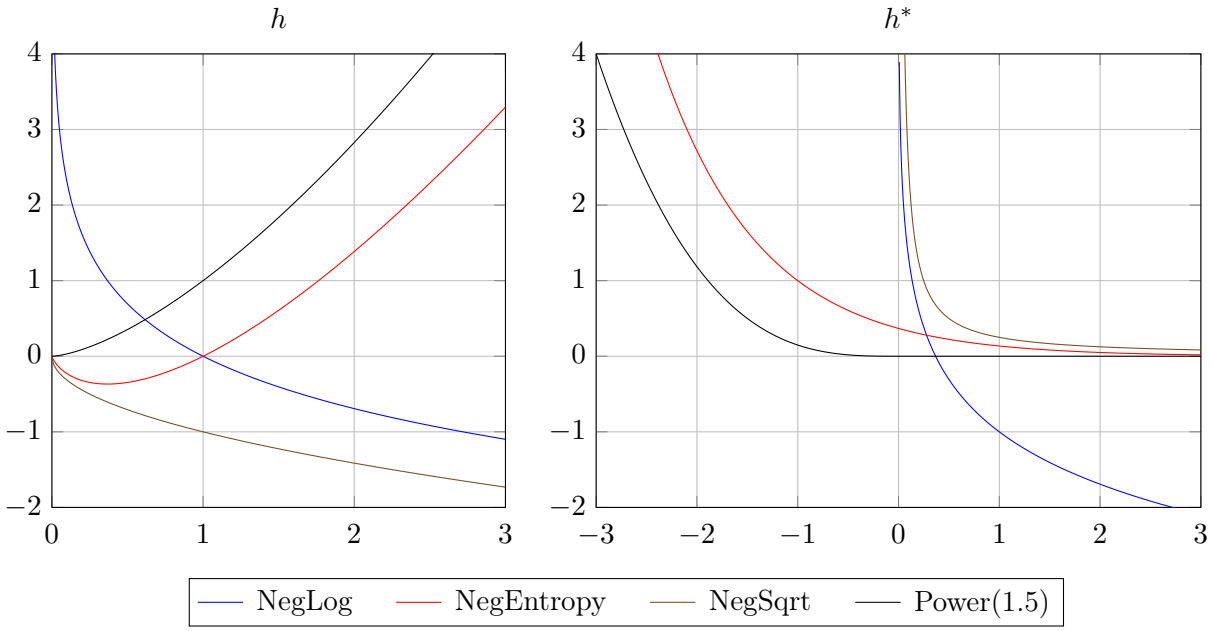
$$\zeta(\tilde{u}) := u - v \operatorname{tr}(h(v^{-1}w)). \tag{3.81}$$

We define the matrix monotone derivative cone $\mathcal{K}_{\text{MMD}}$, a special case of the epigraph-perspective cone $\mathcal{K}_p$ in (3.31), as:

$$\mathcal{K}_{\text{MMD}} := \operatorname{cl}\{\tilde{u} \in \mathcal{E} : u \geq v \operatorname{tr}(h(v^{-1}w))\}, \tag{3.82}$$

which is a proper cone. Note that the negative log-determinant cone, whose barrier function we examined in Section 3.5.4, is a special case of $\mathcal{K}_{\text{MMD}}$ where h is the NegLog function from Table 3.1. For the separable case, the convex conjugate $\varphi^* : V \to \mathbb{R} \cup \infty$ (see (3.33)) of φ is $\varphi^*(r) = \operatorname{tr}(h^*(r))$.

Figure 3.1: Graphs of some MMD functions (h, left) and their conjugates (h^*, right) from Table 3.1.



So from (3.34), the dual cone is:

$$\mathcal{K}^*_{\mathrm{MMD}} := \mathrm{cl}\{\tilde{u} \in \mathbb{R}_> \times \mathbb{R} \times V : v \geq u \operatorname{tr}(h^*(u^{-1}w))\}. \tag{3.83}$$

3.6.3 Derivatives of the separable spectral function

Suppose $w \succ 0$. Since $\varphi(w) = \operatorname{tr}(h(w))$ and ∇h is matrix monotone, from (3.80) we can write the gradient:

$$\nabla\varphi(w) = \nabla h(w) = \alpha e + \beta w + \int_0^\infty (e - t(w + te)^{-1}) \, \mathrm{d}\rho(t). \tag{3.84}$$

Note that $w + te \succ 0$ for $t \geq 0$, so $(w + te)^{-1}$ is well-defined. Differentiating (3.84) in the direction $r \in V$, we have the second order directional derivative (using (3.25)):

$$\nabla^2\varphi(w)[r] = \beta r + \int_0^\infty tP((w + te)^{-1})r \, \mathrm{d}\rho(t), \tag{3.85}$$

and the third order directional derivative (using (3.26)):

$$\nabla^3\varphi(w)[r, r] = -2 \int_0^\infty tP((w + te)^{-1/2})(P((w + te)^{-1/2})r)^2 \, \mathrm{d}\rho(t). \tag{3.86}$$

3.6.4 Self-concordant barrier

For $\mathcal{K}_{\mathrm{MMD}}$, the LHB $\Gamma : \mathrm{int}(\mathcal{K}_{\mathrm{MMD}}) \to \mathbb{R}$ from (3.41) has the form:

$$\Gamma(\tilde{u}) := -\log(\zeta(\tilde{u})) - \log(v) - \operatorname{logdet}(w). \tag{3.87}$$

We describe easily-computable oracles for this Γ in Section 3.5, including an inverse Hessian product $\bar{H}$ in Section 3.5.3 that is as easy to compute as the Hessian product H (since φ is a separable spectral function).

We note that Faybusovich and Tsuchiya (2017) derive a $(1+d)$-self-concordant barrier for the related convex (but not conic) set $\mathcal{S}$:

$$\mathcal{S} \coloneqq \operatorname{cl}\{(u, w) \in \mathbb{R} \times \operatorname{int}(\mathcal{Q}) : u - \varphi(w) \geq 0\}. \tag{3.88}$$

$\mathcal{K}_{\text{MMD}}$ is the conic hull of $\mathcal{S}$. In Proposition 3.6.1, we prove that our barrier Γ in (3.87) is self-concordant, hence it is an LHSCB for $\mathcal{K}_{\text{MMD}}$ with parameter $2+d$. This small additive increment of one in the barrier parameter is in sharp contrast to generic conic hull results, which give barriers with a large multiplicative factor in the parameter (for example, Nesterov and Nemirovski (1994, Proposition 5.1.4) yields the parameter $800(1+d)$). Since the optimal barrier parameter for $\operatorname{cl}(\mathcal{E})$ is $1+d$, our parameter cannot be reduced by more than one.

Proposition 3.6.1. Γ in (3.87) is a $(2+d)$-LHSCB for $\mathcal{K}_{\text{MMD}}$ in (3.82).

Proof. We show that ζ in (3.81) is $(\mathbb{R}_{\geq}, 1)$-compatible with the domain $\mathcal{E}$, in the sense of Nesterov and Nemirovski (1994, Definition 5.1.1). This follows if (i) ζ is C^3-smooth on $\mathcal{E}$, (ii) ζ is concave with respect to $\mathbb{R}_{\geq}$, (iii) for any point $\tilde{u} \in \operatorname{int}(\mathcal{K}_{\text{MMD}})$ and direction $\tilde{p} = (p, q, r) \in \mathbb{R} \times \mathbb{R} \times V$ satisfying $v \pm q \geq 0$ and $w \pm r \succeq 0$ it holds that:

$$\nabla^3 \zeta(\tilde{u})[\tilde{p}, \tilde{p}, \tilde{p}] \leq -3\nabla^2 \zeta(\tilde{u})[\tilde{p}, \tilde{p}]. \tag{3.89}$$

Suppose $v \pm q \geq 0$ and $w \pm r \succeq 0$. As in (3.44), we let $\mu \coloneqq \mu(v, w) = v^{-1}w \succ 0$ and $\xi \coloneqq v^{-1}(r - q\mu)$. From (3.45), the second and third order directional derivatives of ζ at $\tilde{u}$ in direction $\tilde{p}$ are:

$$\nabla^2 \zeta(\tilde{u})[\tilde{p}, \tilde{p}] = -v\nabla^2 \varphi(\mu)[\xi, \xi], \tag{3.90a}$$

$$\nabla^3 \zeta(\tilde{u})[\tilde{p}, \tilde{p}, \tilde{p}] = -v\nabla^3 \varphi(\mu)[\xi, \xi, \xi] + 3q\nabla^2 \varphi(\mu)[\xi, \xi]. \tag{3.90b}$$

Since φ is convex and C^3-smooth on $\operatorname{int}(\mathcal{Q})$ by assumption, (3.81) and (3.90) imply that ζ is concave and C^3-smooth on $\mathcal{E}$. It remains to show that (3.89) holds.

For $t \geq 0$, let:

$$a(t) \coloneqq \mu + te \succ 0, \tag{3.91a}$$

$$\bar{a}(t) \coloneqq a(t)^{-1/2} \succ 0, \tag{3.91b}$$

$$\hat{\xi}(t) \coloneqq P(\bar{a}(t))\xi. \tag{3.91c}$$

By the integral representation result from Section 3.6.1 (Löwner, 1934), there exists a positive measure ρ and $\beta \geq 0$ such that the directional derivatives of φ are:

$$\nabla^2 \varphi(\mu)[\xi, \xi] = \beta \operatorname{tr}(\xi^2) + \int_0^\infty t \operatorname{tr}(\hat{\xi}(t)^2) \, \mathrm{d}\rho(t), \tag{3.92a}$$

$$\nabla^3\varphi(\mu)[\xi,\xi,\xi] = -2\int_0^\infty t\,\mathrm{tr}(\hat{\xi}(t)^3)\;\mathrm{d}\rho(t), \tag{3.92b}$$

using (3.85) and (3.86). From (3.90) and (3.92), the compatibility condition (3.89) is equivalent to nonnegativity of:

$$-3\nabla^2\zeta(\tilde{u})[\tilde{p},\tilde{p}] - \nabla^3\zeta(\tilde{u})[\tilde{p},\tilde{p},\tilde{p}] \tag{3.93a}$$

$$= 3(v-q)\beta\,\mathrm{tr}(\xi^2) + \int_0^\infty t(3(v-q)\,\mathrm{tr}(\hat{\xi}(t)^2) - 2v\,\mathrm{tr}(\hat{\xi}(t)^3))\;\mathrm{d}\rho(t). \tag{3.93b}$$

Since $v \geq q$, the first term in (3.93b) is nonnegative. The second term (the integral) is nonnegative if for all $t \geq 0$, the following inner term is nonnegative:

$$3(v-q)\,\mathrm{tr}(\hat{\xi}(t)^2) - 2v\,\mathrm{tr}(\hat{\xi}(t)^3) = \langle \hat{\xi}(t)^2, 3(v-q)e - 2v\hat{\xi}(t)\rangle. \tag{3.94}$$

By self-duality of $\mathcal{Q}$, (3.94) is nonnegative if $3(v-q)e - 2v\hat{\xi}(t) \succeq 0$, which we now prove. For $t \geq 0$, let:

$$b(t) \coloneqq (1-v^{-1}q)a(t) - \xi = v^{-1}(w-r) + t(1-v^{-1}q)e. \tag{3.95}$$

Since $w \succeq r$ and $1 - v^{-1}q \geq 0$, we have $b(t) \succeq 0$. Hence we have (using (3.95)):

$$(1-v^{-1}q)e - \hat{\xi}(t) = P(\bar{a}(t))b(t) \succeq 0, \tag{3.96}$$

since $\bar{a}(t) \succ 0$ implies $P(\bar{a}(t))$ is an automorphism on $\mathcal{Q}$ (see Faraut and Koranyi (1998, Page 48)). Therefore using (3.96):

$$3(v-q)e - 2v\hat{\xi}(t) \succeq 3(v-q)e - 2v(1-v^{-1}q)e = (v-q)e \succeq 0. \tag{3.97}$$

So (3.94) is nonnegative, which implies the integral term in (3.93b) is nonnegative.

Thus (3.93b) is nonnegative, so (3.89) holds and compatibility is proved. Now by Nesterov and Nemirovski (1994, Proposition 5.1.7), Γ is an LHSCB for $\mathcal{K}_{\mathrm{MMD}}$ with parameter $2+d$.

□

3.7 Root-determinant cones

In Section 3.7.1, we define the root-determinant cone $\mathcal{K}_{\mathrm{rtdet}}$, which is the hypograph of the homogeneous nonseparable spectral root-determinant function. After expressing the derivatives of this function in Section 3.7.2, we prove that our barrier function Γ for $\mathcal{K}_{\mathrm{rtdet}}$ is an LHSCB in Section 3.7.3, and we derive easily-computable barrier oracles in Section 3.7.4.

3.7.1 Cone definition

Suppose $\mathcal{Q}$ is a cone of squares of a Jordan algebra V with rank d. Let $\varphi : \mathcal{Q} \to \mathbb{R}_{\geq}$ denote the root-determinant function (or the geometric mean of the eigenvalues):

$$\varphi(w) \coloneqq \det(w)^{1/d} = \prod_{i \in [\![d]\!]} w_i^{1/d}, \tag{3.98}$$

which is a concave, homogeneous nonseparable spectral function (see Section 3.3.1). We let $\tilde{u} \coloneqq (u, w) \in \mathcal{E} = \mathbb{R} \times \mathcal{Q}$. The function $\zeta : \mathcal{E} \to \mathbb{R}$ has the form:

$$\zeta(\tilde{u}) \coloneqq \varphi(w) - u. \tag{3.99}$$

We define the root-determinant cone $\mathcal{K}_{\text{rtdet}}$ and its dual cone as:

$$\mathcal{K}_{\text{rtdet}} \coloneqq \{(u, w) \in \mathbb{R} \times \mathcal{Q} : u \leq \det(w)^{1/d}\}, \tag{3.100a}$$

$$\mathcal{K}^*_{\text{rtdet}} \coloneqq \{(u, w) \in \mathbb{R}_{\leq} \times \mathcal{Q} : -d^{-1}u \leq \det(w)^{1/d}\}. \tag{3.100b}$$

We note that $\mathcal{K}_{\text{rtdet}}$ is a hypograph modification of the epigraph cone $\mathcal{K}_h$ in (3.30), and it is a primitive proper cone. $\mathcal{K}^*_{\text{rtdet}}$ can be derived by modifying the steps we use to derive $\mathcal{K}^*_h$ in (3.38) and using the convex conjugate of the negative root-determinant function.

3.7.2 Derivatives of root-determinant

Suppose $w \succ 0$. Since $\varphi(w) = \exp(d^{-1} \operatorname{logdet}(w))$, applying the chain rule and using (3.27) gives us the gradient:

$$\nabla\varphi(w) = d^{-1}\varphi(w)w^{-1}. \tag{3.101}$$

Let $r \in V$ and $\hat{r} \coloneqq P(w^{-1/2})r \in V$. Using the product rule on (3.101), we have the second order directional derivative:

$$\begin{aligned}
\nabla^2\varphi(w)[r] &= d^{-2}\langle w^{-1}, r\rangle \nabla\varphi(w) + d^{-1}\varphi(w)\nabla_w(w^{-1})[r] && \text{(3.102a)}\\
&= d^{-1}\varphi(w)\operatorname{tr}(\hat{r})w^{-1} - d^{-1}\varphi(w)P(w^{-1})r && \text{(3.102b)}\\
&= d^{-1}\varphi(w)(d^{-1}\operatorname{tr}(\hat{r})w^{-1} - P(w^{-1})r) && \text{(3.102c)}\\
&= d^{-1}\varphi(w)P(w^{-1/2})(d^{-1}\operatorname{tr}(\hat{r})e - \hat{r}). && \text{(3.102d)}
\end{aligned}$$

Note (3.102b) follows from (3.25). Finally, using the product rule on (3.102c), we have the third order directional derivative:

$$\begin{aligned}
\nabla^3\varphi(w)[r, r] &= d^{-1}\langle d^{-1}\operatorname{tr}(\hat{r})w^{-1} - P(w^{-1})r, r\rangle \nabla\varphi(w) + d^{-1}\varphi(w)(\\
&\quad d^{-1}(\langle w^{-1}, r\rangle \nabla_w(\langle w^{-1}, r\rangle) + \operatorname{tr}(\hat{r})\nabla_w(w^{-1})[r]) - \nabla_w(P(w^{-1})r)[r]) && \text{(3.103a)}\\
&= d^{-2}\varphi(w)(d^{-1}\operatorname{tr}(\hat{r})^2 - \operatorname{tr}(\hat{r}^2))w^{-1} + d^{-1}\varphi(w)(\\
&\quad -2d^{-1}\operatorname{tr}(\hat{r})P(w^{-1})r + 2P(w^{-1/2})(P(w^{-1/2})r)^2) && \text{(3.103b)}
\end{aligned}$$

$$= d^{-1}\varphi(w)P(w^{-1/2})(d^{-1}(d^{-1}\operatorname{tr}(\hat{r})^2 - \operatorname{tr}(\hat{r}^2))e - 2d^{-1}\operatorname{tr}(\hat{r})\hat{r} + 2\hat{r}^2). \tag{3.103c}$$

Note (3.103b) follows from (3.26).

3.7.3 Self-concordant barrier

For $\mathcal{K}_{\text{rtdet}}$, the LHB $\Gamma : \operatorname{int}(\mathcal{K}_{\text{rtdet}}) \to \mathbb{R}$ from (3.41) has the form:

$$\Gamma(\tilde{u}) := -\log(\zeta(\tilde{u})) - \operatorname{logdet}(w). \tag{3.104}$$

In Proposition 3.7.1 we show that Γ is self-concordant with parameter $1 + d$. Since the optimal barrier parameter for $\mathcal{E}$ is d, our parameter cannot be reduced by more than one.

Proposition 3.7.1. Γ in (3.104) is a $(1 + d)$-LHSCB for $\mathcal{K}_{\text{rtdet}}$ in (3.100a).

Proof. Note $\Psi(\tilde{u}) := -\operatorname{logdet}(w)$ is a d-LHSCB for $\mathcal{E}$. We show that ζ in (3.99) is $(\mathbb{R}_{\geq}, 1)$-compatible with the barrier Ψ in the sense of Nesterov and Nemirovski (1994, Definition 5.1.2). Compatibility follows if (i) ζ is C^3-smooth on $\operatorname{int}(\mathcal{E})$, (ii) concave with respect to $\mathbb{R}_{\geq}$, (iii) for any point $\tilde{u} \in \operatorname{int}(\mathcal{K}_{\text{rtdet}})$ and direction $\tilde{p} = (p, r) \in \mathbb{R} \times V$ it holds that:

$$\nabla^3\zeta(\tilde{u})[\tilde{p}, \tilde{p}, \tilde{p}] \leq -3(\nabla^2\Psi(\tilde{u})[\tilde{p}, \tilde{p}])^{1/2}\nabla^2\zeta(\tilde{u})[\tilde{p}, \tilde{p}]. \tag{3.105}$$

Suppose $\tilde{u} \in \operatorname{int}(\mathcal{K}_{\text{rtdet}})$. From (3.99), we have:

$$\nabla^2\zeta(\tilde{u})[\tilde{p}, \tilde{p}] = \nabla^2\varphi(w)[r, r], \tag{3.106a}$$
$$\nabla^3\zeta(\tilde{u})[\tilde{p}, \tilde{p}, \tilde{p}] = \nabla^3\varphi(w)[r, r, r]. \tag{3.106b}$$

Since φ is concave and C^3-smooth on $\operatorname{int}(\mathcal{Q})$, (3.106) implies ζ is concave and C^3-smooth on $\operatorname{int}(\mathcal{E})$. It remains to show that (3.105) holds.

Let $\sigma \in \mathbb{R}^d$ be the eigenvalues of $\hat{r} := P(w^{-1/2})r$. Then using (3.28a):

$$(\nabla^2\Psi(\tilde{u})[\tilde{p}, \tilde{p}])^{1/2} = \operatorname{tr}(\hat{r}^2)^{1/2} = \|\sigma\|. \tag{3.107}$$

Let $m_k := d^{-1}\operatorname{tr}(\hat{r}^k), \forall k \in [\![3]\!]$, and let $\delta_i := \sigma_i - m_1, \forall i \in [\![d]\!]$. By the formulae for variance and skewness, we have:

$$m_2 - m_1^2 = d^{-1}\sum_{i \in [\![d]\!]} \delta_i^2, \tag{3.108a}$$
$$m_3 - 3m_1m_2 + 2m_1^3 = d^{-1}\sum_{i \in [\![d]\!]} \delta_i^3. \tag{3.108b}$$

For convenience, let $\varphi := \varphi(w) > 0$ be a constant. Using (3.102) and (3.108a), we have:

$$\nabla^2\varphi(w)[r, r] = d^{-1}\varphi\langle P(w^{-1/2})(d^{-1}\operatorname{tr}(\hat{r})e - \hat{r}), r\rangle \tag{3.109a}$$
$$= -\varphi(d^{-1}\operatorname{tr}(\hat{r}^2) - d^{-2}\operatorname{tr}(\hat{r})^2) \tag{3.109b}$$

$$= -\varphi(m_2 - m_1^2) \tag{3.109c}$$
$$= -d^{-1}\varphi \sum_{i \in [\![d]\!]} \delta_i^2 \leq 0. \tag{3.109d}$$

Similarly, using (3.103) and (3.108):

$$\nabla^3 \varphi(w)[r,r,r] = d^{-1}\varphi\langle d^{-1}(d^{-1}\operatorname{tr}(\hat{r})^2 - \operatorname{tr}(\hat{r}^2))e - 2d^{-1}\operatorname{tr}(\hat{r})\hat{r} + 2\hat{r}^2, \hat{r}\rangle \tag{3.110a}$$
$$= d^{-1}\varphi(d^{-1}(d^{-1}\operatorname{tr}(\hat{r})^2 - \operatorname{tr}(\hat{r}^2))\operatorname{tr}(\hat{r}) - 2d^{-1}\operatorname{tr}(\hat{r})\operatorname{tr}(\hat{r}^2) + 2\operatorname{tr}(\hat{r}^3)) \tag{3.110b}$$
$$= \varphi(m_1^3 - 3m_1 m_2 + 2m_3) \tag{3.110c}$$
$$= \varphi(3m_1(m_2 - m_1^2) + 2(m_3 - 3m_1 m_2 + 2m_1^3)) \tag{3.110d}$$
$$= d^{-1}\varphi \sum_{i \in [\![d]\!]} (3m_1\delta_i^2 + 2\delta_i^3) \tag{3.110e}$$
$$= d^{-1}\varphi \sum_{i \in [\![d]\!]} \delta_i^2(m_1 + 2\sigma_i). \tag{3.110f}$$

Finally, using (3.107), (3.109) and (3.110) the compatibility condition (3.105) is equivalent to nonnegativity of:

$$-\nabla^3\zeta(\tilde{u})[\tilde{p},\tilde{p},\tilde{p}] - 3(\nabla^2\Psi(\tilde{u})[\tilde{p},\tilde{p}])^{1/2}\nabla^2\zeta(\tilde{u})[\tilde{p},\tilde{p}] \tag{3.111a}$$
$$= -d^{-1}\varphi \sum_{i \in [\![d]\!]} \delta_i^2(m_1 + 2\sigma_i) + 3\|\sigma\| d^{-1}\varphi \sum_{i \in [\![d]\!]} \delta_i^2 \tag{3.111b}$$
$$= d^{-1}\varphi \sum_{i \in [\![d]\!]} \delta_i^2(\|\sigma\| - m_1 + 2(\|\sigma\| - \sigma_i)). \tag{3.111c}$$

Clearly, $d^{-1}\varphi\delta_i^2 \geq 0$ and $\sigma_i \leq \|\sigma\|$ for all $i \in [\![d]\!]$. We have $m_1 \leq d^{-1}\|\sigma\|_1 \leq d^{-1/2}\|\sigma\| \leq \|\sigma\|$. Hence (3.111c) is nonnegative.

Thus (3.105) holds and compatibility is proved. Now by Nesterov and Nemirovski (1994, Proposition 5.1.7), Γ is a $(1+d)$-LHSCB for $\mathcal{K}_{\text{rtdet}}$.

□

3.7.4 Evaluating barrier oracles

Using the derivatives of φ from Section 3.7.2, we derive easily-computable oracles for the LHSCB (3.104). Let $\tilde{u} \in \operatorname{int}(\mathcal{K}_{\text{rtdet}})$ and $\tilde{p} = (p, r) \in \mathbb{R} \times V$. For convenience, let $\varphi := \varphi(w) > 0$ be a constant. We define the scalar constants:

$$\eta := d^{-1}\varphi\zeta^{-1}, \tag{3.112a}$$
$$\theta := 1 + \eta, \tag{3.112b}$$
$$\chi := -\zeta^{-1}p + \eta\operatorname{tr}(\hat{r}), \tag{3.112c}$$
$$\tau := \chi - d^{-1}\operatorname{tr}(\hat{r}), \tag{3.112d}$$
$$\upsilon := \operatorname{tr}(\hat{r}^2) - d^{-1}\operatorname{tr}(\hat{r})^2. \tag{3.112e}$$

Note that:

$$\nabla_u(\zeta^{-1}) = \zeta^{-2}, \tag{3.113a}$$
$$\nabla_u\eta = \zeta^{-1}\eta, \tag{3.113b}$$
$$\nabla_u\chi = \zeta^{-1}\chi, \tag{3.113c}$$
$$\nabla_w(\zeta^{-1}) = -\zeta^{-2}\nabla\varphi(w) = -\zeta^{-1}\eta w^{-1}, \tag{3.113d}$$
$$\nabla_w\eta = \eta(d^{-1} - \eta)w^{-1}, \tag{3.113e}$$
$$\nabla_w\chi = \zeta^{-1}\eta p w^{-1} + \eta(d^{-1} - \eta)\operatorname{tr}(\hat{r})w^{-1} - \eta P(w^{-1})r \tag{3.113f}$$
$$= -\eta(\tau w^{-1} + P(w^{-1})r). \tag{3.113g}$$

Note (3.113d) follows from (3.101), and (3.113f) follows from (3.25).

The gradient of Γ in (3.104) is:

$$g_u = \zeta^{-1}, \tag{3.114a}$$
$$g_w = -\zeta^{-1}\nabla\varphi(w) - w^{-1} \tag{3.114b}$$
$$= -\theta w^{-1}. \tag{3.114c}$$

Differentiating (3.114), the Hessian product is (using (3.25)):

$$H_u = \nabla_u g_u p + \nabla_u g_w[r] \tag{3.115a}$$
$$= \zeta^{-2}p - \zeta^{-1}\eta\operatorname{tr}(\hat{r}) \tag{3.115b}$$
$$= -\zeta^{-1}\chi, \tag{3.115c}$$
$$H_w = \nabla_w g_u p + \nabla_w g_w[r] \tag{3.115d}$$
$$= -\zeta^{-1}\eta p w^{-1} - \operatorname{tr}(\hat{r})\eta(d^{-1} - \eta)w^{-1} + \theta P(w^{-1})r \tag{3.115e}$$
$$= P(w^{-1/2})(\eta\tau e + \theta\hat{r}). \tag{3.115f}$$

Differentiating (3.115), the the third order directional derivative is (using (3.26)):

$$T_u = \nabla_u H_u p + \nabla_u H_w[r] \tag{3.116a}$$
$$= -2\zeta^{-2}p\chi + \zeta^{-1}\eta(\tau\operatorname{tr}(\hat{r}) + \operatorname{tr}(\hat{r}^2)) + \zeta^{-1}\eta\operatorname{tr}(\hat{r})\chi \tag{3.116b}$$
$$= \zeta^{-1}(2\chi^2 + \eta\upsilon), \tag{3.116c}$$
$$T_w = \nabla_w H_u p + \nabla_w H_w[r] \tag{3.116d}$$
$$\begin{aligned} &= \zeta^{-1}\eta p\chi w^{-1} + \zeta^{-1}\eta p(\tau w^{-1} + P(w^{-1})r) - \eta\tau P(w^{-1})r + \\ &\quad \operatorname{tr}(\hat{r})\tau\eta(d^{-1} - \eta)w^{-1} + \eta(-\eta(\tau\operatorname{tr}(\hat{r}) + \operatorname{tr}(\hat{r}^2)) + d^{-1}\operatorname{tr}(\hat{r}^2))w^{-1} + \\ &\quad \eta(d^{-1} - \eta)\operatorname{tr}(\hat{r})P(w^{-1})r - 2\theta P(w^{-1/2})\hat{r}^2 \end{aligned} \tag{3.116e}$$
$$\begin{aligned} &= \eta(-\chi\tau + \zeta^{-1}p\chi + (d^{-1} - \eta)(\operatorname{tr}(\hat{r})\tau + \operatorname{tr}(\hat{r}^2)))w^{-1} + \\ &\quad \eta(-\tau + \zeta^{-1}p + (d^{-1} - \eta)\operatorname{tr}(\hat{r}))P(w^{-1})r - 2\theta P(w^{-1/2})\hat{r}^2 \end{aligned} \tag{3.116f}$$

$$= P(w^{-1/2})(\eta(-2\chi\tau + (d^{-1} - \eta)\upsilon)e - 2\eta\tau\hat{r} - 2\theta\hat{r}^2). \tag{3.116g}$$

In Lemma 3.7.1 below, we give a closed form inverse Hessian product operator. This operator (3.117) a similar structure to the Hessian product operator (3.115), except that it applies $P(w^{1/2})$ instead of $P(w^{-1/2})$.

Lemma 3.7.1. Letting $\check{r} \coloneqq P(w^{1/2})r \in V$, the inverse Hessian product is:

$$\bar{H}_u = (\zeta^2 + d^{-1}\varphi^2)p + d^{-1}\varphi \operatorname{tr}(\check{r}), \tag{3.117a}$$
$$\bar{H}_w = P(w^{1/2})(d^{-1}(\varphi p + \eta\theta^{-1}\operatorname{tr}(\check{r}))e + \theta^{-1}\check{r}). \tag{3.117b}$$

Proof. Note that the Hessian operator (3.115) is a positive definite linear operator, so it has a unique inverse linear operator. We show that $(\nabla^2\Gamma)^{-1}(\nabla^2\Gamma[\tilde{p}]) = \tilde{p}$. Into (3.117), we substitute the values from (3.115) i.e. $p = H_u = -\zeta^{-1}\chi$ and $r = H_w = P(w^{-1/2})(\eta\tau e + \theta\hat{r})$. Since $P(w^{1/2}) = P(w^{-1/2})^{-1}$, we have:

$$\check{r} = P(w^{1/2})H_w = \eta\tau e + \theta\hat{r}, \tag{3.118a}$$
$$\operatorname{tr}(\check{r}) = d\eta\tau + \theta\operatorname{tr}(\hat{r}). \tag{3.118b}$$

We have:

$$\bar{H}_u = (\zeta^2 + d^{-1}\varphi^2)(-\zeta^{-1}\chi) + d^{-1}\varphi(d\eta\tau + \theta\operatorname{tr}(\hat{r})) \tag{3.119a}$$
$$= -\zeta\chi + \varphi\eta(\tau - \chi) + d^{-1}\varphi\theta\operatorname{tr}(\hat{r}) \tag{3.119b}$$
$$= -\zeta(\chi - \eta\operatorname{tr}(\hat{r})) \tag{3.119c}$$
$$= p, \tag{3.119d}$$

and:

$$\bar{H}_w = P(w^{1/2})(d^{-1}(-\varphi\zeta^{-1}\chi + \eta\theta^{-1}(d\eta\tau + \theta\operatorname{tr}(\hat{r})))e + \theta^{-1}(\eta\tau e + \theta\hat{r})) \tag{3.120a}$$
$$= P(w^{1/2})(\eta(-\tau + \eta\theta^{-1}\tau + \theta^{-1}\tau)e + \hat{r}) \tag{3.120b}$$
$$= P(w^{1/2})(\hat{r}) \tag{3.120c}$$
$$= r. \tag{3.120d}$$

Hence (3.117) is the unique inverse operator of (3.115). □

We note the polynomial-like structure of the oracles. In particular, the w components of the g, H, and T oracles are computed by applying $P(w^{-1/2})$ to a polynomial in $\hat{r}$, of degree zero for g, degree one for H, and degree two for T. Analogously to H, its inverse $\bar{H}$ is computed by applying $P(w^{1/2})$ to a polynomial of degree one in $\check{r}$. This structure leads to simple, efficient, and numerically-stable implementations. We also note the structural similarity (ignoring constants) between the u and w components of these oracles and those of the negative log-determinant barrier in Section 3.5.4.

In both cases, the oracles can be computed without an explicit eigendecomposition if it is possible to apply $P(w^{1/2})$ and $P(w^{-1/2})$ directly. For example for $V = \mathbb{S}^d$ and $V = \mathbb{H}^d$, only a Cholesky factorization of w is needed.

3.8 Examples and computational testing

We outline our implementations of the MMD cone and the log-determinant and root-determinant cones in Hypatia in Section 3.8.1. In Sections 3.8.4.1 to 3.8.4.4, we present example problems with simple, natural formulations (NFs) in terms of these cones. Using techniques we describe in Section 3.8.2, we construct equivalent extended formulations (EFs) that can be recognized by MOSEK 9 or ECOS. Our computational benchmarks follow the methodology in Section 3.8.3 and show that Hypatia typically solves the NFs much more efficiently than Hypatia, MOSEK, or ECOS solve the EFs. Finally, in Section 3.8.5, we exemplify the computational impact of efficient oracle procedures by comparing the performance of our closed form inverse Hessian product formula in (3.69) with that of a naive direct solve using the explicit Hessian matrix.

3.8.1 Hypatia solver

Recall Hypatia's primal general conic form, described in Section 1.4. Hypatia's generic cone interface allows specifying a vectorized proper cone $\mathcal{K} \subset \mathbb{R}^q$ for some dimension q. For the real symmetric domain $\mathbb{S}^d$ and the complex Hermitian domain $\mathbb{H}^d$, we use the standard svec transformations to real vector space described in Section 0.3. These transformations preserve inner products and the self-duality of the cones of squares $\mathbb{S}^d_\succeq$ and $\mathbb{H}^d_\succeq$. We adapt these transformations to enable vectorization of spectral function cones. For example, for the epigraph-perspective cone $\mathcal{K}_p$ in (3.31), the vectorization is $(u, v, \text{vec}(w)) \in \mathbb{R}^{2+q}$, where $\text{vec}(w) \in \mathbb{R}^q$ is the appropriate vectorization of $w \in \mathcal{Q}$. Fortunately, the dual cone of this vectorized cone is the analogous vectorization of the dual cone $\mathcal{K}_p^*$ in (3.34).

For the domains $\mathbb{R}^d$, $\mathbb{S}^d$, and $\mathbb{H}^d$, we implement the MMD cone $\mathcal{K}_{\text{MMD}}$, the log-determinant cone $\mathcal{K}_{\text{logdet}}$, and the root-determinant cone $\mathcal{K}_{\text{rtdet}}$ in Hypatia.[1] This allows the user to model with these cones or their dual cones. As we discuss at the end of Sections 3.5.4 and 3.7.4, for $\mathcal{K}_{\text{logdet}}$ and $\mathcal{K}_{\text{rtdet}}$ the oracle procedures are quite specialized, for example we compute a Cholesky factorization rather than an eigendecomposition for the $\mathbb{S}^d$ and $\mathbb{H}^d$ domains.

For $\mathcal{K}_{\text{MMD}}$, we predefine the MMD functions in Table 3.1 (e.g. *NegEntropy*). Recall that $\mathcal{K}^*_{\text{MMD}}$ in (3.83) is defined using the convex conjugate of the MMD function; in the examples below we suffix the MMD function names with *Conj* (e.g. *NegEntropyConj*) to indicate use of the convex conjugate function and $\mathcal{K}^*_{\text{MMD}}$. We write *NegLogdet* or *NegRtdet* for the negative log-determinant or negative root-determinant function, the epigraph of which we represent using $\mathcal{K}_{\text{logdet}}$ or $\mathcal{K}_{\text{rtdet}}$.

[1] Our $\mathcal{K}_{\text{logdet}}$ implementation is for the hypograph form introduced in Section 2.2.3.2 rather than the epigraph form in Section 3.5.4. This only requires minor changes to the oracle derivations and the LHSCB proof from Section 3.6.4 for validity.

In our examples, we choose not to use $\mathcal{K}^*_{\text{rtdet}}$ or $\mathcal{K}^*_{\text{logdet}}$ (or equivalently, $\mathcal{K}^*_{\text{MMD}}$ with *NegLogConj*), because these particular dual cones provide little additional modeling power over their primal cones.

3.8.2 Natural and extended formulations

To assess the computational value of our new cones and efficient oracles, we compare the performance of Hypatia on NFs over $\mathcal{K}_{\text{MMD}}$, $\mathcal{K}_{\text{logdet}}$, and $\mathcal{K}_{\text{rtdet}}$ against that of other conic IPM solvers on equivalent EFs. ECOS (Domahidi, Chu, and Boyd, 2013) is another open source conic IPM solver, but it only supports nonnegative, second order, and three-dimensional exponential cones. MOSEK version 9 (MOSEK ApS, 2020a) is a commercial conic IPM solver that supports the same cones as ECOS as well as three-dimensional power cones and real symmetric PSD cones. As we discuss in Section 2.2, we call the cones supported by MOSEK 9 the standard cones. To build the standard cone EFs, we use a variety of formulation techniques, some of which we discuss and analyze in Section 2.2.3.

For $V = \mathbb{R}^d$, we use EFs of the form (2.24). The EFs for *NegLog*, *NegEntropy*, *NegEntropyConj*, and *NegRtdet* use d exponential cones. The EFs for *NegSqrt* and *NegSqrtConj* use d three-dimensional second order cones. The EFs for *Power*, *NegPower*, *PowerConj*, and *NegPowerConj* use d power cones. The example in Section 3.8.4.1 uses $V = \mathbb{R}^d$.

For $V = \mathbb{S}^d$, we use EFs of the form (2.25). This is a large formulation with many additional variables and PSD constraints. For *NegLog* and *NegRtdet*, we use the simpler EFs in (2.18) and (2.22). Since *NegSqrtConj* is a scaling of the inverse function (see Table 3.1), a Schur complement representation allows us to use an EF with one PSD cone constraint. For $V = \mathbb{H}^d$, we generalize the EF (2.25) for Hermitian matrices and apply the Hermitian PSD cone EF in Section 2.2.1.1. The examples in Sections 3.8.4.2 and 3.8.4.3 use $V = \mathbb{S}^d$ and the example in Section 3.8.4.4 uses $V = \mathbb{H}^d$. The formulation dimensions and barrier parameters associated with these EFs are given in Table 2.1.

3.8.3 Computational methodology

Our computational experiments are similar to those of Section 2.3. We perform instance generation, computational experiments, and results analysis on Ubuntu 21.10 with Julia 1.8.0 and Hypatia 0.5.3.[2] We use dedicated hardware with an AMD Ryzen 9 3950X 16-core processor (32 threads) and 128GB of RAM. For each example problem in Sections 3.8.4.1 to 3.8.4.4, we generate random instances of a range of sizes, using JuMP 0.21.10 and MathOptInterface v0.9.22. All instances are primal-dual feasible, so we expect solvers to return optimality certificates.

We use the conic IPM solvers in MOSEK version 9 and ECOS version 2.0.5 (with no features disabled). Hypatia uses a particular default algorithmic implementation that we describe in Section 1.6 (the combined directions method with the QR-Cholesky linear system procedure). We limit each

[2] Benchmark scripts and instructions for reproducing and analyzing results are available at https://github.com/chriscoey/Hypatia.jl/tree/master/benchmarks/natvsext. A raw output CSV file and detailed results tables are at https://github.com/chriscoey/Hypatia.jl/wiki.

solver to 16 threads and set a solve time limit of 1800 seconds. We set relative feasibility and optimality gap tolerances to 10^{-7} and absolute optimality gap tolerances to 10^{-10}.

For each instance, the relative difference between the objective values of the formulation/solver combinations that converge never exceeds 10^{-4}. For each instance/formulation/solver combination that returns a solution, we measure the maximum violation ϵ of the primal-dual optimality conditions in (2.26). In Figures 3.2 to 3.5, we plot solve times in seconds against an instance size parameter, excluding solves for which $\epsilon > 10^{-5}$. Hyp-NF (i.e. Hypatia solving the NF) is faster than any EF solver (Hyp-EF, MO-EF, ECOS-EF) across all instance sizes and spectral functions tested for each example, and always scales to larger sizes.

3.8.4 Examples and results

3.8.4.1 Nonparametric distribution estimation

Suppose we have a random variable X taking values in the finite set $\{\alpha_i\}_{i\in[\![d]\!]}$. We seek a probability distribution $\rho \in \mathbb{R}^d$ that minimizes a convex spectral function φ, given some prior information expressed with $d/2$ linear equality constraints. Adapting Boyd and Vandenberghe (2004, Section 7.2), the problem is:

$$\min_{\rho\in\mathbb{R}^d} \quad \varphi(\rho) : \tag{3.121a}$$
$$\operatorname{tr}(\rho) = d, \tag{3.121b}$$
$$A\rho = b. \tag{3.121c}$$

For four spectral functions φ on $\mathbb{R}^d_\geq$ (with EFs that ECOS can recognize) and a range of sizes d, we build random instances of (3.121). The results are summarized in Table 3.2 and Figure 3.2. Note that for *NegRtdet*, no solve times are plotted for MO-EF because the optimality condition violations ϵ are too large (see Section 3.8.3); tightening MOSEK's tolerance options improves these violations, though in either case MO-EF is significantly slower than Hyp-NF. We do not plot results for *NegLogdet* (the $\mathcal{K}_{\text{logdet}}$ formulation using the specialized oracles from Section 3.5.4) as they are nearly identical to the results for $\mathcal{K}_{\text{MMD}}$/*NegLog*; however, the efficiency benefits of *NegLogdet* are realized for the matrix domain in Section 3.8.4.2.

3.8.4.2 Experiment design

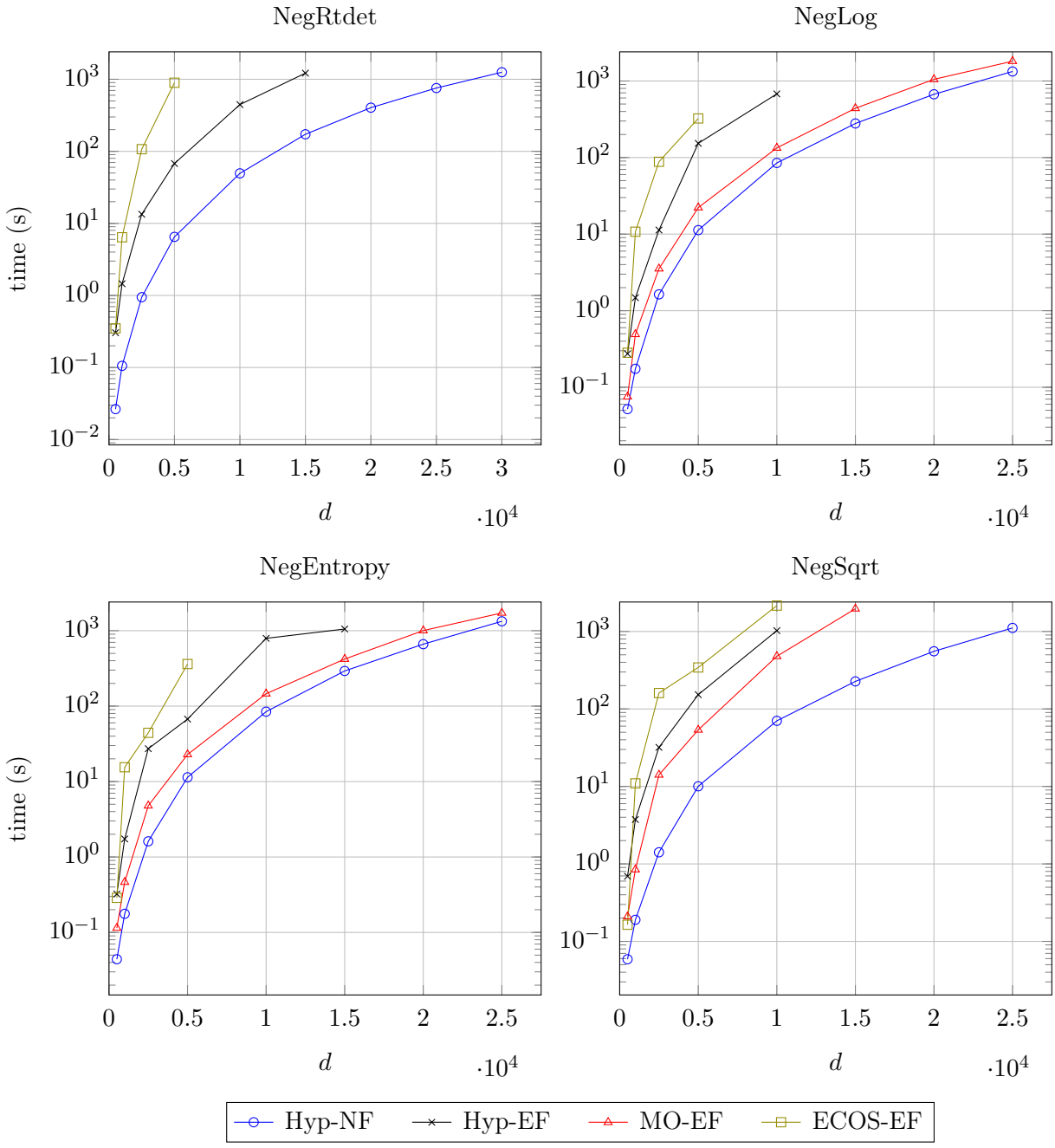
We formulate a continuous relaxation of the experiment design problem, similar to Boyd and Vandenberghe (2004, Section 7.5). The variable $\rho \in \mathbb{R}^{2d}$ is the number of trials to run for each of $2d$ experiments that are useful for estimating a vector in $\mathbb{R}^d$. The experiments are described by the columns of $V \in \mathbb{R}^{d\times 2d}$ and we require that $2d$ experiments are performed. We minimize a convex spectral function of the information matrix:

$$\min_{\rho\in\mathbb{R}^{2d}} \quad \varphi(V \operatorname{Diag}(\rho) V') : \tag{3.122a}$$
$$\operatorname{tr}(\rho) = 2d, \tag{3.122b}$$

Table 3.2: Nonparametric distribution estimation solver statistics.

		Hyp-NF			Hyp-EF			MO-EF			ECOS-EF		
cone	d	st	it	time	st	it	time	st	it	timest	it	time	
NegRtdet	500	co	11	0.0	co	14	0.3	co	8	0.2	co	22	0.3
	1000	co	11	0.1	co	13	1.5	co	8	0.7	co	23	6.4
	2500	co	12	0.9	co	15	13	co	8	9.0	co	23	107
	5000	co	12	6.5	co	14	68	co	8	67	co	24	895
	10000	co	10	49	co	16	446	co	8	496	tl	*	*
	15000	co	14	172	co	14	1216	co	8	1589	sk	*	*
	20000	co	15	404	tl	7	1863	tl	*	*	sk	*	*
	25000	co	14	756	sk	*	*	sk	*	*	sk	*	*
	30000	co	13	1252	sk	*	*	sk	*	*	sk	*	*
NegLog	500	co	17	0.1	co	11	0.3	co	8	0.1	co	18	0.3
	1000	co	17	0.2	co	13	1.5	co	16	0.5	co	17	11
	2500	co	25	1.6	co	12	11	co	8	3.5	co	18	88
	5000	co	29	11	co	13	153	co	11	22	co	18	324
	10000	co	36	85	co	12	678	co	8	134	tl	*	*
	15000	co	40	279	tl	*	*	co	12	440	sk	*	*
	20000	co	46	670	sk	*	*	co	18	1051	sk	*	*
	25000	co	50	1331	sk	*	*	tl	10	1812	sk	*	*
	30000	tl	35	1817	sk	*	*	sk	*	*	sk	*	*
NegEntropy	500	co	14	0.0	co	12	0.3	co	17	0.1	co	19	0.3
	1000	co	17	0.2	co	12	1.7	co	13	0.5	co	18	15
	2500	co	24	1.6	co	13	27	co	19	4.8	co	18	44
	5000	co	28	11	co	12	67	co	11	23	co	19	361
	10000	co	35	84	co	12	792	co	12	146	tl	7	2157
	15000	co	43	293	co	11	1053	co	8	420	sk	*	*
	20000	co	45	663	tl	7	1861	co	15	1003	sk	*	*
	25000	co	50	1332	sk	*	*	co	8	1719	sk	*	*
	30000	tl	35	1824	sk	*	*	tl	*	*	sk	*	*
NegSqrt	500	co	20	0.1	co	10	0.7	co	6	0.2	co	9	0.2
	1000	co	19	0.2	co	10	3.7	co	7	0.8	co	8	11
	2500	co	20	1.4	co	11	32	co	8	14	co	8	161
	5000	co	22	10	co	9	153	co	5	54	co	8	343
	10000	co	25	70	co	11	1026	co	7	478	tl	6	2157
	15000	co	27	227	rl	*	*	tl	6	1966	sk	*	*
	20000	co	32	555	sk	*	*	sk	*	*	sk	*	*
	25000	co	36	1111	sk	*	*	sk	*	*	sk	*	*
	30000	tl	35	1821	sk	*	*	sk	*	*	sk	*	*

Figure 3.2: Nonparametric distribution estimation solver performance.



$$\rho \geq 0. \tag{3.122c}$$

For four different φ on $\mathbb{S}^d_+$ and various d, we build random instances of (3.122). The results are summarized in Tables 3.3 and 3.4 and Figure 3.3. Since ECOS does not support $\mathbb{S}^d_+$, we only compare with MOSEK. The *Hyp-NegLogdet* curve indicates that Hypatia with $\mathcal{K}_{\text{logdet}}$ is somewhat more efficient than Hypatia with the equivalent $\mathcal{K}_{\text{MMD}}$/*NegLog* formulation; this is due to our oracle specializations in Section 3.5.4 and our implementation using a Cholesky factorization rather than an eigendecomposition.

Table 3.3: Experiment design solver statistics.

		Hyp-NF			Hyp-EF			MO-EF		
cone	d	st	it	time	st	it	time	st	it	time
NegRtdet	25	co	20	0.0	co	14	0.2	co	18	0.7
	50	co	24	0.2	co	15	2.7	co	11	8.9
	75	co	18	0.3	co	15	13	co	10	49
	100	co	20	0.6	co	17	51	co	10	190
	150	co	19	1.6	co	17	357	co	9	1178
	200	sp	9	2.3	co	19	1596	rl	$*$	$*$
	300	co	19	16	rl	$*$	$*$	sk	$*$	$*$
	400	co	19	37	sk	$*$	$*$	sk	$*$	$*$
	500	co	23	95	sk	$*$	$*$	sk	$*$	$*$
	600	co	22	159	sk	$*$	$*$	sk	$*$	$*$
	700	co	19	233	sk	$*$	$*$	sk	$*$	$*$
	800	co	20	417	sk	$*$	$*$	sk	$*$	$*$
	900	co	18	554	sk	$*$	$*$	sk	$*$	$*$
NegLog	25	co	23	0.2	co	12	0.2	co	20	0.7
	50	co	24	0.5	co	13	2.4	co	11	9.0
	75	co	23	1.3	co	13	13	co	11	54
	100	co	22	1.7	co	13	40	co	10	201
	150	co	25	5.4	co	15	319	co	10	1305
	200	co	21	10	co	14	1199	tl	0	1853
	300	co	29	49	rl	$*$	$*$	sk	$*$	$*$
	400	co	31	120	sk	$*$	$*$	sk	$*$	$*$
	500	co	39	297	sk	$*$	$*$	sk	$*$	$*$
	600	co	41	582	sk	$*$	$*$	sk	$*$	$*$
	700	co	41	902	sk	$*$	$*$	sk	$*$	$*$
	800	co	44	1528	sk	$*$	$*$	sk	$*$	$*$
	900	tl	35	1801	sk	$*$	$*$	sk	$*$	$*$

3.8.4.3 Central polynomial Gram matrix

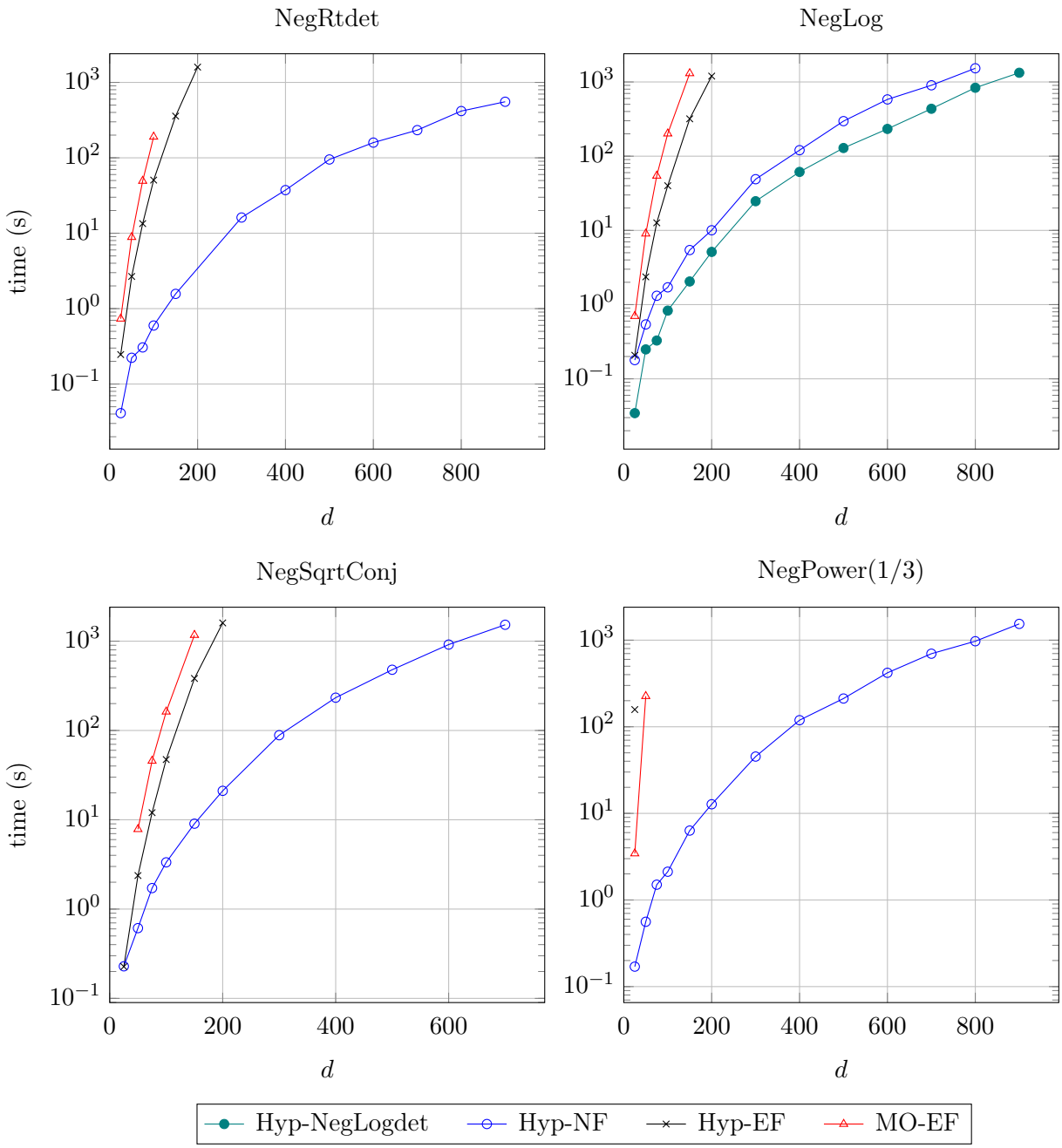
Suppose we have a polynomial of degree $2k$ in m variables. Let $L = \binom{m+k}{m}$ and $U = \binom{m+2k}{m}$, and let $b \in \mathbb{R}^U$ be the monomial coefficients of the polynomial. We seek a Gram matrix $\rho \in \mathbb{S}^L$ corresponding to b (Parrilo, 2012, Lemma 3.33) that minimizes a convex spectral function φ:

$$\min_{\rho \in \mathbb{S}^L} \quad \varphi(\rho) : \tag{3.123a}$$

Table 3.4: Experiment design solver statistics.

		Hyp-NF			Hyp-EF			MO-EF		
cone	d	st	it	time	st	it	time	st	it	time
NegSqrtConj	25	co	26	0.2	co	14	0.2	sp	14	0.6
	50	co	28	0.6	co	14	2.4	co	8	7.8
	75	co	32	1.7	co	15	12	co	8	46
	100	co	36	3.3	co	16	47	co	7	162
	150	co	46	9.0	co	19	382	co	8	1170
	200	co	48	21	co	19	1601	rl	*	*
	300	co	57	89	rl	*	*	sk	*	*
	400	co	66	233	sk	*	*	sk	*	*
	500	co	69	479	sk	*	*	sk	*	*
	600	co	73	916	sk	*	*	sk	*	*
	700	co	75	1527	sk	*	*	sk	*	*
	800	tl	59	1811	sk	*	*	sk	*	*
NegPower(1/3)	25	co	19	0.2	sp	30	158	co	16	3.4
	50	co	26	0.6	rl	*	*	sp	42	225
	75	co	27	1.5	sk	*	*	tl	41	1817
	100	co	24	2.1	sk	*	*	sk	*	*
	150	co	29	6.3	sk	*	*	sk	*	*
	200	co	27	13	sk	*	*	sk	*	*
	300	co	27	45	sk	*	*	sk	*	*
	400	co	30	119	sk	*	*	sk	*	*
	500	co	27	212	sk	*	*	sk	*	*
	600	co	29	420	sk	*	*	sk	*	*
	700	co	30	697	sk	*	*	sk	*	*
	800	co	28	972	sk	*	*	sk	*	*
	900	co	30	1541	sk	*	*	sk	*	*

Figure 3.3: Experiment design solver performance.



$$C \operatorname{vec}(\rho) = b, \tag{3.123b}$$

where the matrix $C \in \mathbb{R}^{U \times L(L+1)/2}$ maps the Gram matrix to the (lower-dimensional) polynomial coefficient space. We build random instances of (3.123), varying $m \in \{1, 4\}$ and k (depending on m). Recall from Table 3.1 that *ConjNegEntr* and *ConjPower-1.5* are defined on $\mathbb{S}^d$, but *NegEntr* and *MatPower12(1.5)* are only defined on $\mathbb{S}^d_+$, which implicitly requires that b be a sum-of-squares polynomial and hence globally nonnegative. The results are summarized in Tables 3.5 and 3.6 and Figure 3.4 (a log-log plot).

Table 3.5: Central polynomial Gram matrix solver statistics.

			Hyp-NF			Hyp-EF			MO-EF		
cone	m	k	st	it	time	st	it	time	st	it	time
NegEntropy	1	15	co	14	0.1	co	19	7.5	co	11	0.4
	1	25	co	16	0.6	co	29	200	co	17	5.7
	1	50	co	24	5.5	rl	*	*	co	25	220
	1	75	co	23	23	sk	*	*	tl	23	1845
	1	100	co	27	72	sk	*	*	sk	*	*
	1	125	co	29	167	sk	*	*	sk	*	*
	1	150	co	33	426	sk	*	*	sk	*	*
	1	175	co	32	826	sk	*	*	sk	*	*
	1	200	tl	37	1807	sk	*	*	sk	*	*
	4	2	co	12	0.1	co	14	4.0	co	11	0.3
	4	3	co	17	1.1	tl	19	1885	co	22	27
	4	4	co	23	15	sk	*	*	co	28	1423
	4	5	co	28	140	sk	*	*	m	*	*
	4	6	tl	33	1830	sk	*	*	sk	*	*
NegEntropyConj	1	15	co	19	0.2	co	44	16	co	27	1.1
	1	25	co	22	0.8	co	53	376	co	25	8.9
	1	50	co	27	6.3	rl	*	*	co	34	316
	1	75	co	28	28	sk	*	*	tl	20	1822
	1	100	co	32	84	sk	*	*	sk	*	*
	1	125	co	33	188	sk	*	*	sk	*	*
	1	150	co	34	423	sk	*	*	sk	*	*
	1	175	co	35	884	sk	*	*	sk	*	*
	1	200	co	36	1729	sk	*	*	sk	*	*
	4	2	co	17	0.1	co	24	6.6	co	14	0.4
	4	3	co	24	1.5	tl	18	1844	co	22	27
	4	4	co	31	20	sk	*	*	co	31	1477
	4	5	co	38	190	sk	*	*	m	*	*
	4	6	tl	34	1820	sk	*	*	sk	*	*

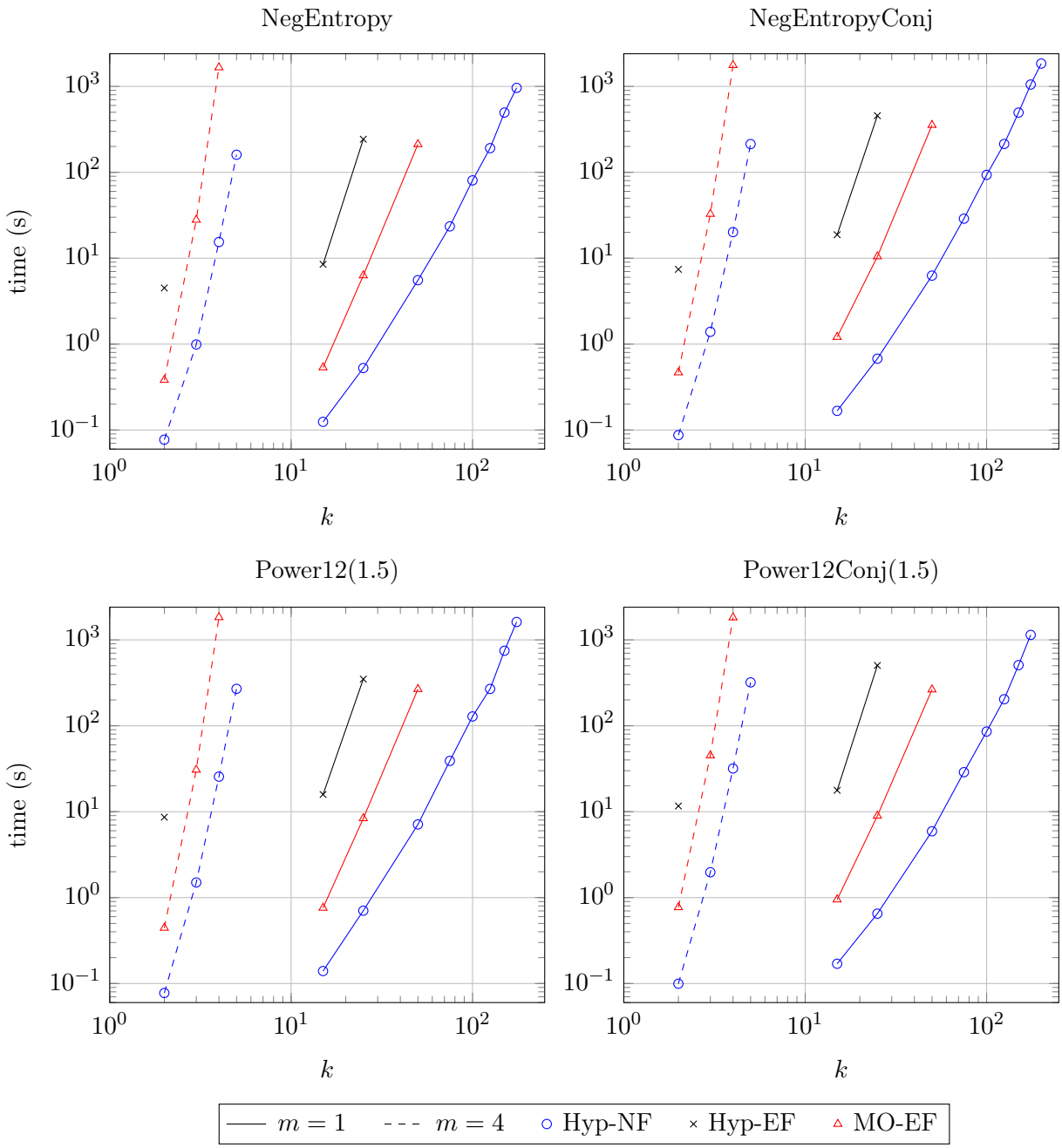
3.8.4.4 Classical-quantum channel capacity

We compute the capacity of a classical-quantum channel, adapting the formulation from Sutter et al. (2015, Example 2.16) and H. Fawzi and O. Fawzi (2018, Section 3.1). The variable $\rho \in \mathbb{R}^d$ is a probability distribution on the d-dimensional input alphabet. For $i \in [\![d]\!]$, let $P_i \in \mathbb{H}^d_+$ be fixed

Table 3.6: Central polynomial Gram matrix solver statistics.

			Hyp-NF			Hyp-EF			MO-EF		
cone	m	k	st	it	time	st	it	time	st	it	time
Power12(1.5)	1	15	co	16	0.2	er	34	14	co	15	0.6
	1	25	co	22	0.8	er	41	291	co	19	6.7
	1	50	co	31	7.0	rl	*	*	co	24	230
	1	75	co	38	38	sk	*	*	tl	22	1834
	1	100	co	44	117	sk	*	*	sk	*	*
	1	125	co	42	243	sk	*	*	sk	*	*
	1	150	co	50	649	sk	*	*	sk	*	*
	1	175	co	54	1400	sk	*	*	sk	*	*
	1	200	tl	37	1822	sk	*	*	sk	*	*
	4	2	co	16	0.1	sp	28	7.7	co	13	0.4
	4	3	co	26	1.7	tl	18	1818	co	18	24
	4	4	co	39	25	sk	*	*	tl	37	1820
	4	5	co	47	235	sk	*	*	sk	*	*
	4	6	tl	33	1837	sk	*	*	sk	*	*
Power12Conj(1.5)	1	15	co	21	0.2	co	43	16	co	20	0.8
	1	25	co	21	0.7	co	60	412	co	22	7.7
	1	50	co	26	5.9	rl	*	*	co	27	254
	1	75	co	28	27	sk	*	*	tl	20	1826
	1	100	co	30	77	sk	*	*	sk	*	*
	1	125	co	32	178	sk	*	*	sk	*	*
	1	150	co	35	430	sk	*	*	sk	*	*
	1	175	co	39	972	sk	*	*	sk	*	*
	1	200	tl	38	1821	sk	*	*	sk	*	*
	4	2	co	21	0.1	co	37	10	co	19	0.6
	4	3	co	35	2.1	tl	18	1828	co	25	33
	4	4	co	49	31	sk	*	*	co	32	1598
	4	5	co	58	286	sk	*	*	m	*	*
	4	6	tl	34	1829	sk	*	*	sk	*	*

Figure 3.4: Central polynomial Gram matrix solver performance.



density matrices satisfying $\operatorname{tr}(P_i) = 1$. Letting φ represent the trace of *NegEntropy* on $\mathbb{H}^d_{\succeq}$, the formulation is:

$$\min_{\rho \in \mathbb{R}^d} \quad \varphi\big(\textstyle\sum_{i \in [\![d]\!]} \rho_i P_i\big) - \sum_{i \in [\![d]\!]} \rho_i \varphi(P_i) : \tag{3.124a}$$

$$\operatorname{tr}(\rho) = 1, \tag{3.124b}$$

$$\rho \geq 0. \tag{3.124c}$$

We generate random instances of (3.124), varying d. The results are summarized in Table 3.7 and Figure 3.5.

Table 3.7: Classical-quantum channel capacity solver statistics.

	Hyp-NF			Hyp-EF			MO-EF		
d	st	it	time	st	it	time	st	it	time
10	co	17	0.0	co	22	2.6	co	14	0.5
20	co	21	0.1	co	44	688	co	19	14
30	co	22	0.2	tl	8	1841	co	30	179
40	co	24	0.4	sk	*	*	co	29	753
50	co	24	0.6	sk	*	*	tl	20	1805
75	co	33	1.8	sk	*	*	sk	*	*
100	co	34	3.7	sk	*	*	sk	*	*
150	co	40	13	sk	*	*	sk	*	*
200	co	43	30	sk	*	*	sk	*	*
250	co	42	57	sk	*	*	sk	*	*
300	co	47	111	sk	*	*	sk	*	*
350	co	47	167	sk	*	*	sk	*	*
400	co	49	280	sk	*	*	sk	*	*
450	co	53	420	sk	*	*	sk	*	*
500	co	49	533	sk	*	*	sk	*	*
550	co	53	740	sk	*	*	sk	*	*
600	co	58	1105	sk	*	*	sk	*	*
650	co	58	1463	sk	*	*	sk	*	*
700	co	59	1748	sk	*	*	sk	*	*
750	tl	49	1805	sk	*	*	sk	*	*

3.8.5 Inverse Hessian product oracle

To illustrate the importance of our efficient and numerically stable oracle procedures, we compare two different approaches to computing the inverse Hessian product oracle $\bar{H}$ in (3.42c) for $\mathcal{K}_{\text{MMD}}$ cones (similar to our analysis in Section 2.6.4 for the spectral norm cone). The naive approach is to compute the explicit Hessian matrix, perform a Cholesky factorization, and use a direct linear solve. Alternatively, we apply the analytic formula for $\bar{H}$ in (3.69), since $\mathcal{K}_{\text{MMD}}$ is a special case of $\mathcal{K}_p$ with a separable spectral function. This formula is essentially as easy to compute as the Hessian product oracle H in (3.51) (which does not use an explicit Hessian matrix). In Table 3.8, we compare the worst-case memory and time complexities for these procedures.

Figure 3.5: Classical-quantum channel capacity solver performance.

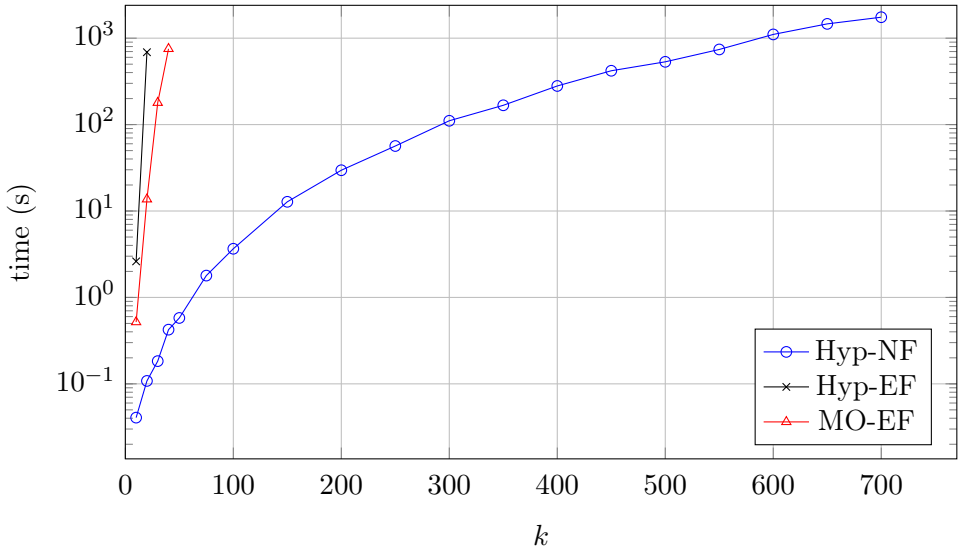


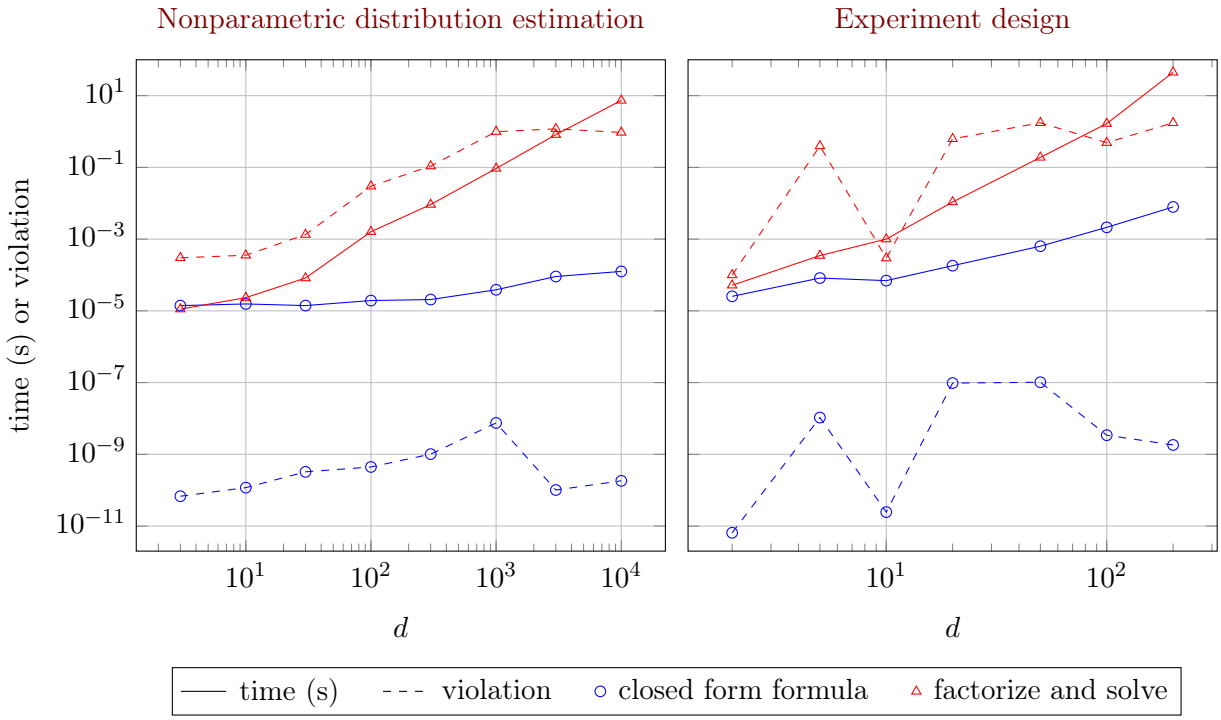
Table 3.8: Cone dimension and worst-case complexities for the two inverse Hessian product procedures.

		closed form formula		factorize and solve	
V	$\dim(\mathcal{K}_{\text{MMD}})$	memory	time	memory	time
$\mathbb{R}^d$	$\mathcal{O}(d)$	$\mathcal{O}(d)$	$\mathcal{O}(d)$	$\mathcal{O}(d^2)$	$\mathcal{O}(d^3)$
$\mathbb{S}^d$ or $\mathbb{H}^d$	$\mathcal{O}(d^2)$	$\mathcal{O}(d^2)$	$\mathcal{O}(d^3)$	$\mathcal{O}(d^4)$	$\mathcal{O}(d^6)$

To compare the practical performance of these procedures, we first use Hypatia to solve NF instances of a range of sizes for the examples from Section 3.8.4.1 (with $V = \mathbb{R}^d$) and Section 3.8.4.2 (with $V = \mathbb{S}^d$), using $\mathcal{K}_{\text{MMD}}$ with the *NegEntropy* function. For each instance, at Hypatia's final IPM iterate, we take the direction $\tilde{p} = g$ (i.e. the gradient oracle in (3.42a) at the iterate) and compute $\bar{H}$ for this direction using both procedures. To measure the numerical accuracy of each procedure, we compute the violation $\epsilon := |1 - \nu^{-1}\langle \bar{H}, g\rangle|$ on a logarithmic homogeneity condition for the LHSCB Γ with parameter $\nu = 2 + d$. We also time each procedure, excluding Hessian memory allocation time for the factorization-based procedure.

Our results are displayed in Figure 3.6. The Cholesky factorization fails at $d = 3000$ for the $\mathbb{R}^d$ example and at $d = 20, 50, 200$ for the $\mathbb{S}^d$ example; when this occurs, Hypatia uses a Bunch-Kaufman factorization as a fallback. Note that we loosen the convergence tolerances specified in Section 3.8.3 by a factor of 100, so that the factorization-based procedure fails less often. These comparisons demonstrate that our closed form formula generally allows computing $\bar{H}$ faster and with greater numerical accuracy.

Figure 3.6: For instances of two examples using $\mathcal{K}_{\text{MMD}}$ with *NegEntropy*, the speed and logarithmic homogeneity condition violation (at the final iterate) for the two inverse Hessian product procedures.



Chapter 4

Outer approximation with conic certificates

Abstract

We present the first conic-duality-based branch-and-bound outer approximation algorithm for mixed-integer conic problems. The polyhedral relaxations are refined with $\mathcal{K}^*$ *cuts* derived from conic certificates for continuous conic subproblems. Under the assumption that all subproblems are well-posed, the algorithm detects infeasibility or unboundedness or returns an optimal solution in finite time. Using properties of the conic certificates, we show that the $\mathcal{K}^*$ cuts imply certain practically-relevant guarantees about the quality of the polyhedral relaxations, and we show how to maintain helpful guarantees when the LP solver uses a positive feasibility tolerance. We discuss techniques for tightening the polyhedral relaxations such as cut disaggregation. Our open source MI-conic solver *Pajarito* uses external mixed-integer linear solvers and continuous conic solvers. Benchmarking on mixed-integer second order cone problems, we find that Pajarito greatly outperforms Bonmin solver and is competitive with CPLEX's specialized algorithm. On new MI-conic examples involving standard second order, exponential, and positive semidefinite cones, we demonstrate the computational benefits of our $\mathcal{K}^*$ cut techniques.

4.1 Introduction

4.1.1 Branch-and-bound algorithms for mixed-integer convex optimization

A mixed-integer convex (MI-convex) problem minimizes a convex objective function over convex constraints and integrality restrictions on a subset of the variables. Belotti et al. (2013) and Bonami, Kılınç, and Linderoth (2012) review MI-convex applications and Lubin, Yamangil, et al. (2016) characterize which nonconvex feasible regions are MI-convex-representable. Since an MI-convex problem without integrality restrictions is just a convex problem, MI-convex optimization generalizes both mixed-integer linear optimization (MILP) and convex optimization. This structure also leads to effective branch-and-bound (B&B) algorithms, which recursively partition the possible values of the integer variables in a search tree and obtain objective bounds and feasible solutions from convex subproblems.

A *nonlinear B&B* (B&B-NL) algorithm for an MI-convex problem solves a nonlinear subproblem that includes all of the convex constraints at every node of the search tree. The Bonmin solver package (Bonami, Biegler, et al., 2008) implements a B&B-NL variant by calling the derivative-oracle-based nonlinear programming (NLP) solver Ipopt to solve the subproblems. The relatively recent SCIP-SDP solver (Gally, Pfetsch, and Ulbrich, 2018) B&B-NL implementation for mixed-integer semidefinite (MISDP) problems uses a primal-dual conic interior-point solver for the SDP subproblems.

Typically, B&B-NL methods need to solve a large number of very similar nonlinear subproblems to near-global optimality and feasibility in order to obtain accurate objective bounds. Linear optimization (LP) solvers based on the Simplex algorithm are able to rapidly reoptimize after variable bounds are changed or linear cuts are added, thus typically benefiting from warm-starting much more so than state-of-the-art NLP or conic solvers. *B&B LP outer approximation* (B&B-OA) algorithms leverage LP warm-starting by solving a polyhedral relaxation of the nonlinear subproblem at every node. Implementations often take advantage of the speed and stability of advanced MILP branch-and-cut solvers.

B&B-OA algorithms differ in how they refine the polyhedral relaxations of the nonlinear constraints and how they obtain feasible solutions. In a *separation-based* algorithm, no nonlinear solver is used. At each node the LP optimal point is first checked for feasibility for the convex constraints; if the violation exceeds a positive tolerance, valid cuts separating the point are added to the LP, otherwise the point may be accepted as a new incumbent if it is integral. Commercial mixed-integer second order cone optimization (MISOCP) solvers typically use separation-based algorithms, but also occasionally solve SOCP subproblems to obtain feasible solutions and fathom nodes. Quesada and Grossmann (1992) and Leyffer (1993) describe *subproblem-based* B&B-OA algorithms that solve smooth subproblems at a subset of the nodes. The subproblems provide points at which cuts based on gradient inequalities can be derived as follows. For a convex function $f : \mathbb{R}^n \to \mathbb{R}$, the set $\mathcal{X} = \{x \in \mathbb{R}^n : f(x) \leq 0\}$ is convex. If f is smooth, then given a point $\bar{x} \in \mathbb{R}^n$, the following gradient cut yields a polyhedral relaxation of $\mathcal{X}$:

$$f(\bar{x}) + (\nabla f(\bar{x}))'(x - \bar{x}) \leq 0. \tag{4.1}$$

Bonami, Biegler, et al. (2008) found that Bonmin's B&B-OA method generally outperforms its B&B-NL method. Since both of these methods rely on NLP subproblems, they frequently fail in the presence of nonsmoothness. Continuous conic solvers are more numerically robust than derivative-oracle-based NLP solvers on nonsmooth problems (such as SOCPs and SDPs). For the special case of MISOCP, Drewes and Ulbrich (2012) propose a conic subproblem-based B&B-OA algorithm that derives cuts from subgradients satisfying subproblem KKT optimality conditions, and hence does not require smoothness assumptions.

Another advantage of conic solvers is that they return simple certificates proving primal or dual infeasibility or optimality of a primal-dual solution pair. Using the theory of conic duality, it is possible to describe an elegant OA algorithm for mixed-integer conic (MI-conic) problems that uses conic certificates returned by primal-dual conic solvers. Unlike the algorithm by Drewes and Ulbrich

(2012) for MISOCPs, the conic-certificate-based algorithm does not need to examine KKT conditions or solve a second modified subproblem in the case of infeasibility. Lubin, Yamangil, et al. (2016) propose and test this approach in an iterative OA algorithm. However a B&B algorithm using a single search tree, instead of solving a sequence of MILP instances each with their own search tree, is more flexible and likely to be significantly faster in practice. We fill this gap with the first conic-certificate-based B&B-OA algorithm.

4.1.2 Mixed-integer conic form

In this chapter, use the following general form for MI-conic problems:

$$\mathfrak{M}\begin{cases} \inf_{x\in\mathbb{R}^n} \quad c'x : & \text{(4.2a)} \\ \qquad h - Gx \in \mathcal{K}, & \text{(4.2b)} \\ \qquad\quad x_i \in \mathbb{Z} \qquad \forall i \in [\![I]\!], & \text{(4.2c)} \end{cases}$$

where $\mathcal{K} \subset \mathbb{R}^q$ is a closed convex cone and $\mathbb{Z}$ is the set of integers. Unlike in Chapters 1 to 3, we do not require $\mathcal{K}$ to be proper. This allows us to represent equality constraints with the zero cone $\{0\}$, which simplifies the notation in this chapter. Compared to Hypatia's primal conic form (1.7), $\mathfrak{M}$ lacks the explicit equality constraints $b - Ax = 0$, but has integer constraints (4.2c) on the first $I \leq n$ variables. Relaxing the integrality constraints results in a convex conic optimization problem.

Any MI-convex problem can be expressed in MI-conic form by homogenizing the convex constraints. As we discuss in Section 2.2, conic solvers recognize the cone $\mathcal{K}$ as a Cartesian product of standard primitive cones. A primitive cone cannot be written as a Cartesian product of two or more lower-dimensional cones. The standard nonnegative, second order, and exponential cones are sufficient for encoding all 333 MI-convex problems in MINLPLIB2 (Vigerske, 2018; Lubin, Yamangil, et al., 2017). With the standard PSD cone, we can additionally represent all problems in the Conic Benchmark Library (CBLIB) compiled by Friberg (2016). MI-conic representations are useful for constructing tight formulations for unions of convex sets (Lubin, Yamangil, et al., 2016; Lubin, Zadik, and Vielma, 2017; Vielma, 2018); see Section 5.6.1.

4.1.3 Overview

In Section 4.2, we start by reviewing the relevant foundations of conic duality and certificates. We then introduce the notion of $\mathcal{K}^*$ cuts, and describe how to refine LP OAs of conic constraints using certificates obtained from continuous primal-dual conic solvers. We describe our conic-certificate-based B&B-OA algorithm, and we show that it detects infeasibility or unboundedness or terminates with an optimal solution in finite time under minimal assumptions.

In Section 4.3, we demonstrate that a $\mathcal{K}^*$ cut from a conic certificate implies useful guarantees about the infeasibility or optimal objective of an LP OA, suggesting that our algorithm can often fathom a node immediately after solving the LP rather than proceeding to the expensive conic subproblem solve. We consider how these guarantees may be lost in the more realistic setting of an

LP solver with a positive feasibility tolerance, and propose a practical methodology for scaling a certificate $\mathcal{K}^*$ cut to recover similar guarantees.

In Section 4.4, we describe how to strengthen the LP OAs by *disaggregating* $\mathcal{K}^*$ cuts, and show that this methodology maintains the guarantees from Section 4.3. We argue for initializing the LP OAs using *initial fixed* $\mathcal{K}^*$ cuts, and offer a procedure for cheaply obtaining *separation* $\mathcal{K}^*$ cuts to cut off an infeasible LP OA solution. These proposed techniques require minimal modifications to our algorithm and are practical to implement.

In Section 4.5, we describe the software architecture and algorithmic implementation of Pajarito, our open source MI-convex solver.[1] This section may be of particular interest to advanced users and developers of mathematical optimization software. We emphasize that our implementations diverge from the idealized algorithmic description in Section 4.2, because of our decision to leverage powerful external mixed-integer linear (MILP) solvers through a solver-independent interface, MathProgBase.

In Section 4.6, we specialize our $\mathcal{K}^*$ cut techniques for the standard second order, exponential, and PSD cones. For the second order cone, we describe how to lift $\mathcal{K}^*$ cuts using an extended formulation, resulting in tighter LP OAs. For the PSD cone, we show how to strengthen $\mathcal{K}^*$ cuts to rotated second order cone constraints, which can be added to an MISOCP OA model.

In Section 4.7, we summarize computational experiments demonstrating the speed and robustness of Pajarito. We benchmark Pajarito and several MISOCP solver packages, and conclude that Pajarito is the fastest and most-reliable open source solver for MISOCP. Finally, we compare the performance of several of Pajarito's algorithmic variants on new MI-conic examples, demonstrating practical advantages of several methodological contributions from Sections 4.3, 4.4 and 4.6.

4.2 Branch-and-bound outer approximation algorithm

For an MI-conic problem $\mathfrak{M}$, we propose a B&B-OA algorithm, the first such method based on conic certificates. In Section 4.2.1, we describe the continuous conic subproblems that a B&B-NL algorithm would solve at each node, and review the relevant foundations of conic duality. In Section 4.2.2, we introduce the notion of $\mathcal{K}^*$ cuts and describe how to refine polyhedral relaxations of the conic subproblems using information from conic certificates. Finally, we outline our B&B-OA algorithm in Section 4.2.3, and discuss finiteness of convergence.

4.2.1 Continuous subproblems and conic duality

Recall from $\mathfrak{M}$ that the first $I \leq n$ variables in x are constrained to be integer. Branch-and-bound algorithms recursively partition the valid integer assignments, so for convenience we assume known finite lower bounds $l^0 \in \mathbb{Z}^I$ and upper bounds $u^0 \in \mathbb{Z}^I$ on the integer variables $x_1, \ldots, x_I$. At a node of the branch-and-bound search tree with lower bounds $l \in \mathbb{Z}^I$ and upper bounds $u \in \mathbb{Z}^I$ on the

[1] Although Pajarito solver was introduced in Lubin, Yamangil, et al. (2016), this early implementation used NLP solvers instead of primal-dual conic solvers for continuous subproblems, and was built to assess the value of extended formulations by counting iterations before convergence. To avoid confusing users, we moved this NLP-based functionality out of Pajarito and into *Pavito* solver at https://github.com/JuliaOpt/Pavito.jl.

integer variables, the natural continuous conic subproblem is $\mathfrak{C}(l,u)$:

$$\mathfrak{C}(l,u)\begin{cases} \inf_x \quad c'x : & (4.3a)\\ \quad h - Gx \in \mathcal{K}, & (4.3b)\\ \quad l_i - x_i \in \mathbb{R}_{\leq} \qquad \forall i \in [\![I]\!], & (4.3c)\\ \quad u_i - x_i \in \mathbb{R}_{\geq} \qquad \forall i \in [\![I]\!], & (4.3d) \end{cases}$$

where the bound constraints (4.3c) and (4.3d) are expressed in conic form using the nonpositive cone $\mathbb{R}_{\leq}$ and the nonnegative cone $\mathbb{R}_{\geq}$.

There exist primal-dual conic algorithms for $\mathfrak{C}(l,u)$ that are powerful in both theory and practice. The foundation for these methods and for much of this chapter is the elegant theory of conic duality (Ben-Tal and Nemirovski, 2001; Boyd and Vandenberghe, 2004). Recall that the cone $\mathcal{K}$ in 4.3b is a closed convex cone; we let $\mathcal{K}^*$ denote the dual cone of $\mathcal{K}$, i.e. the set of points that have nonnegative inner product with all points in $\mathcal{K}$:

$$\mathcal{K}^* = \{z \in \mathbb{R}^q : s'z \geq 0, \forall s \in \mathcal{K}\}. \tag{4.4}$$

$\mathcal{K}^*$ is also a closed convex cone (Boyd and Vandenberghe, 2004). The standard conic dual of $\mathfrak{C}(l,u)$ can be written as $\mathfrak{C}^*(l,u)$:

$$\mathfrak{C}^*(l,u)\begin{cases} \sup_{z,\mu,\nu} \quad -h'z - l'\mu - u'\nu : & (4.5a)\\ \quad c + G'z + \mu^0 + \nu^0 = 0, & (4.5b)\\ \quad z \in \mathcal{K}^*, & (4.5c)\\ \quad \mu \in \mathbb{R}^I_{\leq}, & (4.5d)\\ \quad \nu \in \mathbb{R}^I_{\geq}, & (4.5e) \end{cases}$$

where for ease of exposition we let $\mu^0 = (\mu_1, \ldots, \mu_I, 0) \in \mathbb{R}^n$ and similarly for ν^0. Note that the nonnegative and nonpositive cones are both self-dual, i.e. $\mathbb{R}^*_{\leq} = \mathbb{R}_{\leq}$ and $\mathbb{R}^*_{\geq} = \mathbb{R}_{\geq}$. The zero cone $\{0\}$ (containing only the origin) is dual to the free cone $\mathbb{R}$. The variables z in the dual constraint (4.5c) are associated with the primal constraint (4.3b), and similarly for (4.5d) and (4.3c), (4.5e) and (4.3d).

If the conic primal-dual pair $\mathfrak{C}(l,u)$–$\mathfrak{C}^*(l,u)$ is *well-posed*, then conic duality can be thought of as a simple generalization of LP duality. If $\mathcal{K}$ is polyhedral, then $\mathcal{K}^*$ is polyhedral, and hence $\mathfrak{C}(l,u)$ and $\mathfrak{C}^*(l,u)$ are both LPs. All LPs are well-posed. In particular, the inf and sup can be replaced with min and max, and the possible status combinations for $\mathfrak{C}(l,u)$ and $\mathfrak{C}^*(l,u)$ are: both infeasible, one unbounded and the other infeasible, or both feasible and bounded with equal objective values attained by optimal solutions. The conditions for well-posedness in conic duality are described by Friberg (2016), and are outside the scope of this chapter, so we assume that any primal-dual subproblem we encounter has the well-posed property.

Friberg (2016) discusses certificates that provide easily-verifiable proofs of unboundedness or

infeasibility of the primal or dual problems or of optimality of a given pair of primal and dual points. In terms of the primal subproblem $\mathfrak{C}(l,u)$, the three possible mutually-exclusive cases and their interpretations are as follows.

A dual improving ray certifies that $\mathfrak{C}(l,u)$ is infeasible, via the conic generalization of Farkas' lemma. The improving ray $(\bar{z},\bar{\mu},\bar{\nu}) \in \mathbb{R}^{q+2I}$ of $\mathfrak{C}^*(l,u)$ is a feasible direction for $\mathfrak{C}^*(l,u)$ along which the objective value of any feasible point of $\mathfrak{C}^*(l,u)$ can be improved indefinitely. It satisfies the following conditions:

$$-h'\bar{z} - l'\bar{\mu} - u'\bar{\nu} > 0, \tag{4.6a}$$
$$\bar{z} \in \mathcal{K}^*, \tag{4.6b}$$
$$\bar{\mu} \leq 0, \tag{4.6c}$$
$$\bar{\nu} \geq 0, \tag{4.6d}$$
$$G'\bar{z} + \bar{\mu}^0 + \bar{\nu}^0 = 0. \tag{4.6e}$$

Clearly, if $\mathfrak{C}^*(l,u)$ itself has a feasible point, then it is unbounded, otherwise it is infeasible. Conditions (4.6b) to (4.6e) imply that $(\bar{z},\bar{\mu},\bar{\nu})$ is feasible for a modified $\mathfrak{C}^*(l,u)$ problem in which $c = 0$.

A primal improving ray and a primal feasible point certifies that $\mathfrak{C}(l,u)$ is unbounded, because the improving ray is a feasible direction along which the objective value of the feasible point can be improved indefinitely. The improving ray $\bar{x} \in \mathbb{R}^n$ of $\mathfrak{C}(l,u)$ also implies infeasibility of $\mathfrak{C}^*(l,u)$ and satisfies the following conditions:

$$c'\bar{x} < 0, \tag{4.7a}$$
$$-G\bar{x} \in \mathcal{K}, \tag{4.7b}$$
$$\bar{x}_i = 0 \qquad \forall i \in [\![I]\!], \tag{4.7c}$$

and the feasible point $\hat{x} \in \mathbb{R}^n$ of $\mathfrak{C}(l,u)$ simply satisfies the primal feasibility conditions (4.3). Note that if the objective coefficients of the continuous variables are all zero ($c_{I+1} = \ldots = c_n = 0$), then conditions (4.7a) and (4.7c) can never be satisfied, so there cannot be a primal improving ray. This matches intuition because if the continuous variables have zero objective coefficients and the integer variables are bounded, $\mathfrak{M}$ cannot be unbounded.

A complementary solution pair certifies optimality for $\mathfrak{C}(l,u)$ of the primal feasible point $\hat{x} \in \mathbb{R}^n$ in the pair $(\hat{x},(\hat{z},\hat{\mu},\hat{\nu}))$, via conic weak duality. The dual feasible point $(\hat{z},\hat{\mu},\hat{\nu}) \in \mathbb{R}^{q+2I}$ is also optimal for $\mathfrak{C}^*(l,u)$, and the pair have equal primal and dual objective values:

$$c'\hat{x} = -h'\hat{z} - l'\hat{\mu} - u'\hat{\nu}. \tag{4.8}$$

4.2.2 Dynamic polyhedral relaxations

Recall from (4.4) that $s \in \mathcal{K}$ if and only if $z's \geq 0, \forall z \in \mathcal{K}^*$. This implies that a nonpolyhedral conic constraint $h - Gx \in \mathcal{K}$ has the following equivalent semi-infinite linear representation:

$$z'(h - Gx) \geq 0 \qquad \forall z \in \mathcal{K}^*. \tag{4.9}$$

We refer to a point $z \in \mathcal{K}^*$ as a $\mathcal{K}^*$ point, and call the corresponding linear constraint $z'(h - Gx) \geq 0$ a $\mathcal{K}^*$ cut. A $\mathcal{K}^*$ cut cannot exclude any point x that satisfies $h - Gx \in \mathcal{K}$, so any finite set of $\mathcal{K}^*$ cuts defines a valid polyhedral relaxation of the conic constraint (4.3b).

Given a finite set $\mathcal{Z} \subset \mathcal{K}^*$ of $\mathcal{K}^*$ points, consider modifying the subproblem $\mathfrak{C}(l, u)$ by relaxing the conic constraint and instead imposing the finite number of $\mathcal{K}^*$ cuts implied by $\mathcal{Z}$. We refer to the resulting LP as $\mathfrak{P}(\mathcal{Z}, l, u)$, which we write in inequality form rather than conic form:

$$\mathfrak{P}(\mathcal{Z}, l, u) \begin{cases} \min_x \quad c'x : & & \text{(4.10a)} \\ \qquad x_i \geq l_i & \forall i \in [\![I]\!], & \text{(4.10b)} \\ \qquad x_i \leq u_i & \forall i \in [\![I]\!], & \text{(4.10c)} \\ z'(h - Gx) \geq 0 & \forall z \in \mathcal{Z}. & \text{(4.10d)} \end{cases}$$

Since $\mathfrak{P}(\mathcal{Z}, l, u)$ and $\mathfrak{C}(l, u)$ have the same objective function, and the feasible set of $\mathfrak{P}(\mathcal{Z}, l, u)$ is a polyhedral relaxation of the feasible set of $\mathfrak{C}(l, u)$, solving $\mathfrak{P}(\mathcal{Z}, l, u)$ with an LP solver may give us useful information about $\mathfrak{C}(l, u)$. If $\mathfrak{P}(\mathcal{Z}, l, u)$ is infeasible, then $\mathfrak{C}(l, u)$ must be infeasible. If $\mathfrak{P}(\mathcal{Z}, l, u)$ has an optimal objective value of L, then $\mathfrak{C}(l, u)$ is either infeasible or has an optimal objective no smaller than L. In these cases, we may be able to immediately fathom the node by infeasibility or by bound, or even use a fractional optimal solution for $\mathfrak{P}(\mathcal{Z}, l, u)$ to make a branching decision, without needing to solve $\mathfrak{C}(l, u)$. However, if $\mathfrak{P}(\mathcal{Z}, l, u)$ is unbounded, it does not provide useful information about the status or optimal value of $\mathfrak{C}(l, u)$.

An LP solver based on the Simplex algorithm is able to rapidly reoptimize $\mathfrak{P}(\mathcal{Z}, l, u)$ at each new node, after the bounds on the integer variables are updated and any new $\mathcal{K}^*$ cuts are added. As noted by Skajaa, E. D. Andersen, and Ye (2013), state-of-the-art conic solvers benefit much less from warm-starting, so it may be computationally faster to sacrifice some information about the conic subproblems in order to avoid some expensive conic subproblem solves.

In analogy to B&B-OA algorithms based on gradient cuts, we add a new cut after every infeasible or bounded conic subproblem solve. Our key innovation, however, is to obtain this cut directly from the conic certificate found by the conic subproblem solver. Suppose that at some node, a primal-dual conic subproblem solver yields a dual improving ray $(\bar{z}, \bar{\mu}, \bar{\nu})$: from condition (4.6b), $\bar{z} \in \mathcal{K}^*$, so $\bar{z}$ is a $\mathcal{K}^*$ point. Now suppose that the subproblem solver yields a complementary solution $(\hat{x}, (\hat{z}, \hat{\mu}, \hat{\nu}))$: by the dual feasibility condition (4.5c), $\hat{z} \in \mathcal{K}^*$, so $\hat{z}$ is a $\mathcal{K}^*$ point. In both cases, a subvector of the ray or solution for the dual subproblem $\mathfrak{C}^*(l, u)$ allows us to augment $\mathcal{Z} \subset \mathcal{K}^*$, refining our LP OA model $\mathfrak{P}(\mathcal{Z}, l, u)$. In Section 4.3, we use conic duality theory to show that these $\mathcal{K}^*$ cuts derived from certificates encode important information about conic subproblems into the subsequent polyhedral

relaxations.

4.2.3 Conic-certificate-based algorithm

Our conic-certificate-based B&B-OA algorithm for the MI-conic problem $\mathfrak{M}$ is outlined in Algorithm 1. Recall that $\mathfrak{M}$ is in minimization form. Algorithm 1 maintains an upper bound U (initially ∞), a corresponding best feasible solution set $\mathcal{X}$ (initially empty), and a set of active nodes $\mathcal{N}$ of the search tree. A node (l, u, L) is characterized by the finite variable bound vectors l and u and a lower bound value L. The node's lower bound L signifies that all feasible solutions for $\mathfrak{M}$ that satisfy the node's bounds on integer variables have an objective value of at least L. The node set $\mathcal{N}$ is initialized to contain only the root node $(l^0, u^0, -\infty)$, where $l^0, u^0 \in \mathbb{R}^I$ are the finite initial global bounds on the integer variables.

On Line 5, the main loop removes a node (l, u, L) from $\mathcal{N}$. If the node's lower bound L is no smaller than the current global best upper bound U, Line 7 fathoms the node by bound as it cannot yield a better incumbent. Otherwise, Line 8 solves the node's LP OA model $\mathfrak{P}(\mathcal{Z}, l, u)$, taking advantage of an LP warm-start from a previous node.

If $\mathfrak{P}(\mathcal{Z}, l, u)$ is infeasible, Line 10 immediately fathoms the node by infeasibility. If $\mathfrak{P}(\mathcal{Z}, l, u)$ has an optimal solution $\hat{x}$, then its optimal objective value is the tightest lower bound known for $\mathfrak{C}(l, u)$ (in Section 4.3.1.2, we prove $c'\hat{x} \geq L$ is a consequence of the $\mathcal{K}^*$ cuts), so Line 12 updates L to $c'\hat{x}$. Line 14 fathoms the node by bound if L is no better than the incumbent value U, otherwise if $\hat{x}$ is fractional (i.e. it violates an integrality constraint (4.2c)), Line 17 branches on it. Note we could instead remove lines 15-17 and solve the conic subproblem even if the LP solution is fractional, rather than branching; this variation may perform better if the conic subproblem solves are quite fast in practice. The branch procedure strictly partitions the node's integer bounds l and u by picking an $i \in [\![I]\!] : \hat{x}_i \notin \mathbb{Z}$ and adding two child nodes to $\mathcal{N}$: $(l, (u_1, \ldots, \lfloor \hat{x}_i \rfloor, \ldots, u_n), L)$ and $((l_1, \ldots, \lceil \hat{x}_i \rceil, \ldots, l_n), u, L)$.

If the node is not fathomed or branched on immediately after the LP solve (before Line 18), then $\mathfrak{P}(\mathcal{Z}, l, u)$ is either unbounded or has an optimal solution $\hat{x}$ that is integral (i.e. $\hat{x}_i \in \mathbb{Z}, \forall i \in [\![I]\!]$) with optimal value $c'\hat{x} < U$. Then Line 18 solves the conic subproblem $\mathfrak{C}(l, u)$ with the primal-dual continuous conic solver. Recall from Section 4.2.1 our assumption that the primal-dual subproblem pair $\mathfrak{C}(l, u)$–$\mathfrak{C}^*(l, u)$ is well-posed, so the conic solver returns one of the three possible certificates, which we handle as follows.

A dual improving ray on Line 19 provides a $\mathcal{K}^*$ point, which Line 20 adds to $\mathcal{Z}$ (as described in Section 4.2.2). This certificate proves that $\mathfrak{C}(l, u)$ is infeasible, so Line 21 fathoms the node by infeasibility.

A primal improving ray and feasible point on Line 22 certifies that $\mathfrak{C}(l, u)$ is unbounded. Since the primal improving ray conditions (4.7a) to (4.7c) are the same for any conic subproblem, every subproblem is infeasible or unbounded, so $\mathfrak{M}$ is either infeasible or unbounded. The incumbent solution set must be empty and $U = \infty$. Line 23 checks whether the feasible point

Algorithm 1: Conic-certificate-based branch-and-bound LP outer approximation for $\mathfrak{M}$.

initialize incumbent solution set $\mathcal{X}$ to $\varnothing$, upper bound U to ∞
initialize $\mathcal{K}^*$ point set $\mathcal{Z}$ to $\varnothing$
initialize node list $\mathcal{N}$ with root node $(l^0, u^0, -\infty)$
while $\mathcal{N}$ *contains nodes* **do**
  remove a node (l, u, L) from $\mathcal{N}$
  if *lower bound* $L \geq U$ **then**
    continue ▷ fathomed by bound
  call LP solver on $\mathfrak{P}(\mathcal{Z}, l, u)$
  if *get an infeasibility proof* **then**
    continue ▷ fathomed by infeasibility
  else if *get an optimal solution* $\hat{x}$ **then**
    update L to $c'\hat{x}$
    if $L \geq U$ **then**
      continue ▷ fathomed by bound
    else if $\hat{x}$ *is fractional* **then**
      add branch nodes to $\mathcal{N}$ using $\hat{x}$ and L
      continue ▷ branched
  call primal-dual continuous conic solver on $\mathfrak{C}(l, u)$
  if *get a dual improving ray* $(\bar{z}, \bar{\mu}, \bar{\nu})$ **then**
    add $\mathcal{K}^*$ point $\bar{z}$ to $\mathcal{Z}$
    continue ▷ fathomed by infeasibility
  else if *get a primal improving ray* $\bar{x}$ *and feasible point* $\hat{x}$ **then**
    if $\hat{x}$ *is integral* **then**
      update U to $-\infty$
      break ▷ proven unbounded
  else if *get a complementary solution* $(\hat{x}, (\hat{z}, \hat{\mu}, \hat{\nu}))$ **then**
    add $\mathcal{K}^*$ point $\hat{z}$ to $\mathcal{Z}$
    update L to $c'\hat{x}$
    if $L \geq U$ **then**
      continue ▷ fathomed by bound
    else if $\hat{x}$ *is integral* **then**
      update $\mathcal{X}$ to $\{\hat{x}\}$ and U to $c'\hat{x}$
      continue ▷ fathomed by integrality
  add branch nodes to $\mathcal{N}$ using $\hat{x}$ (fractional) and L
return $\mathcal{X}$, U

$\hat{x}$ is integral. If so, it is a feasible solution for $\mathfrak{M}$, so $\mathfrak{M}$ is unbounded and Line 25 terminates the main loop.

A complementary solution on Line 26 provides a $\mathcal{K}^*$ point that Line 27 adds to $\mathcal{Z}$ (as described in Section 4.2.2) and an optimal solution $\hat{x}$ for $\mathfrak{C}(l,u)$. The optimal objective value gives the tightest lower bound known for the node, so Line 28 updates L to $c'\hat{x}$, and Line 30 fathoms by bound if this value is no better than U. Line 31 checks if $\hat{x}$ is integral, in which case it becomes the new incumbent solution for $\mathfrak{M}$ on Line 32, and the node is fathomed by integrality on Line 33.

If the node is not fathomed immediately after the conic solve (before Line 34), then $\hat{x}$ is a feasible solution for $\mathfrak{C}(l,u)$ that is fractional. L is either ∞ (in the primal improving ray case) or finite (in the complementary solution case), and is the best known lower bound for the node. Line 34 branches on $\hat{x}$ using the same branch procedure we describe above for Line 17.

Since the initial bounds on the integer variables are finite, and the main loop of Algorithm 1 either fathoms each node or strictly partitions its integer bounds or terminates the algorithm, it follows that Algorithm 1 terminates finitely. From the fact that $\mathfrak{P}(\mathcal{Z},l,u)$ is a valid polyhedral relaxation of $\mathfrak{C}(l,u)$, and from the correctness of our inferences from the subproblem certificates, it is clear Algorithm 1 terminates correctly, under the assumption of well-posed conic subproblems. On Line 35, if $U=\infty$, then $\mathfrak{M}$ is proven infeasible, otherwise if U is finite, then $\mathcal{X}$ contains an optimal solution for $\mathfrak{M}$, otherwise $U=-\infty$ and $\mathfrak{M}$ is proven unbounded.

We note that without using the LP $\mathfrak{P}(\mathcal{Z},l,u)$ (i.e. removing Lines 8 to 17 and not creating and augmenting $\mathcal{Z}$ on Lines 2, 20 and 27), we get a simple conic-certificate-based B&B-NL algorithm for $\mathfrak{C}(l,u)$, for which finite termination guarantees and correctness follow from the same assumptions and arguments. We have omitted any discussion of node selection or fractional variable selection for branching. MILP solvers can use LP certificates to make intelligent selections, and we expect that some of these LP-based criteria are generalizable to the conic case, as conic duality theory is a simple extension of LP duality under the well-posed assumption.

4.3 Polyhedral relaxation guarantees from conic certificates

Recall from Section 4.2.2 that $\mathcal{K}^*$ cuts yield valid polyhedral relaxations of the conic constraint $h-Gx\in\mathcal{K}$, and a certificate $\mathcal{K}^*$ cut can be obtained directly from the conic certificate for an infeasible or bounded and feasible subproblem $\mathfrak{C}(l,u)$. We demonstrate in Section 4.3.1 that a certificate $\mathcal{K}^*$ cut implies useful guarantees about the infeasibility or optimal objective of the LP OAs, suggesting that Algorithm 1 can often fathom a node immediately after solving the LP OA rather than proceeding to the expensive conic subproblem solve. By similar arguments, we expect that the certificate $\mathcal{K}^*$ cut may be useful at nearby nodes for duality based prepossessing such as reduced cost fixing (Gally, Pfetsch, and Ulbrich, 2018, sec. 7) or conflict analysis (Witzig, Berthold, and Heinz, 2017). In Section 4.3.2, we consider how these guarantees may be lost in the more realistic

setting of an LP solver with a positive feasibility tolerance, and propose a practical methodology for scaling a certificate $\mathcal{K}^*$ cut to recover similar guarantees.

4.3.1 Under an exact linear programming solver

We continue to assume well-posedness of every conic subproblem at every node. We consider what a certificate $\mathcal{K}^*$ from the conic subproblem $\mathfrak{C}(l,u)$ at a node with bounds l,u on the integer variables implies about the LP OA $\mathfrak{P}(\mathcal{Z},\underset{\sim}{l},\underset{\sim}{u})$ at a different node with bounds $\underset{\sim}{l},\underset{\sim}{u}$.

4.3.1.1 Infeasible subproblems

Suppose $(\bar{z},\bar{\mu},\bar{\nu})$ is an improving ray of the dual subproblem $\mathfrak{C}^*(l,u)$, certifying infeasibility of $\mathfrak{C}(l,u)$. Using properties (4.6a) to (4.6e) of this certificate, any point $x\in\mathbb{R}^n$ satisfying the bounds $\underset{\sim}{l}_i\leq x_i\leq\underset{\sim}{u}_i,\forall i\in[\![I]\!]$ at the new node also satisfies:

$$\begin{aligned}
\bar{z}'(h-Gx) &= h'\bar{z}-x'G'\bar{z} && \text{(4.11a)}\\
&= h'\bar{z}+x'\bar{\mu}^0+x'\bar{\nu}^0 && \text{(4.11b)}\\
&\leq h'\bar{z}+x'\bar{\mu}^0+x'\bar{\nu}^0+\textstyle\sum_{i\in[\![I]\!]}((\underset{\sim}{l}_i-x_i)\bar{\mu}_i+(\underset{\sim}{u}_i-x_i)\bar{\nu}_i) && \text{(4.11c)}\\
&= h'\bar{z}+\underset{\sim}{l}'\bar{\mu}+\underset{\sim}{u}'\bar{\nu} && \text{(4.11d)}\\
&= (h'\bar{z}+l'\bar{\mu}+u'\bar{\nu})-(l-\underset{\sim}{l})'\bar{\mu}-(u-\underset{\sim}{u})'\bar{\nu}. && \text{(4.11e)}
\end{aligned}$$

From property (4.6a) of the certificate, $h'\bar{z}+l'\bar{\mu}+u'\bar{\nu}<0$. If $l_i\leq\underset{\sim}{l}_i\leq\underset{\sim}{u}_i\leq u_i,\forall i\in[\![I]\!]$, then $(l-\underset{\sim}{l})'\bar{\mu}\geq 0$ and $(u-\underset{\sim}{u})'\bar{\nu}\geq 0$. In this case, the value (4.11e) is negative, so from (4.11a) to (4.11e), the certificate $\mathcal{K}^*$ cut $\bar{z}'(h-Gx)\geq 0$ is violated. Therefore, the certificate $\mathcal{K}^*$ cut from the infeasible subproblem $\mathfrak{C}(l,u)$ guarantees infeasibility of any LP OA $\mathfrak{P}(\mathcal{Z},\underset{\sim}{l},\underset{\sim}{u})$ in the subtree of the node with bounds l,u on the integer variables.

More importantly for Algorithm 1, the certificate $\mathcal{K}^*$ cut is likely to remain violated at 'nearby' nodes outside of this subtree, as the conditions (4.11a) to (4.11e) have a natural interpretation from global sensitivity analysis. Perturbing the bounds on the integer variables from l,u to $\underset{\sim}{l},\underset{\sim}{u}$ changes the upper bound on $\bar{z}'(h-Gx)$ through a linear dependence on the values $\mu\leq 0$ and $\nu\geq 0$ of the dual variables in the improving ray of $\mathfrak{C}^*(l,u)$.

4.3.1.2 Feasible subproblems

Suppose $(\hat{x},(\hat{z},\hat{\mu},\hat{\nu}))$ is a complementary solution pair for the subproblem $\mathfrak{C}(l,u)$, certifying optimality of the solution pair. Using the strong duality conditions (property (4.8) and feasibility for $\mathfrak{C}(l,u)$ and $\mathfrak{C}^*(l,u)$), any point $x\in\mathbb{R}^n$ satisfying the bounds $\underset{\sim}{l}_i\leq x_i\leq\underset{\sim}{u}_i,\forall i\in[\![I]\!]$ at the new node and the certificate $\mathcal{K}^*$ cut $\hat{z}'(h-Gx)\geq 0$ has objective value:

$$\begin{aligned}
c'x &= -(G'\hat{z}+\hat{\mu}^0+\hat{\nu}^0)'x && \text{(4.12a)}\\
&= -\hat{z}'Gx-x'(\hat{\mu}^0+\hat{\nu}^0) && \text{(4.12b)}
\end{aligned}$$

$$= -h'\hat{z} + \hat{z}'(h - Gx)' - x'(\hat{\mu}^0 + \hat{\nu}^0) \tag{4.12c}$$
$$\geq -h'\hat{z} - x'(\hat{\mu}^0 + \hat{\nu}^0) \tag{4.12d}$$
$$\geq -h'\hat{z} - x'(\hat{\mu}^0 + \hat{\nu}^0) - \textstyle\sum_{i \in [\![I]\!]}((\underset{\sim}{l}_i - x_i)\hat{\mu}_i + (\underset{\sim}{u}_i - x_i)\hat{\nu}_i) \tag{4.12e}$$
$$= -h'\hat{z} - \underset{\sim}{l}'\hat{\mu} - \underset{\sim}{u}'\hat{\nu} \tag{4.12f}$$
$$= (-h'\hat{z} - l'\hat{\mu} - u'\hat{\nu}) + (l - \underset{\sim}{l})'\hat{\mu} + (u - \underset{\sim}{u})'\hat{\nu} \tag{4.12g}$$
$$= c'\hat{x} + (l - \underset{\sim}{l})'\hat{\mu} + (u - \underset{\sim}{u})'\hat{\nu}. \tag{4.12h}$$

If $l_i \leq \underset{\sim}{l}_i \leq \underset{\sim}{u}_i \leq u_i, \forall i \in [\![I]\!]$, then $(l - \underset{\sim}{l})'\hat{\mu} \geq 0$ and $(u - \underset{\sim}{u})'\hat{\nu} \geq 0$. In this case, the value (4.12h) is no smaller than $c'\hat{x}$, the lower bound from the subproblem $\mathfrak{C}(l, u)$. Therefore, the certificate $\mathcal{K}^*$ cut from the feasible subproblem $\mathfrak{C}(l, u)$ guarantees that the optimal value of any LP OA $\mathfrak{P}(\mathcal{Z}, \underset{\sim}{l}, \underset{\sim}{u})$ in the subtree of the node with bounds l, u on the integer variables does not decrease, but may actually improve. If Algorithm 1 branches on Line 34 after solving a bounded and feasible conic subproblem to get the tightest lower bound, then when examining a child node, this objective guarantee ensures the node's lower bound L does not decrease when we update it to the optimal value of the LP OA on Line 12.

More importantly for Algorithm 1, at 'nearby' nodes outside of this subtree, the objective bounds implied by the certificate $\mathcal{K}^*$ cut in the LP OA model are likely to remain fairly tight. Perturbing the bounds on the integer variables from l, u to $\underset{\sim}{l}, \underset{\sim}{u}$ changes the lower bound on $c'x$ through a linear dependence on the values $\mu \leq 0$ and $\nu \geq 0$ of the dual variables in the complementary solution pair for $\mathfrak{C}(l, u)$.

4.3.2 Under a linear programming solver with a feasibility tolerance

So far, we have been assuming that the LP solver computes a solution that satisfies all the $\mathcal{K}^*$ cuts in the LP OAs exactly. In practice, LP solvers based on the Simplex method (except those that use rational arithmetic) enforce constraints up to an absolute constraint-wise violation tolerance $\delta > 0$ (typically set by the user). Therefore, a more realistic assumption is that any solution returned by the LP solver does not violate any $\mathcal{K}^*$ cut by more than δ, i.e. a $\mathcal{K}^*$ point z effectively yields a 'relaxed $\mathcal{K}^*$ cut' $z'(h - Gx) \geq -\delta$. Under this relaxed condition, we may lose the 'within-subtree' guarantees described in Section 4.3.1. However, noting that any positive scaling of a $\mathcal{K}^*$ point is still a $\mathcal{K}^*$ point, we demonstrate how to recover the infeasibility guarantee from Section 4.3.1.1 exactly, and the objective bound guarantee from Section 4.3.1.2 to within a given relative objective gap tolerance. Such an analysis appears to be novel in the MI-convex literature.

4.3.2.1 Infeasible subproblems

Suppose $(\bar{z}, \bar{\mu}, \bar{\nu})$ is an improving ray of the dual subproblem $\mathfrak{C}^*(l, u)$. From the property (4.6a) of the certificate and the conditions (4.11a) to (4.11e), any point $x \in \mathbb{R}^n$ satisfying the bounds

$l_i \leq x_i \leq u_i, \forall i \in [\![I]\!]$ and the relaxed certificate $\mathcal{K}^*$ cut condition $\bar{z}'(h - Gx) \geq -\delta$ must satisfy:

$$0 > h'\bar{z} + l'\bar{\mu} + u'\bar{\nu} \geq \bar{z}'(h - Gx) \geq -\delta. \tag{4.13}$$

Therefore, if $\delta > 0$ is sufficiently large, the relaxed certificate $\mathcal{K}^*$ cut condition fails to enforce the infeasibility guarantee from Section 4.3.1.1.

However, for a positive multiplier $\bar{\gamma} > 0$ satisfying:

$$\bar{\gamma} > \frac{\delta}{-h'\bar{z} - l'\bar{\mu} - u'\bar{\nu}} > 0, \tag{4.14}$$

we have $\bar{\gamma}(h'\bar{z} + l'\bar{\mu} + u'\bar{\nu}) < -\delta$. Therefore, the relaxed scaled certificate $\mathcal{K}^*$ cut condition $\bar{\gamma}\bar{z}'(h - Gx) \geq -\delta$ recovers the infeasibility guarantee within the subtree of the node from which the certificate is obtained. Note that the scaling factor (4.14) depends only on δ, problem data, and the certificate for the infeasible subproblem $\mathfrak{C}(l, u)$. We can modify Algorithm 1 on Line 20 to add the scaled $\mathcal{K}^*$ point $\bar{\gamma}\bar{z}$ to $\mathcal{Z}$.

4.3.2.2 Feasible subproblems

Suppose $(\hat{x}, (\hat{z}, \hat{\mu}, \hat{\nu}))$ is a complementary solution pair for the subproblem $\mathfrak{C}(l, u)$. From the conditions (4.12a) to (4.12h), any point $x \in \mathbb{R}^n$ satisfying the bounds $l_i \leq x_i \leq u_i, \forall i \in [\![I]\!]$ and the relaxed certificate $\mathcal{K}^*$ cut condition $\hat{z}'(h - Gx) \geq -\delta$ has objective value:

$$c'x \geq -h'\hat{z} + \hat{z}'(h - Gx) - l'\hat{\mu} - u'\hat{\nu} \geq L - \delta. \tag{4.15}$$

Recall $L = c'\hat{x} = -h'\hat{z} - l'\hat{\mu} - u'\hat{\nu}$ is the optimal objective value of $\mathfrak{C}(l, u)$ and $\mathfrak{C}^*(l, u)$. Therefore, the relaxed certificate $\mathcal{K}^*$ cut condition only enforces the objective guarantee from Section 4.3.1.2 to an absolute tolerance of δ. In general, it makes little sense for an objective guarantee to depend on the LP solver's feasibility tolerance.

Instead, for a relative optimality gap tolerance $\epsilon > 0$, we can easily motivate a relative objective gap condition such as:

$$\frac{L - c'x}{|L| + \theta} \leq \epsilon. \tag{4.16}$$

Consider a positive multiplier $\hat{\gamma} > 0$ satisfying:

$$\hat{\gamma} \geq \frac{\delta}{\epsilon(|L| + \theta)} > 0. \tag{4.17}$$

Modifying the conditions (4.15) for the relaxed scaled certificate $\mathcal{K}^*$ cut condition $\hat{\gamma}\hat{z}'(h - Gx) \geq -\delta$, we get $c'x \geq L - \delta/\hat{\gamma}$. Rearranging, this implies:

$$\frac{L - c'x}{|L| + \theta} \leq \frac{\delta}{\hat{\gamma}(|L| + \theta)} \leq \epsilon, \tag{4.18}$$

so by scaling the certificate $\mathcal{K}^*$ cut by $\hat{\gamma}$, we achieve the relative objective gap guarantee (4.16) within the subtree of the node from which the certificate is obtained. Note that the scaling factor (4.17)

depends only on ϵ, δ, problem data, and the certificate for the bounded and feasible subproblem $\mathfrak{C}(l, u)$. We can modify Algorithm 1 on Line 27 to add the scaled $\mathcal{K}^*$ point $\hat{\gamma}\hat{z}$ to $\mathcal{Z}$.

4.4 Tightening polyhedral relaxations

In Section 4.4.1, we outline a two-stage procedure for disaggregating $\mathcal{K}^*$ cuts to get stronger polyhedral relaxations, and show how to maintain the certificate $\mathcal{K}^*$ cut guarantees from Section 4.3. In Section 4.4.2, we argue for initializing the polyhedral relaxations using *initial fixed* $\mathcal{K}^*$ cuts, and in Section 4.4.3, we describe a procedure for cheaply obtaining *separation* $\mathcal{K}^*$ cuts to cut off an infeasible LP OA solution. All of our proposed techniques for tightening the LP OAs require minimal modifications to Algorithm 1 and are practical to implement.

4.4.1 Extreme ray disaggregation

Consider a set of $\mathcal{K}^*$ points $\mathcal{Z} = \{z^1, \dots, z^J\} \subset \mathcal{K}^*$. By aggregating the corresponding $\mathcal{K}^*$ cuts, we see they imply infinitely many $\mathcal{K}^*$ cuts:

$$z'(h - Gx) \geq 0 \qquad \forall z \in \text{cone}(\mathcal{Z}), \tag{4.19}$$

where $\text{cone}(\mathcal{Z})$ is the conic hull of $\mathcal{Z}$, i.e. the set of conic (nonnegative) combinations of $z^1, \dots, z^J$:

$$\text{cone}(\mathcal{Z}) = \{\alpha^1 z^1 + \cdots + \alpha^J z^J : \alpha^1, \dots, \alpha^j \geq 0\} \subset \mathcal{K}^*. \tag{4.20}$$

Thus for a redundant $\mathcal{K}^*$ point $z^{J+1} \in \text{cone}(\mathcal{Z})$, the polyhedral relaxation of the conic constraint $h - Gx \in \mathcal{K}$ implied by $\mathcal{Z} \cup \{z^{J+1}\}$ is no stronger than that implied by $\mathcal{Z}$ alone. An extreme ray of $\mathcal{K}^*$ is a point $z \in \mathcal{K}^*$ that cannot be written as a nontrivial conic combination of other points in $\mathcal{K}^*$ that are not positive rescalings of z. To maximize the efficiency of our polyhedral relaxations, we propose adding only extreme rays of $\mathcal{K}^*$ to the $\mathcal{K}^*$ point set $\mathcal{Z}$ maintained by Algorithm 1.

Recall from Section 4.1.2 that our closed convex cone $\mathcal{K}$ is encoded as a Cartesian product $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K$ of standard primitive cones $\mathcal{K}_1, \dots, \mathcal{K}_K$. A primitive closed convex cone cannot be written as a Cartesian product of two or more lower-dimensional closed convex cones (Friberg, 2016). If $\mathcal{K}$ is separable, then its dual cone $\mathcal{K}^*$ is also separable:

$$\mathcal{K}^* = (\mathcal{K}_1 \times \cdots \times \mathcal{K}_K)^* = \mathcal{K}_1^* \times \cdots \times \mathcal{K}_K^*. \tag{4.21}$$

We exploit this separability and our understanding of the structure of the standard primitive cones to disaggregate a $\mathcal{K}^*$ point z into extreme rays of $\mathcal{K}^*$.

First, we note that $z = (\tilde{z}^1, \dots, \tilde{z}^K) \in \mathcal{K}^*$, where $\tilde{z}^k \in \mathcal{K}_k^*, \forall k \in [\![K]\!]$. Second, for each $k \in [\![K]\!]$, we disaggregate $\tilde{z}^k$ into extreme rays of the primitive standard dual cone $\mathcal{K}_k^*$. This step is trivial for linear cones For second order, exponential, and PSD cones, we describe practical computational procedures for dual disaggregation in Section 4.6. For example, if $\mathcal{K}_k$ is a PSD cone, we disaggregate $\tilde{z}^k \in \mathcal{K}_k^*$ by performing an eigendecomposition on it; see Section 4.6.4. We have $\tilde{z}^k = \sum_{j \in [\![J_k]\!]} \tilde{z}^{k,j}$,

where $\tilde{z}^{k,j} \neq 0$ is an extreme ray of $\mathcal{K}_k^*$, for all $j \in [\![J_k]\!]$. We choose these extreme rays so that none is a positive scaling of another, and J_k does not exceed $\dim(\mathcal{K}_k^*)$. Note that $J_k = 0$ if $\tilde{z}^k = 0$.

For some $k \in [\![K]\!]$ and $j \in [\![J_k]\!]$, consider the point $z^{k,j} = (0, \ldots, 0, \tilde{z}^{k,j}, 0, \ldots, 0)$, which is nonzero only on the elements corresponding to the kth primitive dual cone. Since any cone contains the origin 0, and $\tilde{z}^{k,j} \in \mathcal{K}_k^*$, $z^{k,j} \in \mathcal{K}^*$ by (4.21). Furthermore, since $\tilde{z}^{k,j}$ is an extreme ray of $\mathcal{K}_k^*$, it cannot be written as a nontrivial sum of extreme rays of $\mathcal{K}_k^*$, and so $z^{k,j}$ cannot be written as a nontrivial sum of extreme rays of $\mathcal{K}^*$. Thus $z^{k,j}$ is an extreme ray of $\mathcal{K}^*$.

Our two-stage disaggregation procedure for $z \in \mathcal{K}^*$ yields $\sum_{k \in [\![K]\!]} J_k \leq \dim(\mathcal{K}) = q$ extreme rays of $\mathcal{K}^*$:

$$z = \textstyle\sum_{k \in [\![K]\!], j \in [\![J_k]\!]} z^{k,j}. \tag{4.22}$$

Besides adding potentially multiple $\mathcal{K}^*$ points to $\mathcal{Z}$, no modifications are needed to the description of Algorithm 1. Since z is clearly contained in the conic hull of these $\mathcal{K}^*$ points, there is no loss of strength in the polyhedral relaxations, so the certificate $\mathcal{K}^*$ guarantees from Section 4.3.1 are maintained. The polyhedral relaxations are potentially much tighter, improving the power of the LP OA for fathoming a node by infeasibility or objective bound without proceeding to an expensive conic subproblem solve. The LP solver may need to deal with more cuts at nodes visited early in the search tree, but is ultimately likely to need to examine fewer nodes overall and solve fewer expensive conic subproblems, so the tradeoff can be worthwhile.

We can also recover the guarantees from Section 4.3.2 for an LP solver with a feasibility tolerance $\delta > 0$. We assume z is a certificate $\mathcal{K}^*$ point that has already been scaled according to Section 4.3.2. After disaggregating z, we scale each extreme ray up by $J = \sum_{k \in [\![K]\!]} J_k$ before adding it to $\mathcal{Z}$. The J relaxed scaled disaggregated $\mathcal{K}^*$ cut conditions are:

$$(Jz^{k,j})'(h - Gx) \geq -\delta \qquad \forall k \in [\![K]\!], j \in [\![J_k]\!]. \tag{4.23}$$

Summing and using (4.22), and dividing by J, we see that these conditions imply the relaxed scaled original $\mathcal{K}^*$ cut condition $z'(h - Gx) \geq -\delta$.

4.4.2 Initial fixed polyhedral relaxations

We can modify Algorithm 1 on Line 2 to initialize a nonempty set $\mathcal{Z}$ of initial fixed $\mathcal{K}^*$ extreme rays that are not derived from subproblem certificates, but depend only on the geometry of $\mathcal{K}^*$. If $\mathcal{K}$ is a separable product of standard primitive cones, we can obtain initial fixed $\mathcal{K}^*$ extreme rays by treating each primitive cone constraint separately. In particular, a linear cone constraint need not be relaxed at all, since it is equivalent to one $\mathcal{K}^*$ cut (for a nonnegative or nonpositive cone) or two $\mathcal{K}^*$ cuts (for the zero cone). In Section 4.6, we describe simple sets of initial fixed $\mathcal{K}^*$ extreme rays for second order, exponential, or PSD cones. For example, for a PSD cone, we use the extreme rays of the polyhedral cone of diagonally dominant symmetric matrices as initial fixed $\mathcal{K}^*$ extreme rays; see Section 4.6.4. We show in Section 4.6 how knowledge of the initial fixed $\mathcal{K}^*$ extreme rays allows us to tailor our extreme ray disaggregation procedures from Section 4.4.1 for certificate $\mathcal{K}^*$ points to

further increase the strength of the polyhedral relaxations and reduce redundancy in $\mathcal{Z}$. However, to be able to recover the guarantees from Section 4.3.2 under an LP solver with a feasibility tolerance, we would need the ability to dynamically scale up the initial fixed $\mathcal{K}^*$ points.

4.4.3 Separation of infeasible points

Inspired by separation-based OA algorithms, we can modify Algorithm 1 on Line 16 to add separation $\mathcal{K}^*$ points to $\mathcal{Z}$ that cut off a fractional optimal LP solution $\hat{x}$ that violates the conic constraint, right before branching on $\hat{x}$. We show that a separation $\mathcal{K}^*$ point exists when $h - G\hat{x} \notin \mathcal{K}$. Since $\mathcal{K}$ is closed and convex, there exists a hyperplane (z, θ) that separates $\hat{s} = h - G\hat{x}$ from $\mathcal{K}$, i.e. $z'\hat{s} < \theta$ and $z's \geq \theta, \forall s \in \mathcal{K}$. Since the problem $\inf_{s \in \mathcal{K}} z's$ is homogeneous (as $\mathcal{K}$ is a cone) and the optimal value is bounded below by finite θ, the optimal value must equal zero. So $\theta \leq 0$, implying $z'\hat{s} < 0$ and $z's \geq 0, \forall s \in \mathcal{K}$. Thus $z \in \mathcal{K}^*$ (by definition (4.4) of $\mathcal{K}^*$), and it implies a $\mathcal{K}^*$ cut that separates $\hat{x}$ from the feasible set of the conic constraint.

A separation $\mathcal{K}^*$ point may fail to improve the objective lower bound from the LP OA, and does not in general possess the sort of guarantees from Section 4.3 that a certificate $\mathcal{K}^*$ point implies. However, deriving a separation $\mathcal{K}^*$ point can be much cheaper than solving a continuous conic subproblem. If $\mathcal{K}$ is a separable product of standard primitive cones, we can obtain separation $\mathcal{K}^*$ extreme rays easily by treating each primitive cone constraint separately. In Section 4.6, we describe practical computational methods for obtaining separation $\mathcal{K}^*$ extreme rays for the three standard primitive cones Pajarito supports. For example, we obtain separation $\mathcal{K}^*$ extreme rays for a point that violates a PSD cone constraint by performing an eigendecomposition on it; see Section 4.6.4.

4.5 Pajarito solver and related software

We describe the software architecture and algorithmic implementation of Pajarito, our open source MI-convex solver. This section may be of particular interest to advanced users and developers of mathematical optimization software. We emphasize that our implementations diverge from the idealized description of Algorithm 1 in Section 4.2.3, because of our decision to leverage powerful external mixed-integer linear (MILP/MIP) solvers through a limited solver-independent interface, MathProgBase. Developers of MI-conic software with low-level control of the MIP search tree are able to implement features of Algorithm 1 that we cannot implement in Pajarito. In Pajarito's readme file at `https://github.com/JuliaOpt/Pajarito.jl`, we provide more guidance on recommended ways of using the solver as well as default options and tolerances.

4.5.1 Integration with MathProgBase

Pajarito is integrated with the MathProgBase abstraction layer. MathProgBase is a standardized API in Julia for interacting with optimization solvers, designed in part to allow the user to write solver-independent code. It includes specifications for continuous and mixed-integer solvers that use linear/quadratic, conic, or oracle-based NLP (nonlinear programming) forms. As we discuss in

Section 5.2.1, MathProgBase has now been replaced by a redesigned API, MathOptInterface. The process of building Pajarito has motivated many of the improvements in MathOptInterface.

Pajarito itself implements MathProgBase's conic interface. Pajarito's use of conic form is a significant architectural difference from most existing MI-convex solvers, which interact with an MI-convex instance almost exclusively through NLP oracles to query values and derivatives of the constraint and objective functions. MathProgBase conic form can be described compactly from a constraint matrix in sparse or dense format, right-hand side and objective coefficient vectors, variable and constraint cones expressed as lists of standard vectorized primitive cones with corresponding ordered row indices, and a vector of variable types (continuous, binary, or general integer).

In addition to the basic linear cones (nonnegative, nonpositive, zero, and free cones), Pajarito recognizes three standard primitive nonpolyhedral cones: exponential cones (see Section 4.6.1), second order cones (see Section 4.6.2), and PSD cones (see Section 4.6.4). Pajarito also recognizes rotated second order cones, but for simplicity converts them to second order cones during preprocessing.

In Section 4.5.2, we summarize Pajarito's main algorithmic implementations. Pajarito uses the modeling package JuMP to conveniently build and manage the external MIP solver's OA model. JuMP itself interacts with the MIP solver via MathProgBase's linear/quadratic interface. To solve a continuous conic subproblem for a conic certificate, Pajarito calls the external primal-dual conic solver through the conic interface.

4.5.2 Basic algorithmic implementations

We discuss the main conic-certificate-based methods Pajarito uses to solve the MI-conic model $\mathfrak{M}$. We do not describe Pajarito's preprocessing techniques and we omit many options, enhancements, and numerical details. For explaining computational experiments, Section 4.7.3 briefly introduces several other algorithmic variants that we do not discuss here, such as separation-based methods that do not utilize conic certificates.

In Section 4.5.2.1, we summarize the initialization procedure for the OA model, an MILP relaxation of $\mathfrak{M}$ that Pajarito constructs and later refines (with extreme ray $\mathcal{K}^*$ cuts) using JuMP. In Section 4.5.2.2, we describe the 'iterative' method, an extension of the simple sequential OA algorithm by Lubin, Yamangil, et al. (2017). In Section 4.5.2.3, we describe the 'MIP-solver-driven' (MSD) method, so-called because it relies on the power of the branch-and-cut MIP solver to manage convergence in a single tree. Since MathProgBase's solver-independent abstraction for MIP solver callbacks is designed primarily around shared behavior between CPLEX and Gurobi, Pajarito is limited to interacting with the MIP solver through a *lazy cut callback* function and a *heuristic callback* function. Although the MSD method is generally faster than the iterative method, the latter may be used with MILP solvers for which callback functionality is unavailable or unreliable.

4.5.2.1 Initializing the outer approximation model

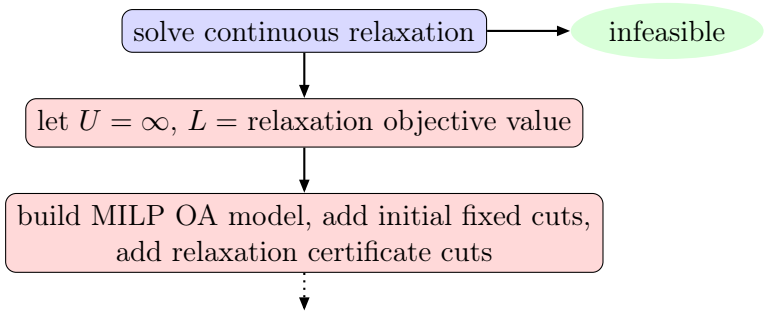
We first solve the continuous relaxation of $\mathfrak{M}$ (in which only the integrality constraints are relaxed), using the primal-dual conic solver (see top of Figure 4.1). The conic relaxation is analogous to

the first node subproblem in Algorithm 1, but without finite bounds on the integer variables. We preprocess this conic model slightly to tighten any non-integral bounds on the integer variables. If the conic solver indicates this relaxation is infeasible, then $\mathfrak{M}$ must be infeasible, so we terminate with an 'infeasible' status. If the conic solver returns a complementary solution pair, the optimal value gives an objective lower bound $L > -\infty$ for $\mathfrak{M}$. Otherwise, we set $L = -\infty$. We initialize the objective upper bound U for $\mathfrak{M}$ to ∞.

Using JuMP, we build the initial OA model, adding the variables and integrality constraints and setting the objective (see bottom of Figure 4.1). We then add initial fixed cuts for each primitive cone, as we describe in Section 4.4.2. Primitive linear cone constraints are imposed entirely (as equivalent LP equality or inequality constraints), and for each primitive nonpolyhedral cone, we add a small number of initial fixed cuts.

A complementary solution pair from the conic relaxation solve yields a $\mathcal{K}^*$ point, so we perform an extreme ray disaggregation from Section 4.4.1 and add certificate cuts for each primitive nonpolyhedral cone. These continuous relaxation certificate cuts technically guarantee that the root node of the OA model has an optimal value no smaller than L. This can be seen from a simple modification of the complementary solution case polyhedral relaxation guarantee we prove in Section 4.3.1.2, with trivial bounds on the integer variables. This is important because we cannot handle unboundedness of the OA model.

Figure 4.1: Pajarito's OA model initialization.



4.5.2.2 Iterative method

The iterative method, following initialization in Figure 4.1, is outlined in Figure 4.2. At each iteration of the main loop, Pajarito solves the current OA model using the MIP solver. We suggest the user set the MIP solver's relative optimality gap tolerance to its smallest possible value. If the OA model is infeasible, $\mathfrak{M}$ must be infeasible, so we terminate with an 'infeasible' status. If it is unbounded, Pajarito fails with an 'OA fail' error status, as we are unable to handle unbounded rays. If the MIP solver returns an optimal solution to the OA model, this OA solution satisfies the integrality constraints and initial fixed cuts, but in general not all of nonpolyhedral primitive cone constraints. The MIP solver's objective bound provides a lower bound for $\mathfrak{M}$, so we update L. Pajarito terminates with an 'optimal' status if the relative optimality gap condition (4.16) on L, U

is satisfied. Pajarito uses $\theta = 10^{-5}$ (to avoid division by 0). The gap tolerance $\epsilon > 0$ is specified by the user, but defaults to 10^{-5}.

If after solving the OA model we have an optimal OA solution and the objective bounds haven't converged, we check whether the OA sub-solution on the integer variables has been encountered before. If so, we check the conic feasibility of the OA solution. We calculate the absolute violation on each primitive nonpolyhedral cone constraint as the violation of the appropriate separation cut. If the worst absolute violation does not exceed Pajarito's feasibility tolerance (set by the user), then the OA solution is considered feasible. In this case, since the solution is optimal for the OA model, we can consider it optimal for $\mathfrak{M}$, so we update the incumbent and upper bound and terminate the solve immediately. If the OA solution is not considered feasible, we add all of the separation cuts that are (significantly) violated to the OA model.

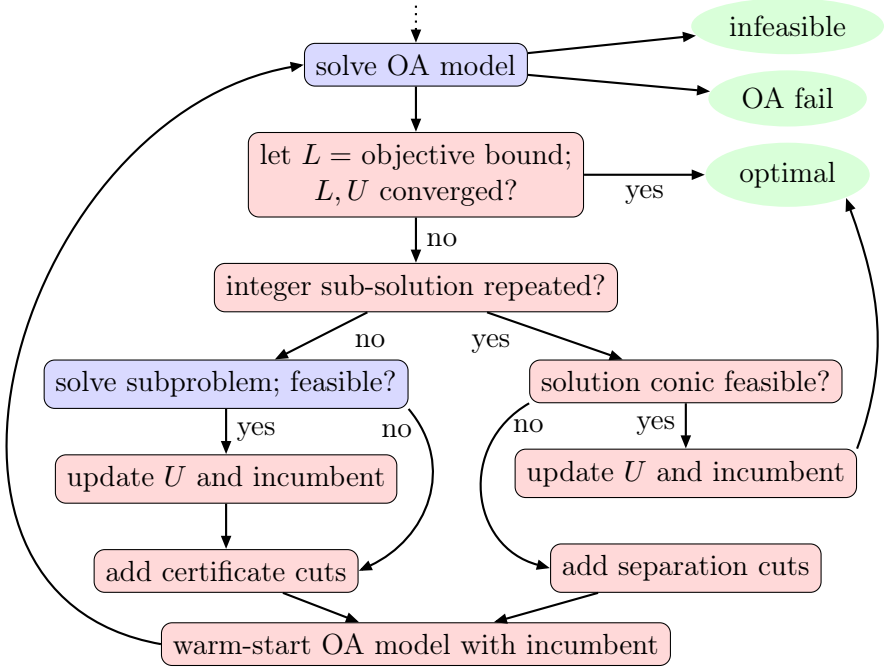
If the integer sub-solution has not already been encountered at a previous iteration, then we solve a continuous conic subproblem in which the integer variables are fixed to their values in the integer sub-solution. This subproblem is analogous to $\mathfrak{C}(l, u)$ from Section 4.2.1, with $l = u$. In preprocessing, we remove any subproblem equality constraints that effectively have no variables when an integer sub-solution is fixed. For efficient loading of the subproblem data at each iteration, we only change the constant vector h of the preprocessed conic subproblem, as this is the only data that changes.

Since it is more constrained than the OA model, the conic subproblem is bounded or infeasible. If the conic subproblem solver fails to return a certificate, we backtrack and perform the separation procedure (as if the integer sub-solution repeated). Otherwise, we scale the certificate's dual solution or dual ray according to Section 4.3.2 (using the tolerance values set as Pajarito options), then disaggregate the scaled $\mathcal{K}^*$ point and add extreme ray certificate cuts to the OA model (as we described for the continuous relaxation certificate in Section 4.5.2.1). In the case of a complementary solution pair, the primal solution yields a feasible point for $\mathfrak{M}$, since it satisfies both the integrality and conic constraints. If it has an objective value better than U, we update U and the incumbent solution and check the relative optimality gap condition again. Conic solvers typically do not use an absolute primitive constraint-wise feasibility tolerance, as Pajarito does for checking feasibility of OA solutions for the conic constraint. Our incumbent may not satisfy this notion of feasibility, since we do not perform a feasibility check on the conic solver's primal subproblem solutions.

After adding separation or certificate cuts, we warm-start the MIP solver with our incumbent and re-execute the main loop. The procedure in Figure 4.2 is iterated until L and U converge or the MIP solver detects infeasibility. If the user sets a time limit, Pajarito may terminate with the status 'user limit'. Pajarito sets the time limit on each MIP or conic solve to the remaining time. Note that the vast majority of Pajarito execution time is spent in MIP or conic solves. Since we only add cuts to the OA model on every loop, if the first OA model is bounded, then all subsequent (refined) OA models are bounded or infeasible, and the sequence of lower bounds L is nondecreasing. In case of failures of strong duality at some conic subproblems, Pajarito may fail to converge, as there exists no finite set of cuts that can tighten the lower bound sufficiently to meet the upper bound. See

Lubin, Yamangil, et al. (2016) for a discussion of strong duality in OA.

Figure 4.2: Pajarito's iterative method, following initialization.



4.5.2.3 Mixed-integer solver-driven method

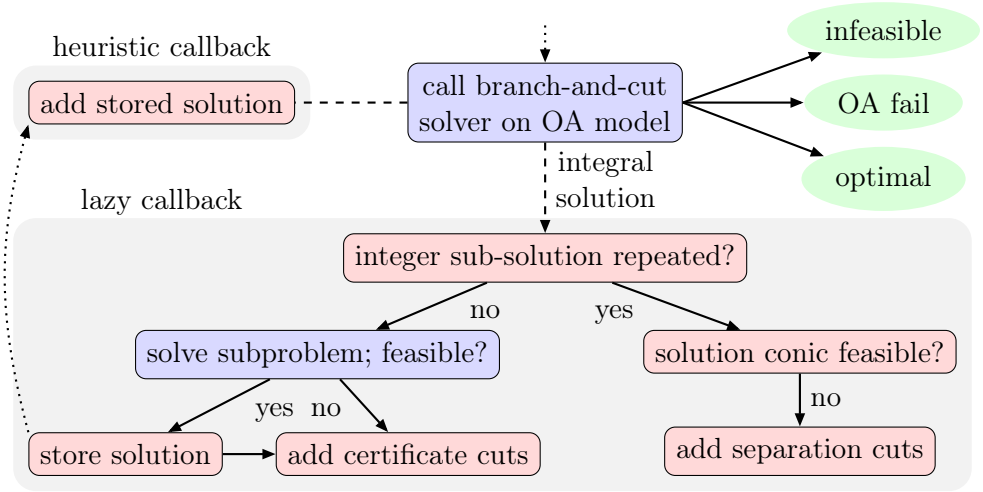
The MSD method, following initialization in Figure 4.1, is outlined in Figure 4.3. As in the iterative method, Pajarito returns an 'OA fail' status if the MIP solver detects unboundedness (as we are unable to handle unbounded rays), or an 'infeasible' status in the case of infeasibility. If the user sets a time limit, Pajarito sets a time limit on the MIP solver, and terminates with a 'user limit' status if this limit is reached. The MIP-solver-independent callback interface allows us to pass in lazy cuts in during a lazy callback and feasible solutions during a heuristic callback, however we cannot exert any control over branching decisions, node selection, fathoming, or node lower bound updating.

The MIP solver calls the lazy callback function whenever it finds an integer-feasible OA solution at a node. During a lazy callback, we first check whether the integral OA solution from the MIP solver is repeated. If so, we derive separation cuts to add as lazy constraints; if none can be added, the MIP solver considers the solution feasible and may update its incumbent. If the integer sub-solution is repeated, we solve a new (bounded or infeasible) conic subproblem. Since we lack the ability to query the node's bounds on the integer variables, we only solve subproblems with fixed integer sub-solutions, as in the iterative method. If the conic solver returns a certificate, we scale and disaggregate the $\mathcal{K}^*$ point (as we described for the iterative method in Section 4.5.2.2), and add extreme ray cuts as lazy constraints. The MIP solver is not guaranteed to respect the cuts that we add, and we may need to re-add the same cuts during multiple lazy callbacks (unlike in the iterative method, where cuts previously added are respected). For each repeated integer

sub-solution, we re-add these saved certificate cuts, in addition to the new separation cuts. In the case of a complementary solution pair, the primal solution yields a feasible point for $\mathfrak{M}$, which we store. During a heuristic callback, if there is a stored feasible solution to $\mathfrak{M}$ that has never been added as a heuristic solution, we add it.

Since there are no guarantees on when or how frequently the MIP solver calls the heuristic callback function, we may not be able to indirectly update the MIP solver's incumbent and upper bound when we are able to. Partly for this reason, Pajarito maintains its own upper bound and incumbent (not illustrated in Figure 4.3), which we update during lazy callbacks. During each lazy callback, we ask the MIP solver for its lower bound and check our relative optimality gap condition (as we described for the iterative method). If the condition is met, we force the MIP solver to terminate early (if the MIP solver allows). In this case, or if the MIP solver terminates with an optimal solution and we verify that the relative optimality gap condition is met, we return our incumbent solution with an 'optimal' status. Note the user is responsible for setting the desired relative optimality gap tolerance on both the MIP solver and on Pajarito directly.

Figure 4.3: Pajarito's MIP-solver-driven (MSD) method, following initialization.



4.5.3 Advanced algorithmic enhancements

We conclude by describing two optional OA enhancements we implemented in Pajarito. First, Pajarito by default uses an extended formulation (EF) for each second order cone constraint. Vielma, Dunning, et al. (2017) demonstrate on a testset of MISOCP problems that OA algorithms tend to converge much faster when using this extended representation for each second order cone constraint. Pajarito keeps the original natural second order cone formulation in the conic subproblems because conic solvers are likely to perform better with this representation than with the EF. In Section 4.6.3, we describe how to lift a $\mathcal{K}^*$ cut for the second order cone into $\mathcal{K}^*$ cuts for the EF. This technique also allows us to describe a much more economical set of initial fixed $\mathcal{K}^*$ cuts for the second order cone.

Second, Pajarito can optionally use an MISOCP OA model instead of an MILP OA model. There exist several powerful MISOCP solvers that can be used, or Pajarito itself may be used. Since a second order cone constraint can imply an infinite number of $\mathcal{K}^*$ cuts, Pajarito can achieve tighter relaxations of the conic constraint in the OA model. This may be helpful for example when the MI-conic problem has second order cones as well as exponential and/or PSD cones. Also, in Section 4.6.5 we demonstrate how to strengthen $\mathcal{K}^*$ cuts for PSD constraints to rotated-second order cone constraints. By solving SOCP relaxations of PSD constraints, we can reduce the number of expensive calls to an SDP subproblem solver. Note that for the MSD method, since most MISOCP solvers don't currently allow adding lazy quadratic constraints, only the initial fixed cuts can be strengthened in this way.

4.6 Standard primitive nonpolyhedral cones

In this section, we tailor the general $\mathcal{K}^*$ cut techniques for tightening OAs from Section 4.4 to the three nonpolyhedral cones recognized by Pajarito: the exponential cone in Section 4.6.1, the second order cone in Section 4.6.2, and the PSD cone in Section 4.6.4. In particular, we describe the extreme ray disaggregations (see Section 4.4.1), initial fixed cuts (see Section 4.4.2), and separation cuts (see Section 4.4.3) implemented in Pajarito. We also discuss the two advanced features from Section 4.5.3: the second order cone EF in Section 4.6.3, and second order conic cuts for PSD cones in Section 4.6.5.

4.6.1 Exponential cone

MathProgBase's exponential cone $\mathcal{K}_{\text{exp}} \subset \mathbb{R}^3$ is a permutation of the three-dimensional standard exponential cone we define in Section 2.2 (which is equivalent to our logarithm cone (2.19) for $d = 1$). $\mathcal{K}_{\text{exp}}$ is the epigraph of the perspective of the exponential function:

$$\mathcal{K}_{\text{exp}} \coloneqq \text{cl}\{(r,s,t) \in \mathbb{R}^3 : s > 0, r \geq s\exp(t/s)\} \tag{4.24a}$$
$$= \{(r,0,t) : r \geq 0, t \leq 0\} \cup \{(r,s,t) : s > 0, r \geq s\exp(t/s)\}, \tag{4.24b}$$
$$\mathcal{K}^*_{\text{exp}} \coloneqq \text{cl}\{(u,v,w) \in \mathbb{R}^3 : u > 0, w < 0, v \geq w - w\log(-w/u)\} \tag{4.24c}$$
$$= \{(u,v,0) : u, v \geq 0\} \cup \{(u,v,w) : u > 0, w < 0, v \geq w - w\log(-w/u)\}. \tag{4.24d}$$

Suppose we have a constraint $(r,s,t) \in \mathcal{K}_{\text{exp}}$. For initial fixed cuts, we use the two $\mathcal{K}^*_{\text{exp}}$ extreme rays $(1,0,0), (0,1,0)$ to impose the simple bound constraints $r, s \geq 0$. We also use $\mathcal{K}^*_{\text{exp}}$ extreme rays of the form $(1, w - w\log(-w), w)$ by picking several different values $w < 0$. Note the corresponding cuts separate any point $(0,0,t)$ satisfying $t > 0$.

Suppose we have the $\mathcal{K}^*_{\text{exp}}$ point (u,v,w). If $w = 0$, the point is already a nonnegative combination of the initial fixed $\mathcal{K}^*_{\text{exp}}$ points above, so we discard it. If $w < 0$, we use the $\mathcal{K}^*_{\text{exp}}$ extreme ray $(u, w - w\log(-w/u), w)$, which when added to some nonnegative multiple of $(0,1,0)$, gives (u,v,w). Note this also projects $(u,v,w) \notin \mathcal{K}^*_{\text{exp}}$ with $u > 0, w < 0$ onto $\mathcal{K}^*_{\text{exp}}$.

Suppose we want to separate a point $(r, s, t) \notin \mathcal{K}_{\text{exp}}$ that satisfies the initial fixed cuts. Then $r, s \geq 0$ and if $r = s = 0$ then $t \leq 0$. If $s = 0$, then $t > 0$ and $r > 0$, and we use the $\mathcal{K}^*_{\text{exp}}$ extreme ray $(t/r, -2 + 2\log(2r/t), -2)$. If $s > 0$, then $r < s\exp(t/s)$, and we use the $\mathcal{K}^*_{\text{exp}}$ extreme ray $(1, (t/s - 1)\exp(t/s), -\exp(t/s))$.

4.6.2 Second order cone

The second order cone (or Euclidean norm cone) and the rotated second order cone (or Euclidean norm square cone) are:

$$\mathcal{K}_{\ell_2} := \{(r, t) \in \mathbb{R}^{1+d} : r \geq \|t\|\}, \tag{4.25a}$$

$$\mathcal{K}_{\text{sqr}} := \{(r, s, t) \in \mathbb{R}^{2+d} : r \geq 0, s \geq 0, 2rs \geq \|t\|^2\}. \tag{4.25b}$$

These cones are self-dual. $\mathcal{K}_{\text{sqr}}$ is an invertible linear transformation of $\mathcal{K}_{\ell_2}$, since:

$$(r, s, t) \in \mathcal{K}_{\text{sqr}} \quad \Leftrightarrow \quad (r + s, r - s, \sqrt{2}t) \in \mathcal{K}_{\ell_2}, \tag{4.26}$$

and these equivalent representations have the same dimension. As noted in Section 4.5, Pajarito converts $\mathcal{K}_{\text{sqr}}$ constraints to $\mathcal{K}_{\ell_2}$ constraints during preprocessing.

We describe our initial fixed cuts for a constraint $(r, t) \in \mathcal{K}_{\ell_2}$. First, note that the ℓ_∞ norm lower-bounds the ℓ_2 norm, since $\|t\|_\infty = \max_{i \in [\![d]\!]} |t_i| \leq \|t\|$. We use the $2d$ $\mathcal{K}^*_{\ell_2}$ extreme rays $(1, \pm e_i), \forall i \in [\![d]\!]$, which imply $r \geq |t_i|, \forall i \in [\![d]\!]$, equivalent to the homogenized box relaxation $r \geq \|t\|_\infty$. Second, we note that the ℓ_1 norm also provides a lower bound for the ℓ_2 norm, since $\|t\|_1 = \sum_{i \in [\![d]\!]} |t_i| \leq \sqrt{d}\|t\|$. We use the 2^d $\mathcal{K}^*_{\ell_2}$ extreme rays $(1, \sigma/\sqrt{d}), \forall \sigma \in \{-1, 1\}^d$, which imply the homogenized diamond relaxation $r \geq \|t\|_1/\sqrt{d}$. Although the number of initial fixed cuts is exponential in d, in Section 4.6.3 we describe how Pajarito uses an EF to imply an initial fixed OA that is at least as strong but uses only a polynomial number of cuts. Note that the $\mathcal{K}^*_{\ell_2}$ point $(1, 0)$, which corresponds to the simple variable bound $r \geq 0$, is a nontrivial conic combination of these initial fixed $\mathcal{K}^*_{\ell_2}$ extreme rays.

Suppose we have the $\mathcal{K}^*_{\ell_2}$ point (u, w). If $w = 0$, the point is already a nonnegative multiple of the $\mathcal{K}^*_{\ell_2}$ point $(1, 0)$, so we discard it. Otherwise, we use the $\mathcal{K}^*_{\ell_2}$ extreme ray $(\|w\|, w)$, which when added to some nonnegative multiple of $(1, 0)$, gives the original point (u, w). Note this also projects $(u, w) \notin \mathcal{K}^*_{\ell_2}$ onto $\mathcal{K}^*_{\ell_2}$.

Suppose we want to separate a point $(r, t) \notin \mathcal{K}_{\ell_2}$ that satisfies the initial fixed cuts. Then $r \geq 0$ and so $t \neq 0$, and we use the $\mathcal{K}^*_{\ell_2}$ extreme ray $(1, -t/\|t\|)$.

4.6.3 Extended formulation for the second order cone

As discussed in Section 4.5.3, Pajarito can optionally use an EF for second order cone constraints proposed by Vielma, Dunning, et al. (2017), leading to tighter polyhedral relaxations. The constraint

$(r, t) \in \mathcal{K}_{\ell_2}$ is equivalent to the following constraints on r, t, and the auxiliary variables $\pi \in \mathbb{R}^d$:

$$2e'\pi \leq r, \tag{4.27a}$$

$$(r, \pi_i, t_i) \in \mathcal{K}_{\text{sqr}} \qquad \forall i \in [\![d]\!]. \tag{4.27b}$$

Suppose (u, w) is a $\mathcal{K}^*_{\ell_2}$ extreme ray, so from Section 4.6.2, we have $w \neq 0$ and $u = \|w\| > 0$. Then $ur + w't \geq 0$ is a $\mathcal{K}^*_{\ell_2}$ cut. Note that the linear constraint (4.27a) in the EF implies:

$$ur + w't \geq ur/2 + ue'\pi + w't = \textstyle\sum_{i \in [\![d]\!]}(w_i^2 r/(2u) + u\pi_i + w_i t_i). \tag{4.28}$$

For each $i \in [\![d]\!]$, consider the $\mathcal{K}^*_{\text{sqr}}$ extreme ray $(w_i^2/(2u), u, w_i)$, which implies a $\mathcal{K}^*_{\text{sqr}}$ cut for the ith constraint (4.27b) in the EF. The RHS of (4.28) is an aggregation of these d $\mathcal{K}^*_{\text{sqr}}$ cuts, which means the $\mathcal{K}^*_{\text{sqr}}$ cuts imply the $\mathcal{K}^*_{\ell_2}$ cut $ur + w't \geq 0$. Therefore, there is no loss of strength in the polyhedral relaxations, and we maintain the certificate $\mathcal{K}^*$ cut guarantees from Section 4.3.1. Note that without the ability to rescale the linear constraint (4.27a), we cannot recover the guarantees under an LP solver with a feasibility tolerance from Section 4.3.2.

We now apply this lifting procedure to the initial fixed $\mathcal{K}^*_{\ell_2}$ points described in Section 4.6.2. The $\mathcal{K}^*_{\ell_2}$ points for the ℓ_∞ norm relaxation are $(1, \pm e_i), \forall i \in [\![d]\!]$; for each $i \in [\![d]\!]$, we get three unique $\mathcal{K}^*_{\text{sqr}}$ extreme rays $(0, 1, 0)$ (for $w_i = 0$) and $(1/2, 1, \pm 1)$ (for $w_i = \pm 1$). The $\mathcal{K}^*_{\ell_2}$ points for the ℓ_1 norm relaxation are $(1, \sigma/\sqrt{d}), \forall \sigma \in \{-1, 1\}^d$; for each $i \in [\![d]\!]$, we get two unique $\mathcal{K}^*_{\text{sqr}}$ extreme rays $(1/(2d), 1, \pm 1/\sqrt{d})$ (for $w_i = \pm 1/\sqrt{d}$). The polyhedral relaxation implied by these $5d$ $\mathcal{K}^*_{\text{sqr}}$ points in the EF (4.27a) and (4.27b) is at least as strong as that implied by the $2d + 2^d$ $\mathcal{K}^*_{\ell_2}$ initial fixed points from Section 4.6.2, so our initial fixed OA can be imposed much more economically with the EF.

4.6.4 Positive semidefinite cone

MathProgBase, Pajarito, and our MI-conic form $\mathfrak{M}$ use vectorized cone definitions. The matrix cone $\mathbb{S}^d_\succeq$ is the smat PSD cone, and its equivalent vectorized definition $\mathcal{K}_\succeq \subset \mathbb{R}^{\text{sd}(d)}$ is the svec PSD cone (see Section 0.3). $\mathbb{S}^d_\succeq$ and $\mathcal{K}_\succeq$ are both self-dual. For convenience, we describe our cut techniques using the smat space.

The extreme rays of $\mathbb{S}^d_\succeq$ are the rank-one PSD matrices (Ben-Tal and Nemirovski, 2001), i.e. $\omega\omega' \in \mathbb{S}^d_\succeq$ for $\omega \in \mathbb{R}^d$. Hence an extreme ray $\mathbb{S}^*_\succeq$ cut has the form:

$$\langle \omega\omega', T \rangle = \omega' T \omega \geq 0. \tag{4.29}$$

We describe our initial fixed cuts for a constraint $T \in \mathbb{S}^d_\succeq$. Let $e_i \in \mathbb{R}^d$ be the ith unit vector in d dimensions. For each $i \in [\![d]\!]$, we let $\omega = e_i$ in (4.29), which imposes the diagonal nonnegativity condition $T_{i,i} \geq 0$ necessary for PSDness. For each $i, j \in [\![d]\!] : i > j$, we let $\omega = e_i \pm e_j$ in (4.29), which enforces the condition $T_{i,i} + T_{j,j} \geq 2|T_{i,j}|$ necessary for PSDness. Ahmadi and Hall (2015) discuss a polyhedral inner approximation of the PSD cone called the cone of diagonally dominant (DD) matrices. Our initial fixed $\mathbb{S}^*_\succeq$ points are exactly the extreme rays of the DD cone, so our initial fixed OA is the dual cone of the DD cone.

Suppose we have an $\mathbb{S}^*_\succeq$ point W, not necessarily an extreme ray of $\mathbb{S}^*_\succeq$. We perform an eigendecomposition $W = \sum_{i\in[\![d]\!]}\lambda_i\tilde{\omega}_i\tilde{\omega}_i'$, where for all $i \in [\![d]\!]$, λ_i is the ith eigenvalue and $\tilde{\omega}_i$ is the corresponding eigenvector. Since W is PSD, every eigenvalue is nonnegative, and there are $\text{rank}(W) \leq d$ positive eigenvalues. For each $i \in [\![d]\!] : \lambda_i > 0$, we let $\omega = \sqrt{\lambda_i}\tilde{\omega}_i$ in (4.29). These extreme ray $\mathbb{S}^*_\succeq$ cuts aggregate to imply the original $\mathbb{S}^*_\succeq$ cut $\langle W, T\rangle \geq 0$.

Suppose we want to separate a point $T \notin \mathbb{S}^d_\succeq$. We perform an eigendecomposition $T = \sum_{i\in[\![d]\!]}\lambda_i\tau_i\tau_i'$, for which at least one eigenvalue is negative. For each $i \in [\![d]\!] : \lambda_i < 0$, we let $\omega = \tau_i$ in (4.29). Each of these extreme ray $\mathbb{S}^*_\succeq$ cuts separates T because $\langle \tau_i\tau_i', T\rangle = \lambda_i < 0$.

4.6.5 Second order conic cuts for the positive semidefinite cone

For a PSD cone constraint $T \in \mathbb{S}^d_\succeq$, we demonstrate how to strengthen an $\mathbb{S}^*_\succeq$ extreme ray cut $\langle \omega\omega', T\rangle \geq 0$ to up to d three-dimensional $\mathcal{K}_{\text{sqr}}$ constraints. Note $\mathcal{K}_{\text{sqr}} \subset \mathbb{R}^3$ is a simple linear transformation of $\mathbb{S}^2_\succeq$. As discussed in Section 4.5.3, Pajarito can optionally solve an MISOCP OA model including these $\mathcal{K}_{\text{sqr}}$ constraints, leading to tighter relaxations.

We fix the index $i \in [\![d]\!]$. Let $\omega = \omega_i$ be the ith element of ω, and $\underline{\omega} = (\omega_j)_{j\in[\![d]\!]\setminus\{i\}} \in \mathbb{R}^{d-1}$ be the subvector of ω with the ith element removed. Similarly, let $\underline{t} = T_{i,i}$ and $\underline{t} = (T_{i,j})_{j\in[\![d]\!]\setminus\{i\}} \in \mathbb{R}^{d-1}$, and let $\underline{T} = (T_{k,j})_{k,j\in[\![d]\!]\setminus\{i\}} \in \mathbb{S}^{d-1}$ be the submatrix of T with the ith column and row removed. Kim, Kojima, and Yamashita (2003) prove a variant of the standard Schur-complement result that $T \in \mathbb{S}^d_\succeq$ if and only if $\underline{t} \geq 0$ and:

$$\underline{T} \in \mathbb{S}^{d-1}_\succeq, \tag{4.30a}$$

$$\underline{t}\underline{T} - \underline{t}\underline{t}' \in \mathbb{S}^{d-1}_\succeq. \tag{4.30b}$$

Consider the three-dimensional $\mathcal{K}_{\text{sqr}}$ constraint:

$$(\underline{t}, \underline{\omega}'\underline{T}\underline{\omega}, \sqrt{2}\underline{\omega}'\underline{t}) \in \mathcal{K}_{\text{sqr}}. \tag{4.31}$$

By the definition of $\mathcal{K}_{\text{sqr}}$, (4.31) is equivalent to $\underline{t} \geq 0$ and:

$$\underline{\omega}'\underline{T}\underline{\omega} \geq 0, \tag{4.32a}$$

$$\underline{\omega}'(\underline{t}\underline{T})\underline{\omega} \geq (\underline{\omega}'\underline{t})^2. \tag{4.32b}$$

(4.30a) implies (4.32a), by the dual cone definition (4.4). Since $(\underline{\omega}'\underline{t})^2 = \underline{\omega}'\underline{t}\underline{t}'\underline{\omega}$, (4.32b) is equivalent to $\underline{\omega}'(\underline{t}\underline{T} - \underline{t}\underline{t}')\underline{\omega} \geq 0$. Thus, by the dual cone definition again, (4.30b) implies (4.32b). Therefore, (4.31) is a valid relaxation of $T \in \mathbb{S}^d_\succeq$. Furthermore, from Theorem 3.3 of Kim, Kojima, and Yamashita (2003), (4.31) holds if and only if:

$$\langle W, T\rangle \geq 0 \qquad \forall W \in \mathbb{S}^d_\succeq : (W_{k,j} = \omega_k\omega_j, \forall k, j \in [\![d]\!]\setminus\{i\}). \tag{4.33}$$

Thus the constraint (4.31) potentially implies an infinite family of $\mathbb{S}^*_\succeq$ cuts, including the original $\mathbb{S}^*_\succeq$ cut $\langle \omega\omega', T\rangle \geq 0$. We do not explore how to scale these $\mathcal{K}_{\text{sqr}}$ constraints to recover the guarantees

from Section 4.3.2 for an SOCP solver with an absolute feasibility tolerance. Note that the choice of $i \in [\![d]\!]$ is arbitrary, so we can derive d different $\mathcal{K}_{\text{sqr}}$ constraints of the form (4.31). For strengthening separation or certificate $\mathcal{K}^*$ cuts, Pajarito heuristically picks one of the d possible $\mathcal{K}_{\text{sqr}}$ constraints by choosing i as the coordinate of the largest absolute value in ω.

We now apply this strengthening procedure to the initial fixed $\mathbb{S}^*_\succeq$ extreme rays described in Section 4.6.4. Letting $\omega = e_i \pm e_j$ for each $i, j \in [\![d]\!] : j > i$ in (4.31), we get the $d(d-1)/2$ initial fixed $\mathcal{K}_{\text{sqr}}$ constraints:

$$(T_{i,i}, T_{j,j}, \sqrt{2}T_{i,j}) \in \mathcal{K}_{\text{sqr}} \qquad \forall i, j \in [\![d]\!] : j > i. \tag{4.34}$$

These constraints enforce that every 2×2 principal matrix of T is PSD, a necessary but insufficient condition for $T \in \mathbb{S}^d_\succeq$. Ahmadi and Hall (2015) discuss an SOCP inner approximation of the PSD cone called the cone of scaled diagonally dominant (SDD) matrices. Our initial fixed SOCP OA is the dual SDD matrix cone, a strict subset of the polyhedral dual DD matrix cone that corresponds to our initial fixed polyhedral OA from Section 4.6.4.

4.7 Computational experiments

As we emphasize in Section 4.5, our algorithmic implementations in Pajarito differ from Algorithm 1 because of our practical decision to use external MILP solvers through a limited, solver-independent interface. In Section 4.7.1, we summarize our metrics for comparing the performance of MI-conic solvers, and we describe our presentation of tables and performance profile plots. In Section 4.7.2, we benchmark Pajarito and several MISOCP solvers on a library of MISOCP problems. In Section 4.7.3, we compare the performance of several of Pajarito's algorithmic variants on applied MI-conic problems over exponential, second order, and PSD cones. The scripts and results data for all experiments are available in the supplement at `http://github.com/chriscoey/PajaritoSupplement`. These experiments were run in 2018 with Pajarito version 0.5.1.

4.7.1 Presentation of results

We define a 'solver' as a MathProgBase solver object given a particular set of algorithmic options. Each solver we test is deterministic, i.e. it performs consistently on a particular dedicated machine. We define an 'instance' as a particular MI-conic problem that is known to be feasible and bounded. For a given instance, a solver may return a 'solution', which is a vector representing an assignment of the variables (not necessarily feasible).

First, we measure solver performance on a particular testset of instances by counting the number of instances for which the solver finds an approximately optimal solution. To be more precise, we use the following four categories to characterize a solver's apparent success or failure on an instance.

ex: (exclude) means either the solver incorrectly claims the instance is infeasible or unbounded, or the solver returns a solution it claims is approximately optimal but we detect one of the

following inconsistencies.

- The solution significantly violates at least one primitive cone constraint or integrality constraint. The absolute violation on a cone constraint is computed as the worst violation over the inequalities defining the cone (see Section 4.6; for the PSD cone, this is nonnegativity of the minimum eigenvalue). Our tolerances are 10^{-6} for linear/polyhedral cones, 10^{-5} for second order and exponential cones, and 10^{-4} for PSD cones. Variable-wise integrality violation is computed as distance to the nearest integer, and our tolerance is 10^{-6}.
- The relative objective gap condition (4.16) (which matches that used by most MIP solvers) for optimality is significantly violated. We set the constant $\theta = 10^{-5}$ (to avoid division by zero) and use the tolerance $\epsilon = 10^{-5}$.
- The objective value or objective bound significantly differs from that of a preponderance of other solvers (assessed semi-manually from output of our scripts).

co: (converge) means the solver returns a solution that it claims is approximately optimal (and it is not excluded for the reasons above).

li: (reach limit) means the solver does not terminate before the time limit, or (rarely) the solver reaches a memory limit and is forced to terminate.

er: (error) means the solver crashes or terminates with an error message.

Second, we compare aggregate measures of solver performance. Recall the definition of the shifted geometric mean in (1.52), which decreases the relative influence of smaller values, thus giving less weight to easy instances (small values are preferable for all of our metrics). We shift by 10 seconds for execution times, 1 for iteration counts, and 10 for MIP-solver-reported node counts. To compare a list of solvers $S_1, \ldots, S_n$ on a particular performance metric (such as execution time), we compute for each solver S_i the following three shifted geometric means, each over a different subset of the testset.

aco: (all solvers converge) is computed over the instances for which $S_1, \ldots, S_n$ all have a *co* status.

tco: (this solver converges) is computed over the instances for which S_i has a *co* status.

all: (all instances) is computed over all instances. Missing execution times are set to the time limit, and missing iteration/node counts are ignored.

Finally, we employ performance profiles (Dolan and Moré, 2002; Gould and Scott, 2016) to visually compare the relative execution times and iteration or node counts of pairs of solvers. Again, we decrease the relative influence of easy instances by shifting the metrics by the same shift values above. We describe how to interpret performance profiles at the end of Section 1.9.2.

4.7.2 Mixed-integer second order conic solver comparisons

Our open source Pajarito solvers, *Iter-GLPK* and *Iter-CBC*, use the iterative method (see Section 4.5.2.2) with ECOS (Domahidi, Chu, and Boyd, 2013) for continuous conic subproblems and CBC or GLPK for MILPs. We do not test Pajarito's MIP-solver-driven (MSD) method (see Section 4.5.2.3) with the CBC or GLPK MIP solvers because their support for MathProgBase callbacks is limited. Our two restricted-license Pajarito solvers, *Iter-CPLEX* (using the iterative method) and *MSD-CPLEX* (using the MSD method), call MOSEK's continuous conic solver and CPLEX's MILP solver.

The open source Bonmin solver package is described in detail by Bonami, Biegler, et al. (2008) and uses CBC to manage branching and Ipopt to solve continuous NLP (derivative-based nonlinear programming) subproblems. We are unaware of any mainstream open source solvers designed for MISOCP. The functional representation of the second order cone has points of nondifferentiability that may cause Bonmin to crash or suffer numerical issues. Our *Bonmin-BB* solver uses the nonlinear B&B method (no polyhedral approximation), *Bonmin-OA* uses the B&B OA method, and *Bonmin-OA-D* is equivalent to the *Bonmin-OA* solver but applied to transformed instances that use the second order cone EF we describe in Section 4.5.3.

Our two restricted-license MISOCP solvers are SCIP and CPLEX. Unlike Bonmin, these MISOCP solvers use the second order cone EF internally. CPLEX is available under an academic or commercial licence, and SCIP is an academic solver that is not released under an OSI-approved open source license. We use CPLEX version 12.7.0 and SCIP version 4.0.0.

These nine MISOCP solvers are each given a relative optimality gap tolerance of 10^{-5}. The SCIP and CPLEX solvers are given an absolute linear-constraint-wise feasibility tolerance of 10^{-8}, and CPLEX is given an integrality tolerance of 10^{-9}. The MILP solvers used by Pajarito are given an absolute linear-constraint-wise feasibility tolerance of 10^{-8}, an integrality tolerance of 10^{-9}, and a relative optimality gap tolerance of 0 for *Iter-GLPK*, *Iter-CBC*, and *Iter-CPLEX* and 10^{-5} for *MSD-CPLEX*. Due to limited resources, we set a one hour time limit for each run of a solver on an instance, and run all solvers (including the MILP and conic solvers called by Pajarito) in single-threaded mode.

We use a testset of 120 MISOCP instances drawn from the larger CBLIB library compiled by Friberg (2016). The testset contains randomly selected subsets of most of the major families of models in CBLIB. We exclude instances that are not bounded and feasible, or are solved in under 5 seconds by all solvers, or are unable to be solved by all solvers in under an hour. Our computations are performed on the Amazon EC2 cloud computing platform with m4.xlarge computing nodes having 16GB of RAM. As the computing nodes are virtual machines, timing results on EC2 are subject to random variability, but repeated runs suggest the variation is sufficiently small to avoid impacting our conclusions. The nodes run Ubuntu 16.04 with Julia version 0.6.0. Version information for the Julia packages can be obtained from the supplement.

Table 4.1 summarizes the status counts and shifted geometric means of performance metrics on instance subsets (explained in Section 4.7.1) for the nine MISOCP solvers on the 120 MISOCP

instances. The Bonmin solvers fail on most instances, and overall solve significantly fewer instances than the open source Pajarito solvers. Pajarito tends perform faster using CBC rather than GLPK. However, for most of the 9 excluded instances from *Iter-CBC*, we verify that CBC is responsible for the significant integrality violations that result in exclusion. Figure 4.4 (left) is a performance profile (explained in Section 4.7.1) comparing the execution times of the open source Pajarito (with CBC) solver and the instance-wise best of the three Bonmin solvers. From these results, we claim that Pajarito with ECOS and CBC is the fastest and most reliable open source MISOCP solver.

Using CPLEX, Pajarito's MSD method is significantly faster and more reliable than its iterative method. For *MSD-CPLEX*, the two errors occur where Pajarito claims a solution is suboptimal and has an objective gap no worse than 1.04×10^{-5}, and the one exclusion occurs where Pajarito's solution violates a linear constraint by 9.78×10^{-6}. The performance profile Figure 4.4 (right) compares the execution times of Pajarito's MSD method using CPLEX's MILP solver against CPLEX's specialized MISOCP solver. The execution time comparisons between *CPLEX* and *MSD-CPLEX* are ambiguous, however we argue that, at least by our metrics, Pajarito is a more reliable MISOCP solver.

Table 4.1: MISOCP solver performance summary.

		statuses				time (s)		
	solver	co	li	er	ex	aco	tco	all
open source	Bonmin-BB	34	44	11	31	38.0	83.8	463
	Bonmin-OA	25	53	29	13	64.2	64.5	726
	Bonmin-OA-D	30	48	29	13	15.1	61.6	610
	Iter-GLPK	56	60	3	1	2.0	29.7	377
	Iter-CBC	78	30	3	9	1.6	50.3	163
restricted	SCIP	74	35	8	3	3.2	41.5	160
	CPLEX	90	16	5	9	0.9	16.1	50
	Iter-CPLEX	86	26	0	8	0.4	37.0	106
	MSD-CPLEX	97	20	2	1	0.4	18.2	56

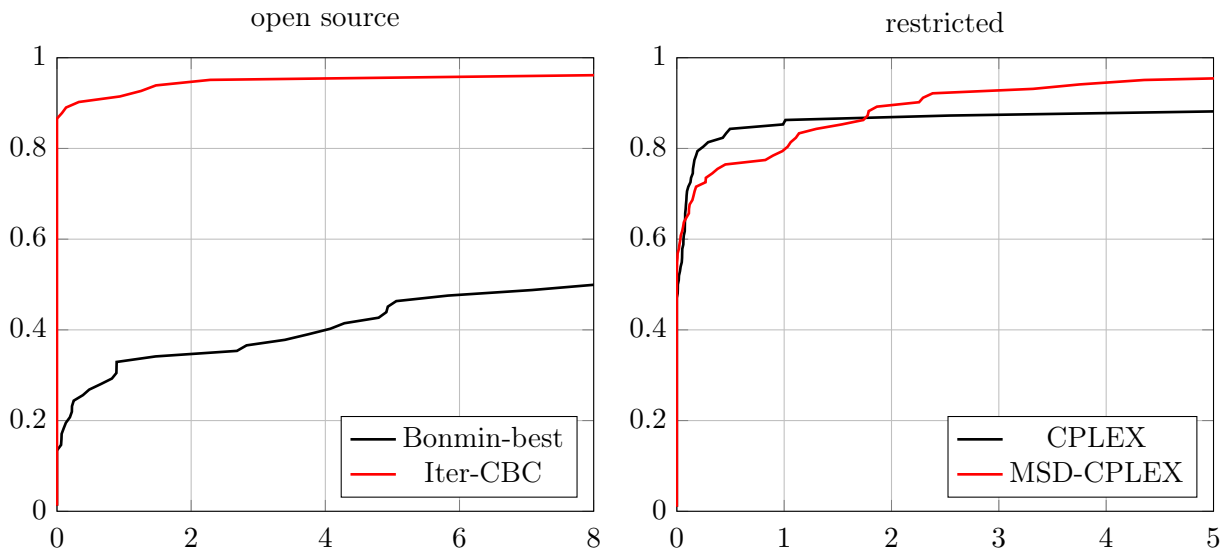
4.7.3 Comparisons of algorithmic variants

To compare the performance of several of Pajarito's algorithmic variants, we use a testset of 95 MI-conic instances. These instances are all bounded and feasible and come from the following four sources. Formulations for instances we generated can be found at https://github.com/JuliaOpt/Pajarito.jl/tree/master/examples.

Discrete experiment design (14 instances). Boyd and Vandenberghe (2004, Chapter 7.5) describe MI-convex experiment design problems. We generate A-optimal and E-optimal instances that include PSD cones, and D-optimal instances that include PSD and exponential cones.

Portfolios with mixed risk constraints (16 instances). We formulate a portfolio problem that

Figure 4.4: Performance profiles of MISOCP solver execution time. Left: Open source Bonmin (instance-wise best of 3) and Pajarito iterative solvers. Right: CPLEX MISOCP and Pajarito MSD solvers.



maximizes expected returns subject to some combinatorial constraints on stocks and three types of convex risk constraints on subsets of stocks with known covariances. Each instance includes multiple exponential cones from entropy risk constraints, second order cones from norm risk constraints, and PSD cones from robust norm risk constraints.

Retrofit-synthesis of process networks (32 instances). We select a subset of the two CBLIB families 'syn' and 'rsyn'. Each instance includes exponential cones.

A subset of the MISOCP testset (33 instances). We select a subset of the CBLIB families 'estein', 'ccknapsack', 'sssd', 'uflquad', and 'portfoliocard'.

We use Pajarito with Gurobi version 7.5.2 as the MILP solver and MOSEK version 9.0.0.29-alpha as the continuous conic solver. Note MOSEK 9 is the first version to recognize exponential cones. Pajarito is given a relative optimality gap tolerance of 10^{-5}. Gurobi is given an absolute linear-constraint-wise feasibility tolerance of 10^{-8}, an integrality tolerance of 10^{-9}, and a relative optimality gap tolerance of 0 when the iterative method is used and 10^{-5} when the MSD method is used. We set a one hour time limit for each run of a solver on an instance, and limit Gurobi and MOSEK to 8 threads. We run the computations on dedicated hardware with 16 Intel Xeon E5-2650 CPUs (2GHz) and 64GB of RAM. Repeated runs suggest the variation is sufficiently small to avoid impacting our conclusions. The machine runs Ubuntu 17.10 and Julia 0.6.2. Version information for the Julia packages can be obtained from the supplement.

4.7.3.1 Initial fixed cuts, certificate cuts, and separation cuts

Recall from Section 4.5.2 that Pajarito by default uses three different types of $\mathcal{K}^*$ cuts for non-polyhedral cones: initial fixed cuts, certificate cuts (from conic subproblems), and separation cuts (which separate infeasible OA solutions found by the MILP solver). We compare the following four important algorithmic variants of Pajarito that use different combinations of these three types of $\mathcal{K}^*$ cuts.

c: only add certificate cuts and only obtain feasible solutions from the primal subproblems.

cs: modify *c* above to add separation cuts on repeated integral OA solutions and accept approximately feasible OA solutions.

ics: modify *cs* above to add initial fixed cuts; this is Pajarito's default algorithm described in Section 4.5.2.

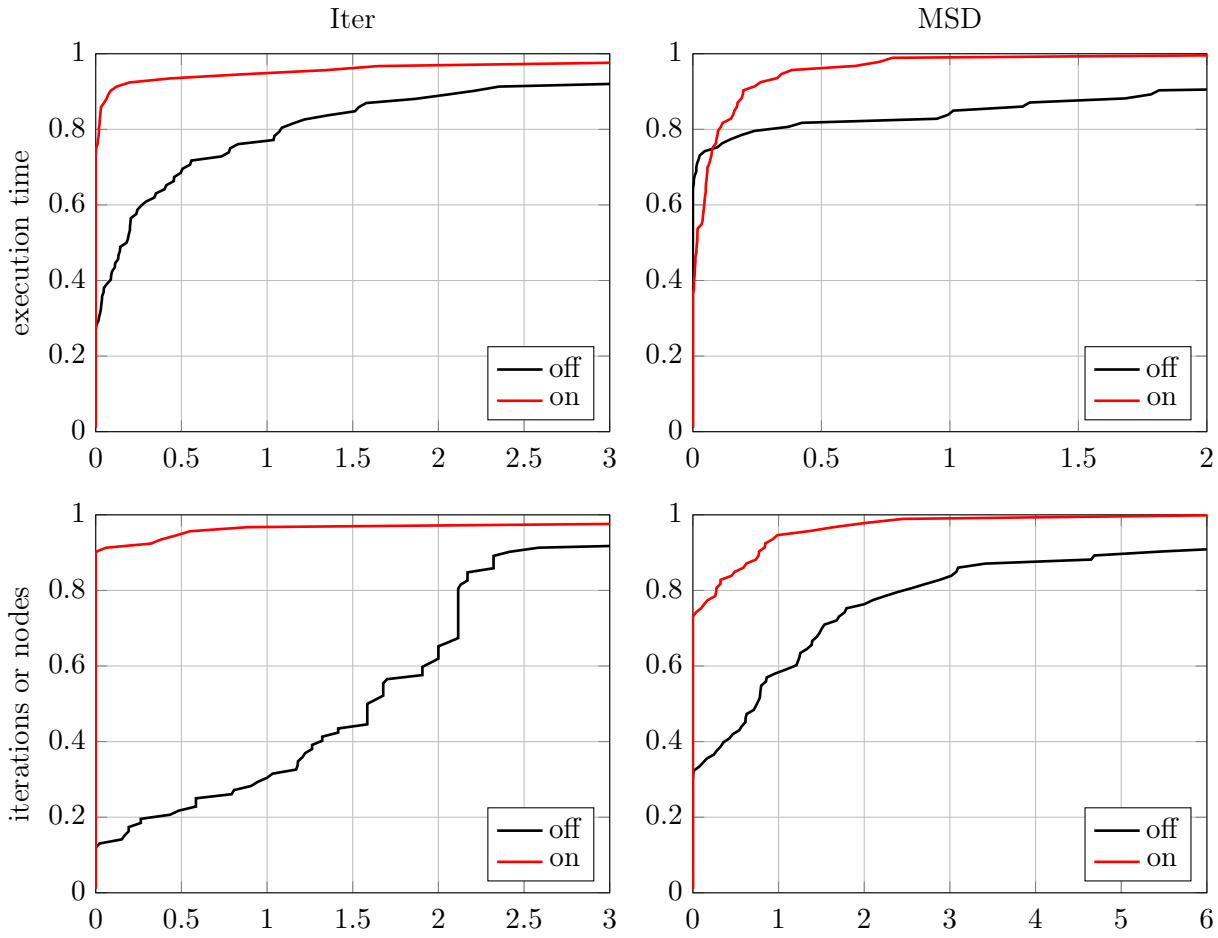
is: do not solve conic subproblems, only add initial fixed cuts and separation cuts, and accept approximately feasible OA solutions.

Table 4.2 summarizes the status counts and shifted geometric means of performance metrics on instance subsets. When using certificate cuts only (*c*), Pajarito often fails to converge numerically to the desired optimality gap (though typically it comes close). By also using separation cuts on repeated integer sub-solutions and accepting approximately conic feasible OA solutions as incumbents, Pajarito is able to converge on many more instances. Starting with initial fixed cuts (*ics*) further increases Pajarito's robustness, particularly for the MSD method. Comparing the *ics* variant against the separation-based variant *is*, we see significantly faster overall performance and fewer iterations or nodes for the subproblem-based *ics* variant. The performance profiles in Figure 4.5 compare the execution times or iteration/node counts for the *ics* and *is* solvers, demonstrating superiority of Pajarito's default *ics* method.

Table 4.2: $\mathcal{K}^*$ cut types performance summary.

		statuses				time (s)			subproblems			iters or nodes		
	cuts	co	li	er	ex	aco	tco	all	aco	tco	all	aco	tco	all
Iter	c	72	1	21	1	5.6	6.5	7.2	5.3	5.4	4.2	5.5	5.5	4.5
	cs	88	1	3	3	5.6	12.7	14.4	5.3	7.0	6.4	5.5	7.2	6.8
	ics	89	2	0	4	4.7	11.6	14.8	4.2	5.9	6.0	4.3	6.2	6.3
	is	84	1	0	10	8.4	14.5	22.1	-	-	-	13.4	16.5	18.1
MSD	c	76	0	18	1	2.4	3.4	3.5	12.6	15.8	12.7	223	438	348
	cs	88	0	5	2	3.3	6.5	7.8	19.0	26.8	24.9	295	843	815
	ics	92	0	1	2	2.2	6.3	6.5	15.6	24.6	25.0	273	796	857
	is	84	1	0	10	3.1	5.3	7.5	-	-	-	522	932	1345

Figure 4.5: $\mathcal{K}^*$ cut types performance profiles.



4.7.3.2 Extreme ray disaggregation

To test the $\mathcal{K}^*$ extreme ray disaggregation techniques in Sections 4.4.1 and 4.6, we run Pajarito using only certificate cuts (the *c* variant in Section 4.7.3.1), with and without disaggregation. Note that disabling disaggregation disables use of the second order cone EF, which has no benefit without disaggregation.

Table 4.3 summarizes the status counts and shifted geometric means of performance metrics on instance subsets, and the performance profiles in Figure 4.6 compare the execution times or iteration/node counts. For both the iterative and MSD methods, disaggregation improves performance on nearly every solved instance. For the iterative method, it enables the pure-certificate-based variant to converge on more than twice as many instances, and it more than halves the execution time and iteration count. Without disaggregation, the MSD method manages to converge on many more instances than the iterative method. Disaggregation greatly improves the performance of the MSD method, though the comparison is not quite as striking as for the iterative method.

Table 4.3: $\mathcal{K}^*$ cut disaggregation performance summary.

		statuses				time (s)			subproblems			iters or nodes		
	disag	co	li	er	ex	aco	tco	all	aco	tco	all	aco	tco	all
Iter	off	33	10	52	0	11.0	11.1	17.2	11.6	12.4	11.8	12.0	12.7	12.5
	on	72	1	21	1	4.5	6.5	7.2	4.0	5.4	4.2	4.3	5.5	4.5
MSD	off	51	3	41	0	1.7	6.2	6.7	15.5	50.5	28.0	70	613	261
	on	76	0	18	1	1.1	3.4	3.5	7.6	15.8	12.7	36	438	348

4.7.3.3 Certificate-based scaling

Finally, we test the $\mathcal{K}^*$ certificate cut scaling techniques for an LP solver with a feasibility tolerance, described in Section 4.3.2. We run Pajarito using only certificate cuts (the *c* variant in Section 4.7.3.1), with and without scaling. We set a larger feasibility tolerance on these four Pajarito solvers ($\delta = 10^{-6}$ instead of 10^{-8}, which we used for all other tests), to reduce the chance that any observed effects are caused by numerical issues near machine epsilon.

Table 4.4 summarizes the status counts and shifted geometric means of performance metrics on instance subsets, and the performance profiles in Figure 4.7 compare the execution times or iteration/node counts. For both the iterative and MSD methods, using scaling improves the robustness of the pure-certificate-based variant, allowing us to converge on 6 or 7 additional instances. On the subset of instances solved by all four solvers (the *aco* columns), scaling slightly reduces conic subproblem counts and iteration or node counts, but has small and ambiguous effects on the execution times.

Figure 4.6: $\mathcal{K}^*$ cut disaggregation performance profiles.

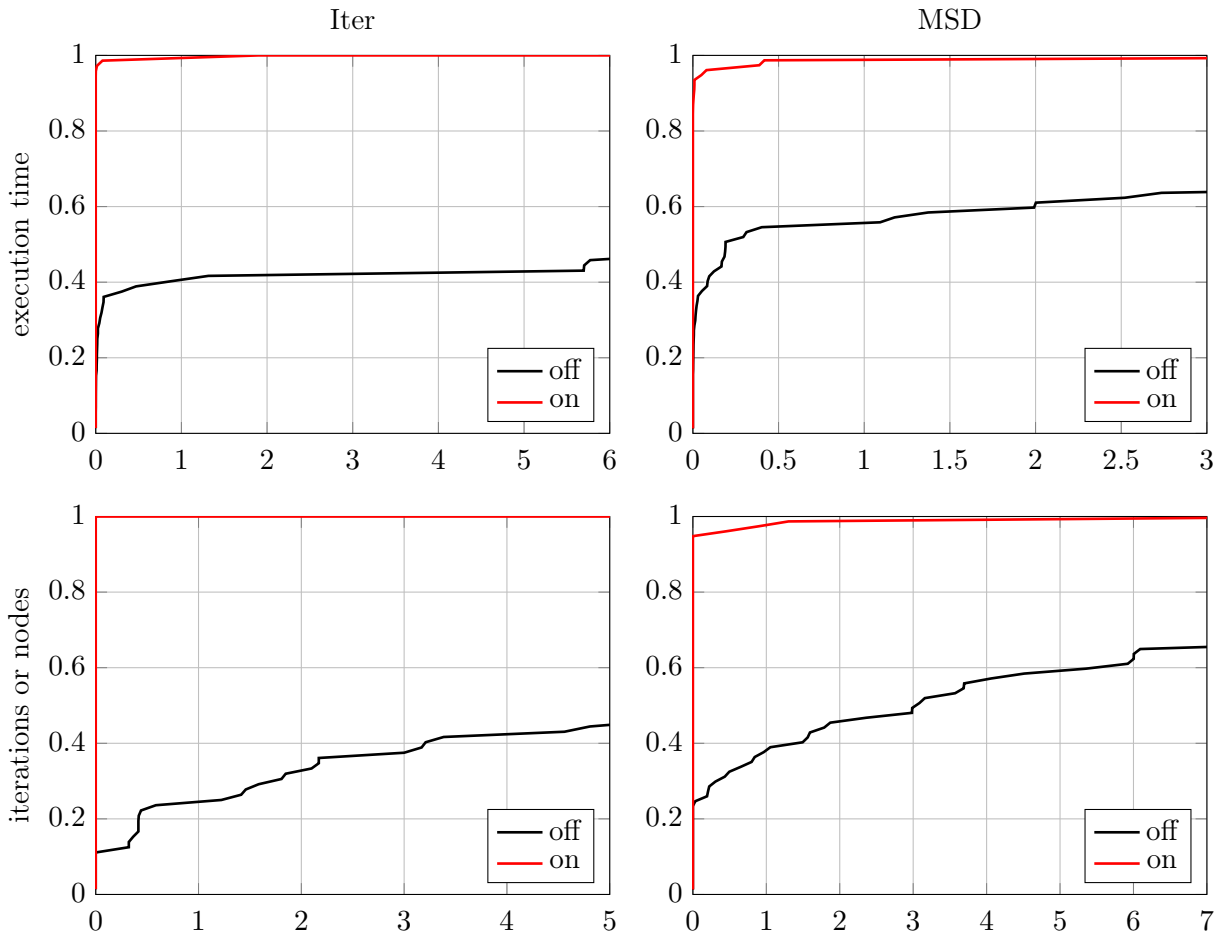
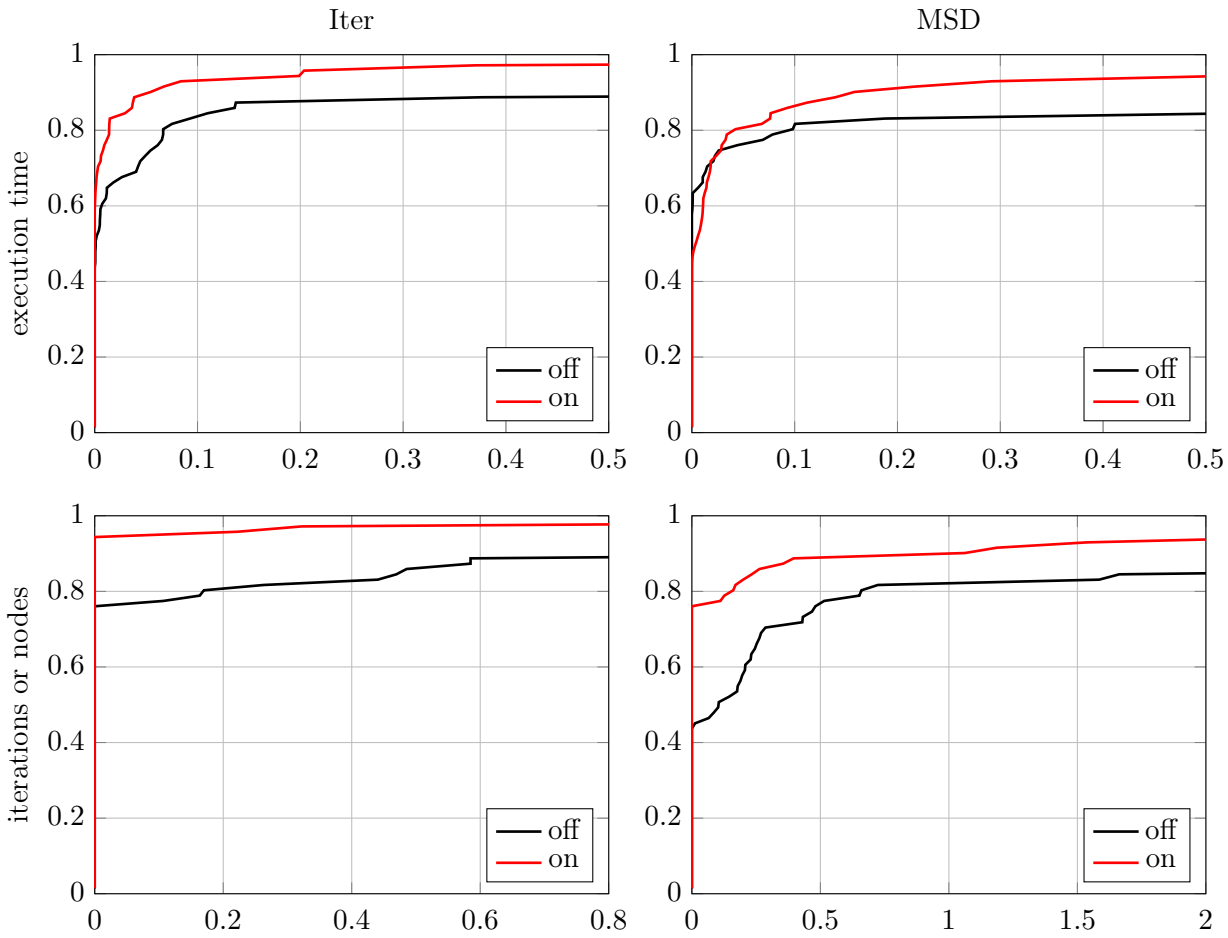


Table 4.4: $\mathcal{K}^*$ certificate cut scaling performance summary (larger δ).

		statuses				time (s)			subproblems			iters or nodes		
	scale	co	li	er	ex	aco	tco	all	aco	tco	all	aco	tco	all
Iter	off	63	1	28	3	4.5	4.4	6.6	5.2	5.0	4.2	5.2	5.1	4.4
	on	69	1	22	3	4.4	5.2	6.7	4.9	4.9	3.9	5.0	5.0	4.0
MSD	off	60	0	30	5	2.7	2.8	3.2	12.4	14.5	12.8	193	240	366
	on	67	0	26	2	2.9	4.0	3.9	11.9	15.8	12.1	188	392	393

Figure 4.7: $\mathcal{K}^*$ certificate cut scaling performance profiles.



Chapter 5

Formulations and oracles for mixed-integer conic optimization

Abstract

We introduce our open source MI-conic solver *MOIPajarito*, the new MathOptInterface-based successor to Pajarito. In addition to several new algorithmic features not available in Pajarito, MOIPajarito has a generic cone interface inspired by that of Hypatia, allowing it to be easily extended for new cones. In this chapter, we describe our implementations of the three broad classes of cones introduced in Chapter 2. Our $\mathcal{K}^*$ cut techniques enable MOIPajarito to solve natural formulations of a wide variety of applied examples. We also describe extended formulations (EFs) for vector spectral function cones that can accelerate MOIPajarito by tightening polyhedral relaxations. Finally, we develop efficient MI-conic formulations for homogenized piecewise linear constraints, which enable formulating tight relaxations of common types of nonconvex constraints and disjunctions of piecewise linear constraints. One of our formulations is the first logarithmic-sized MI-convex formulation for a finite union of convex sets with different recession cones. Using conic EFs, we extend our MI-conic relaxation techniques to the high-dimensional nonconvex setting.

5.1 Introduction

We use the following general form for MI-conic problems, over variable $x \in \mathbb{R}^n$:

$$\inf_x \quad c'x : \tag{5.1a}$$
$$b - Ax = 0, \tag{5.1b}$$
$$h - Gx \in \mathcal{K}, \tag{5.1c}$$
$$x_i \in \mathbb{Z} \qquad \forall i \in [\![I]\!], \tag{5.1d}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^p$, and $h \in \mathbb{R}^q$ are vectors, $A : \mathbb{R}^n \to \mathbb{R}^p$ and $G : \mathbb{R}^n \to \mathbb{R}^q$ are linear maps, $I \leq n$, and $\mathcal{K} \subset \mathbb{R}^q$ is a Cartesian product $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K$ of proper cones. Relaxing the integrality constraints (5.1d), we recover Hypatia's primal conic form discussed in Section 1.4.1.

5.1.1 Solving mixed-integer conic problems

In Section 4.2, we present the first conic-duality-based branch-and-bound outer approximation (OA) algorithm for MI-conic optimization problems, and in Section 4.5 we describe our algorithmic implementations in Pajarito.jl. Pajarito is succeeded by our new open source MI-conic solver MOIPajarito.jl, which we introduce here in Section 5.2.1. Like Pajarito, MOIPajarito uses an external MIP solver to solve polyhedral relaxations and an external continuous conic solver to solve primal-dual conic subproblems that provide feasible solutions and $\mathcal{K}^*$ cuts for refining the polyhedral relaxations.

A key feature of MOIPajarito is its generic cone interface, inspired by that of Hypatia. This enables the user to add support for new convex cones through the implementation of a small list of oracles, which we list in Section 5.2.2. The oracles for initial cuts, subproblem cuts, and separation cuts add $\mathcal{K}^*$ cuts directly to the MIP solver's OA model. In Section 5.2.4, we discuss optional oracles that enable internal support for extended formulations (EFs), generalizing the approach we describe for the second order cone EF Section 4.6.3.

In addition to the standard second order, PSD, and three-dimensional exponential and power cones (see Section 2.2), we predefine around a dozen cones (and their dual cones) through MOIPajarito's cone interface. These cones are recognized by Hypatia and belong to the three broad classes introduced in Chapter 2. In this chapter, we describe our $\mathcal{K}^*$ cut oracle implementations for these classes. For each class, we implement MI-conic formulations for applied examples. We perform computational testing to compare MOIPajarito's performance under natural formulations (NFs) and equivalent standard conic EFs.

In Section 5.3, we consider the PSD slice cones from Section 2.2.1, such as polynomial weighted sum-of-squares (SOS) cones and sparse PSD-completable matrix cones. In Section 5.4, we discuss the infinity/spectral norm cones from Section 2.2.2, particularly the matrix spectral and nuclear norm epigraph cones. We formulate four examples over the SOS and PSD-completable cones in Section 5.3.3 and three examples over the spectral/nuclear norm cones in Section 5.4.4. Our computational results show that MOIPajarito solves the NFs much more efficiently and reliably than it solves the EFs over PSD cones.

Finally in Section 5.5, we handle the spectral function cones from Section 2.2.3. For geometric mean cones and vector separable spectral function cones, we describe how MOIPajarito optionally manages EFs internally, by converting dual solutions and rays from the conic subproblems in the NF space to $\mathcal{K}^*$ cuts for the OA model in the EF space. We illustrate the impact of these EFs on the strength of polyhedral relaxations. We formulate four spectral function cone examples in Section 5.5.4. Our results demonstrate that the internally-managed EFs for the geometric mean cones and vector separable spectral function cones can reduce MOIPajarito's iteration counts and solve times significantly. For the matrix domain spectral function cones, MOIPajarito generally solves the NFs faster and more reliably than it solves the much larger EFs (which are not internally managed due to their complexity).

5.1.2 Advanced mixed-integer conic formulations

In Section 5.6, we develop MI-conic formulations for disjunctions and relaxations of common types of nonconvex constraints. We start by discussing tight MI-conic formulations for finite unions of convex sets. Then, for a nonconvex constraint involving the graph of a univariate convex function, we describe a simple sandwiching relaxation expressible with convex constraints and SOS2 (special ordered sets of type two) constraints. We extend this technique for the three-dimensional graph of the perspective function of a univariate convex function. This approach homogenizes standard univariate piecewise linear (PWL) formulations, utilizing the conic structure of the perspective to obtain simpler and smaller representations than can be achieved with a naive bivariate discretization.

We describe MISOCP and MILP formulations for SOS2 constraints, specializing these for the homogenized PWL setting. One of our formulations is the first MI-convex formulation to use only a logarithmic number of binary variables to represent a finite union of convex sets with different recession cones. The homogenized PWL formulations allow us to write tight formulations of disjunctions of PWL constraints, or MI-conic relaxations of disjunctions of nonconvex constraints.

By adapting conic EFs to the nonconvex setting, we extend these relaxation techniques to higher dimensions. This allows us to build tight and efficient MI-conic relaxations of common types of multivariate nonconvex constraints, for example a norm equality $u = \|w\|$. In Section 5.6.5, we describe three examples involving nonconvex constraints or disjunctions over convex or nonconvex constraints, and we perform computational testing to compare formulations.

5.2 MOIPajarito and cone oracles

5.2.1 Software architecture

In Section 4.5, we introduce Pajarito, our first open source MI-conic solver. Pajarito was written in Julia for MathProgBase, a low-level solver interface that has since been superseded by the significantly more powerful MathOptInterface (MOI) (Benoit Legat et al., 2020). The development of MOI was in part motivated by the desire for much more general conic modeling. Whereas MathProgBase and Pajarito only support the standard nonnegative, (rotated) second order, exponential, and PSD cones, MOI supports many more predefined cones and allows users or packages to define new cones (as subtypes of MOI.AbstractVectorSet). Users can model optimization problems over these cones using the convenient high-level modeling language JuMP (Dunning, Huchette, and Lubin, 2017). In March of 2022, MOI version 1.0 and JuMP version 1.0 were released.

The maturation of MOI has prompted us to replace Pajarito with a more powerful MOI-based successor, designed and written from scratch. We introduce two new Julia packages: MOIPajarito.jl, which is our new MI-conic solver, and PajaritoExtras.jl, which extends MOIPajarito to support more cones.[1] Like Pajarito, MOIPajarito uses external MIP and continuous conic solvers through

[1] Currently, these packages are available at `https://github.com/chriscoey/MOIPajarito.jl` and `https://github.com/chriscoey/PajaritoExtras.jl`. These package names and URLs may change in future, but they can be located from the author's GitHub page at `https://github.com/chriscoey`.

the MOI's solver-independent interface.

MOIPajarito has iterative and MIP-solver-driven (single tree) algorithms, conic subproblem-based and separation-based variants, and a similar set of algorithmic options to Pajarito. MOIPajarito also has some algorithmic features not available in Pajarito, such as support for SOS1/SOS2 constraints (currently limited to separation-based algorithms), separation $\mathcal{K}^*$ cuts for rays from an unbounded continuous relaxation, the ability to perform optimization-based separation (discussed in Section 5.2.3), and the ability to pass to the OA (MIP) solver any cones it directly supports (allowing nonpolyhedral OA).

Inspired by Hypatia's generic cone interface, MOIPajarito is easily extended to support new convex cones through the implementation of a small list of oracles. In MOIPajarito itself, we have predefined the standard second order and PSD cones and three-dimensional exponential and power cones. In the extension package PajaritoExtras, we add support for a dozen cones that are recognized by Hypatia. For most of these cones, we also support their dual cones. Since Hypatia is the only continuous conic solver that supports such a wide variety of cones, when using PajaritoExtras, MOIPajarito must be used with Hypatia. In the examples folder of PajaritoExtras, we use these new cones to model a dozen applied MI-conic examples, each with multiple formulation variants.

5.2.2 Cut oracles

We now describe the main cut oracles to be implemented through MOIPajarito's generic cone interface for a given cone $\mathcal{K}$. Most of these oracles directly add $\mathcal{K}^*$ cuts to the MIP solver's OA model. First, the initial cuts oracle adds a set of $\mathcal{K}^*$ cuts that define a fixed initial polyhedral relaxation of $\mathcal{K}$. We discuss initial fixed cuts in Section 4.4.2. Second, the subproblem cuts oracle takes as input a point $z \in \mathcal{K}^*$, which MOIPajarito obtains from a dual solution or ray found by the continuous conic solver, and adds $\mathcal{K}^*$ cuts that are at least as strong as the cut from z itself. We discuss $\mathcal{K}^*$ cut strengthening and extreme ray decompositions in Section 4.4.1.

Third, the separation cuts oracle takes as input a point s and checks whether $s \in \mathcal{K}$; this feasibility check is approximate and uses a feasibility tolerance (specified as an option to MOIPajarito). If s is significantly infeasible, the oracle adds at least one separation $\mathcal{K}^*$ cut that makes s infeasible. Separation cuts are discussed in Section 4.4.3; although they are not needed in theory for a finite-convergent algorithm, in practice they are often useful when subproblem cuts alone are not sufficient to tighten the objective gap to the desired optimality tolerances (due to numerical inexactness of the conic and MIP solvers). In a separation-only algorithm, MOIPajarito does not solve conic subproblems or add subproblem $\mathcal{K}^*$ cuts. Rather it simply separates infeasible points found by the OA solver until the OA solution no longer significantly violates the conic constraints. Note the separation-only algorithm is not known to be finitely convergent in general.

5.2.3 Optimization-based separation

For some cones, we do not know an analytic procedure for checking feasibility of a point or obtaining separation cuts. In such cases, we can use optimization. Suppose we have a cone $\mathcal{K}$ and a point

$\bar{s} \neq 0$ (since the origin is always feasible for $\mathcal{K}$). We set up the following simple separation problem:

$$\min \quad 0 : \tag{5.2a}$$

$$\bar{s} \in \mathcal{K}. \tag{5.2b}$$

Clearly, this problem (which has no variables or primal equalities) is feasible if and only if $\bar{s}$ is feasible. The conic dual of this problem is:

$$\max_z \quad -\bar{s}'z : \tag{5.3a}$$

$$z \in \mathcal{K}^*. \tag{5.3b}$$

This dual can be interpreted as finding a point z in $\mathcal{K}^*$ maximizing the violation on a potential separation cut $\bar{s}'z \geq 0$. Note $z = 0$ is feasible for the dual, so if the dual has an optimal solution, its value is nonnegative. If the primal problem (5.2b) is feasible, then its objective value is zero, hence the dual has an optimal value of zero also. If the primal is infeasible, then the dual has an improving ray $z \in \mathcal{K}^*$ with $\bar{s}'z < 0$. Therefore $s'z \geq 0$ is a $\mathcal{K}^*$ cut that is violated by the infeasible point $\bar{s}$.

Hypatia is an ideal tool for solving this primal-dual separation problem. This is our approach for sparse PSD-completable cones and primal SOS cones, as discussed in Section 5.3.2. Since there are no primal variables, Hypatia's QR-Cholesky linear system solver needs no matrix factorizations besides Hessian factorizations (if no closed form inverse Hessian product is available); see Section 1.6. The Hypatia model/solver object is constructed once for each unique cone, and whenever we need to solve the separation problem for a new point $\bar{s}$, we simply update the conic constraint constant vector h in-place (using Hypatia's model modification interface). Generally, Hypatia solves these problems so quickly and reliably that optimization-based separation does not become a bottleneck in MOIPajarito.

5.2.4 Extended formulations

Tawarmalani and Sahinidis (2005) and Hijazi, Bonami, and Ouorou (2014) observe that, for smooth mixed-integer nonlinear problems, simple EFs that exploit separability can be used to accelerate OA algorithms. Intuitively, EFs can tighten polyhedral relaxations, because a polynomial number of cuts in EF space may project to an exponential number of cuts in NF space. Vielma, Dunning, et al. (2017) show that using a conic EF for the second order cone can greatly improve MISOCP solver performance.

In Section 4.6.3, we discuss how Pajarito optionally manages the second order cone EF in the OA model while still using the NF in the continuous conic subproblems, to avoid slowing down the conic solver. This requires extending primal solutions and $\mathcal{K}^*$ cuts from the NF space of the conic subproblem to the EF space of the OA model. MOIPajarito generalizes this approach, allowing any cone to be associated with an EF by specifying EF oracles through the generic cone interface. Currently, PajaritoExtras only internally manages EFs that can be written in terms of linear constraints and multiple three-dimensional cones. We have predefined the second order cone

EF through this interface, as well as EFs for geometric mean cones and vector separable spectral function cones, which we discuss in Section 5.5.

To associate a cone $\mathcal{K}$ with an EF that will be internally managed by MOIPajarito, two cone oracles must be defined. The auxiliary variables oracle adds the appropriate auxiliary variables to the OA model, stores these in a data structure associated with the cone, and adds any linear constraints associated with the EF. The primal solution extender oracle takes an NF space point $s \in \mathcal{K}$ obtained by the continuous conic solver during a subproblem, extends it to a feasible solution in EF space, and sets the full solution as an OA model warm-start (for the iterative algorithm in Section 4.5.2.2) or heuristic callback solution (for the MIP-solver-driven algorithm in Section 4.5.2.3). Furthermore, the three $\mathcal{K}^*$ cut oracles (initial cuts, subproblem cuts, and separation cuts) described in Section 5.2.2 are modified to extend any cuts from the NF space into the EF space.

5.2.5 Computational testing setup

Using Julia and JuMP, we build instances of our examples in the PajaritoExtras repository. These instances enable computational testing of MOIPajarito to compare performance under different formulations or algorithmic options. For simplicity, all example instances are feasible and bounded and are randomly generated on-the-fly (using random seeds to ensure reproducibility).

We use MOIPajarito's iterative algorithm rather than the MIP-solver-driven (single-tree) algorithm. The basic implementations of these algorithms are described in Section 4.5.2.2. For initial fixed cuts, we make MOIPajarito only add variable bound constraints. These options allow us to directly compare (across different formulations or algorithmic options) MOIPajarito's iteration counts, which are a fairly reliable measure of the strength/quality of polyhedral relaxations.

Our scripts for running these benchmarks and analyzing raw results are available in the PajaritoExtras repository. We perform all computational experiments and results analysis on dedicated hardware with an AMD Ryzen 9 3950X 16-core processor (32 threads) and 128GB of RAM. We use Ubuntu 21.10, Julia 1.9, MOIPajarito 0.0.0, and PajaritoExtras 0.0.0. For MOIPajarito, we set Hypatia 0.7.0 as the continuous conic solver and Gurobi 9.5 as the MIP OA solver.

We use a time limit of 600 seconds for all instances. We set MOIPajarito's feasibility tolerance to 10^{-7}, relative optimality gap tolerance to 10^{-5}, and absolute optimality gap tolerance to 10^{-4}. We set various feasibility and optimality gap tolerances for Gurobi to 10^{-9} and for Hypatia to between 10^{-8} and 10^{-10}. For instances with very sparse constraint matrices, Hypatia uses the sparse symmetric indefinite linear system solver technique, otherwise it uses the dense QR-Cholesky technique with extensive preprocessing (see Section 1.6).

Across equivalent instances, we compare MOIPajarito's termination statuses, iteration counts (or MIP solver node counts, where specified), and solve times in seconds. These solver statistics are reported in tables, under the columns *st*, *it*, and *time* respectively. In these tables, asterisks indicate missing data, and we use the following codes for the termination status columns:

co: the solver claims it has an approximate optimality certificate,

tl: the solver stops itself due to a solve time limit of 600 seconds,

er: the solver encounters a numerical convergence error.

For each instance, the analysis script verifies that objective values approximately match between each pair of corresponding solve runs with *co* status codes. For most examples, we also verify the solutions are approximately feasible.

5.3 Positive semidefinite slice cones

In Section 2.5 we discussed the PSD slice cones. Consider a cone $\mathcal{K}$ that can be characterized as an intersection of slices of the PSD cone:

$$\mathcal{K} \coloneqq \{s \in \mathbb{R}^q : \Lambda_l(s) \succeq 0, \forall l \in [\![r]\!]\}, \tag{5.4}$$

where $\Lambda_l : \mathbb{R}^q \to \mathbb{S}^{o_l}, \forall l \in [\![r]\!]$ are linear operators. We note that for $\mathcal{K}_\succeq$ (the self-dual vectorized PSD cone), we have $r = 1$ and $q = \text{sd}(o)$, $\Lambda(s) = \text{mat}(s)$, and $\Lambda^*(S) = \text{vec}(S)$. The dual cone is:

$$\mathcal{K}^* \coloneqq \left\{s \in \mathbb{R}^q : \exists S_1, \ldots, S_r \succeq 0, s = \textstyle\sum_{l \in [\![r]\!]} \Lambda_l^*(S_l)\right\}, \tag{5.5}$$

where $\Lambda_l^* : \mathbb{S}^{o_l} \to \mathbb{R}^q$ is the adjoint linear operator of Λ_l. $\mathcal{K}$ and $\mathcal{K}^*$ belong to our PSD slice cone class.

We describe Pajarito's implementation of the real PSD cone $\mathcal{K}_\succeq$ itself in Section 4.6.4 (though with different notation). These techniques generalize easily for the complex Hermitian PSD cone $\mathcal{K}_{c\succeq}$, which is implemented in MOIPajarito. In this section, we further generalize for cones of the form (5.4), such as the sparse PSD cone $\mathcal{K}_{\text{sPSD}}$ defined in Section 2.2.1.2, and for cones of the form (5.5), such as the polynomial weighted sum-of-squares (SOS) cone $\mathcal{K}_{\text{SOS}}$ defined in Section 2.2.1.3. Recall for $\mathcal{K}_{\text{SOS}}$ parametrized by the collection of matrices $P_l \in \mathbb{R}^{d \times o_l}, \forall l \in [\![r]\!]$, we have:

$$\Lambda_l(s) = P_l' \,\text{Diag}(s) P_l, \tag{5.6}$$

$$\Lambda_l^*(S_l) = \text{diag}(P_l \Theta_l P_l'). \tag{5.7}$$

5.3.1 Initial fixed cuts

First consider a constraint $s \in \mathcal{K}$, where $\mathcal{K}$ is the primal cone (5.4). Since $\Lambda \succeq 0$ implies $\text{diag}(\Lambda) \geq 0$, we can add the $\sum_l o_l$ initial cuts:

$$\Lambda_{l,i,i}(s) \geq 0 \qquad \forall l \in [\![r]\!], i \in [\![o_l]\!]. \tag{5.8}$$

A stronger relaxation of $\Lambda \succeq 0$ is the dual cone of the diagonally dominant matrices (Ahmadi and Hall, 2015). This gives $\sum_l o_l(o_l - 1)$ additional initial cuts:

$$\Lambda_{l,i,i}(s) + \Lambda_{l,j,j}(s) \geq \pm 2\Lambda_{l,i,j}(s) \qquad \forall l \in [\![r]\!], i, j \in [\![o_l]\!] : i < j. \tag{5.9}$$

For $\mathcal{K}_{\text{sPSD}}$ of side dimension d, all diagonal elements are present in the sparsity pattern, so we can impose all d diagonal nonnegativity constraints (5.8). For each off-diagonal element in the sparsity pattern, we can impose the two cuts (5.9). For $\mathcal{K}^*_{\text{SOS}}$, the constraints (5.8) for example have the simple form:

$$\textstyle\sum_{j\in[\![q]\!]} P^2_{l,j,i} s_j \geq 0 \qquad \forall l \in [\![d]\!], i \in [\![o_l]\!]. \tag{5.10}$$

Now consider a constraint $s \in \mathcal{K}^*$, where $\mathcal{K}^*$ is the dual cone (5.5). The initial cuts are more ad hoc, depending on the structure of Λ^*. For $\mathcal{K}^*_{\text{sPSD}}$, which is the cone of PSD-completable matrices with the given sparsity pattern, we can actually impose the same set of initial cuts as for $\mathcal{K}_{\text{sPSD}}$ (diagonal nonnegativity and two cuts for each off-diagonal in the pattern), since these must be satisfied by any PSD matrix. For $\mathcal{K}_{\text{SOS}}$, we can impose the q initial cuts $s \geq 0$, since the dual cone representation shows that s is a sum of diagonals of PSD matrices. This makes intuitive sense as s represents the evaluations of the polynomial at the q interpolant points within the domain of the nonnegativity constraint.

5.3.2 Separation cuts

Consider a constraint $s \in \mathcal{K}$. Given a point $\bar{s} \in \mathbb{R}^q$, for each $l \in [\![r]\!]$ we form $\Lambda_l(\bar{s})$ and compute its symmetric/Hermitian eigendecomposition:

$$\Lambda_l(\bar{s}) = V_l \operatorname{Diag}(\sigma_l) V_l' = \textstyle\sum_{i\in[\![o_l]\!]} \sigma_{l,i} v_{l,i} v_{l,i}', \tag{5.11}$$

where the square eigenvector matrix V_l is orthonormal and the eigenvalues $\sigma_l \in \mathbb{R}^{o_l}$ are real. If $\sigma_l \geq 0, \forall l$, then by (5.4) $\bar{s} \in \mathcal{K}$, so there are no separation cuts to add. Otherwise, for each $l \in [\![r]\!]$, we add one separation cut for each negative eigenvalue:

$$\langle v_{l,i} v_{l,i}', \Lambda_l(s) \rangle \geq 0 \qquad \forall i \in [\![o_l]\!] : \sigma_{l,i} < 0, \tag{5.12}$$

which is violated by the infeasible point $\bar{s}$ because:

$$\langle v_{l,i} v_{l,i}', \Lambda_l(\bar{s}) \rangle = \textstyle\sum_{k\in[\![o_l]\!]} \sigma_{l,k} \langle v_{l,i} v_{l,i}', v_{l,k} v_{l,k}' \rangle = \sigma_{l,i} < 0. \tag{5.13}$$

For $\mathcal{K}_{\text{sPSD}}$, the extreme separation cuts (5.12) are simply the usual eigenvector cuts for $\mathcal{K}_{\succeq}$ projected onto the sparsity pattern. For $\mathcal{K}^*_{\text{SOS}}$, for each $l \in [\![r]\!]$, we perform an eigendecomposition of Λ_l and compute the matrix $Q_l = P_l V_l$, where the columns of V_l are the eigenvectors corresponding to negative eigenvalues. The extreme separation cuts (5.12) then have the simple form:

$$\textstyle\sum_{j\in[\![q]\!]} Q^2_{l,j,i} s_j \geq 0 \qquad \forall l \in [\![d]\!], i \in [\![o_l]\!] : \sigma_{l,i} < 0. \tag{5.14}$$

For a dual cone constraint $s \in \mathcal{K}^*$, separation cuts are often harder to derive. In general, we can use optimization-based separation as described in Section 5.2.3. This is our approach for $\mathcal{K}_{\text{SOS}}$ and $\mathcal{K}^*_{\text{sPSD}}$.

5.3.3 Examples for dual sparse positive semidefinite and sum-of-squares cones

5.3.3.1 Positive semidefinite completable matrix

Given a symmetric sparsity pattern with some sparse entries unknown, we choose integer values for a subset of the unknowns such that the resulting sparse symmetric matrix is PSD-completable. We maximize the minimum eigenvalue of any symmetric completion and restrict the unknown entries to lie in an interval $[-M, M]$. Let d be the side dimension, $S \in \mathbb{S}^d$ be a sparse matrix containing the known entries, and $\mathcal{S}$ be the sparsity pattern of the completable matrix (including diagonal entries). The variable $x \in \mathbb{R}^n$ represents the unknown entries in the lower triangle, and we introduce an auxiliary variable $y \in \mathbb{R}$. Recall $\mathcal{K}^*_{\text{sPSD}}(\mathcal{S})$ is the cone of PSD-completable matrices given the sparsity pattern $\mathcal{S}$. The conic form model is:

$$\begin{aligned}
\max_{x,y} \quad & y : && \text{(5.15a)}\\
& Me - x \in \mathbb{R}^n_{\geq}, && \text{(5.15b)}\\
& Me + x \in \mathbb{R}^n_{\geq}, && \text{(5.15c)}\\
& \text{vec}(S + \text{mat}(x) - yI(d)) \in \mathcal{K}^*_{\text{sPSD}}(\mathcal{S}), && \text{(5.15d)}\\
& x \in \mathbb{Z}^n, && \text{(5.15e)}
\end{aligned}$$

where the mat operator maps the vector x to its corresponding sparse matrix, and the vec operator maps the sparse matrix with pattern $\mathcal{S}$ to a vector of its nonzero lower triangle entries (note these are simple linear transformations). The standard conic EF for the $\mathcal{K}^*_{\text{sPSD}}$ constraint (5.15d) is described in Section 2.5.3; it uses one $\mathcal{K}_{\geq}$ cone of dimension $\text{sd}(d)$ and introduces $\text{sd}(d) - |\mathcal{S}|$ auxiliary variables corresponding to the zeros in the sparsity pattern $\mathcal{S}$.

We generate random instances of (5.15) for various side dimensions d. First, we generate the sparsity pattern $\mathcal{S}$ from a matrix with sparsity factor approximately $d^{-1/2}$, and we include all diagonal elements. Then we select approximately 70% of the elements in $\mathcal{S}$ to be known elements. Our approach ensures that all instances are feasible and bounded. Our results are summarized in Table 5.1. The NF converges on all instances up to size $d = 55$ within the time limit, but the EF hits a time limit or encounters numerical convergence difficulties on all instances larger than $d = 30$.

5.3.3.2 Polynomial facility location

We formulate a capacitated facility location problem with nonnegative polynomial flows over continuous time, from time $t = 0$ to $t = 1$. Suppose we have facilities $i \in [\![n]\!]$ and customers $j \in [\![m]\!]$. The fixed cost of opening facility i is $f_i \in \mathbb{R}_{\geq}$ and its maximum output rate is $u_i \in \mathbb{R}_{\geq}$. The binary variable x_i is one if and only if facility i is opened. The demand rate of customer j is $d_j \in \mathbb{R}_{\geq}$ and the cost per unit of flow from i to j is $c_{i,j} \in \mathbb{R}_{\geq}$. Customers can be served by multiple facilities. Each polynomial variable $y_{i,j} \in \mathbb{R}_{1,d}[t]$ (i.e. the ring of univariate polynomials in t of maximum degree d) represents the flow rate from i to j over $t \in [0, 1]$. We seek to minimize total cost (of opening facilities and shipping units of flow on arcs) while satisfying the capacity limit of each facility and

Table 5.1: Positive semidefinite completable matrix solver statistics.

	nat			ext		
d	st	it	time	st	it	time
10	co	0	0.0	co	0	0.0
15	co	7	2.8	co	7	1.6
20	co	27	1.5	co	47	5.9
25	co	66	5.9	tl	47	600
30	co	68	9.6	co	65	19
35	co	70	19	er	45	36
40	co	114	38	er	64	67
45	co	1026	550	tl	15	600
50	co	223	174	er	32	60
55	co	462	443	tl	20	600
60	tl	467	600	tl	26	600
65	tl	384	600	er	32	194

the demand of each customer. The high-level mixed-integer polynomial model is:

$$\min_{x,y} \quad f'x + \sum_{i\in[\![n]\!],j\in[\![m]\!]} c_{i,j}\int_0^1 y_{i,j}(t)\,\mathrm{d}t : \tag{5.16a}$$

$$x \in \{0,1\}^n, \tag{5.16b}$$

$$\sum_{j\in[\![m]\!]} y_{i,j}(t) \leq u_i x_i \qquad \forall i \in [\![n]\!], t \in [0,1], \tag{5.16c}$$

$$\sum_{i\in[\![n]\!]} y_{i,j}(t) \geq d_j \qquad \forall j \in [\![m]\!], t \in [0,1], \tag{5.16d}$$

$$y_{i,j}(t) \geq 0 \qquad \forall i \in [\![n]\!], j \in [\![m]\!], t \in [0,1]. \tag{5.16e}$$

Since nonnegativity of univariate polynomials over basic semialgebraic domains is equivalent to membership in a $\mathcal{K}_{\mathrm{SOS}}$ cone, we can formulate an exact mixed-integer conic representation of (5.16). Similar to the three SOS examples in Section 2.3 and following the approach in Papp and Yıldız (2019), we select $U = \binom{1+d}{1} = 1 + d$ interpolant basis points in $[0,1]$. To parametrize $\mathcal{K}_{\mathrm{SOS}(P)}$, we set up the collection of matrices P as described in Section 2.3.5. From the interpolation points and the domain $[0,1]$, we compute a vector of quadrature weights $w \in \mathbb{R}^U$, which allow us to express the integrals in the objective linearly. The new variables $y_{i,j} \in \mathbb{R}^U, \forall i \in [\![n]\!], j \in [\![m]\!]$ represent the values of the polynomials at the U interpolant basis points (from which we can perform a linear transformation using a Vandermonde matrix to obtain coefficients on the degree d monomial basis). The conic form model is:

$$\min_{x,y} \quad f'x + \sum_{i\in[\![n]\!],j\in[\![m]\!]} c_{i,j} w' y_{i,j} : \tag{5.17a}$$

$$u_i x_i e - \sum_{j\in[\![m]\!]} y_{i,j} \in \mathcal{K}_{\mathrm{SOS}(P)} \qquad \forall i \in [\![n]\!], \tag{5.17b}$$

$$\sum_{i\in[\![n]\!]} y_{i,j} - d_j e \in \mathcal{K}_{\mathrm{SOS}(P)} \qquad \forall j \in [\![m]\!], \tag{5.17c}$$

$$y_{i,j} \in \mathcal{K}_{\mathrm{SOS}(P)} \qquad \forall i \in [\![n]\!], j \in [\![m]\!], \tag{5.17d}$$

$$x \in \{0,1\}^n. \tag{5.17e}$$

The standard conic EF for each of the $m+n+mn$ $\mathcal{K}_{\mathrm{SOS}(P)}$ constraints is described in Section 2.5.2; it uses multiple $\mathcal{K}_{\geq}$ cones and introduces auxiliary variables. Note that whereas in the continuous polynomial minimization example in Section 2.3.5 we could use a conic dual formulation with fewer primal variables, we cannot take that approach here because of the integrality constraints.

We generate random instances of (5.17) with polynomial degree $d = 6$, varying the number of facilities n and letting the number of customers be $m = 2n$. First, we generate random values for the parameters, then we solve small polynomial minimization problems (using Hypatia) to determine how much to increase the values of the capacity limits u_i to allow total demand to be met. This ensures that the instances are feasible and bounded. Our results are summarized in Table 5.2. The NF converges on all instances up to size $n = 40$ within the time limit, but the EF hits a time limit or encounters numerical convergence difficulties on all instances. Note that very few iterations are needed to converge, typically around two. We verify feasibility of the solutions returned by solving a small polynomial minimization problem for each SOS constraint.

Table 5.2: Polynomial facility location solver statistics.

	nat			ext		
n	st	it	time	st	it	time
5	co	1	0.2	er	1	2.0
10	co	2	1.2	er	1	19
15	co	2	6.1	tl	4	600
20	co	2	20	tl	3	602
25	co	3	51	tl	2	611
30	co	3	125	tl	0	614
35	co	2	127	tl	0	603
40	co	3	332	tl	0	601
45	tl	4	604	*	*	*
50	tl	3	606	*	*	*

5.3.3.3 Polynomial two-stage stochastic problem

We have n different crops and m equal-sized plots of land for planting. For crop $i \in [\![n]\!]$, $a_i \geq 0$ is the per-plot fixed cost of planting, and $\xi_i \sim U(0,1)$ is the uncertain uniformly distributed per-plot yield in crop units. In the first stage we do not yet know ξ_i, but we must choose the number of plots to allocate to crop i, denoted by the variable $x_i \in \mathbb{Z}$. In the second stage, we harvest the crops and decide how many units of each crop to buy and sell on the market, balancing inputs and outputs. For each crop i, $d_i \geq 0$ is the contractual demand that must be met, $c_i \geq 0$ is the selling price per unit (excluding contractual demand), and $b_i \geq 0$ is the purchasing cost per unit. Since the second stage decisions decompose by crop $i \in [\![n]\!]$, we represent these variables as univariate polynomials in ξ_i of maximum degree $2k$. The purchasing quantity $y_i \in \mathbb{R}_{1,2k}[\xi_i]$ and the selling quantity $z_i \in \mathbb{R}_{1,2k}[\xi_i]$ must be nonnegative for any realized yield $\xi_i \in [0,1]$. We minimize the total

expected cost, so the high-level stochastic mixed-integer polynomial model is:

$$\min_{x,y,z} \quad a'x + \sum_{i\in[\![n]\!]} \mathbb{E}_{\xi_i}[b_i y_i(\xi_i) - c_i z_i(\xi_i)] : \tag{5.18a}$$

$$e'x \leq m, \tag{5.18b}$$

$$x \geq 0, \tag{5.18c}$$

$$x \in \mathbb{Z}^n, \tag{5.18d}$$

$$\xi_i x_i + y_i(\xi_i) - z_i(\xi_i) - d_i = 0 \qquad \forall i \in [\![n]\!], \xi_i \in [0,1], \tag{5.18e}$$

$$y_i(\xi_i) \geq 0 \qquad \forall i \in [\![n]\!], \xi_i \in [0,1], \tag{5.18f}$$

$$z_i(\xi_i) \geq 0 \qquad \forall i \in [\![n]\!], \xi_i \in [0,1]. \tag{5.18g}$$

Note that the expectations in the objective can be rewritten as integrals - for example, since $\xi_i \sim U(0,1)$, we have:

$$\mathbb{E}_{\xi_i}[y_i(\xi_i)] = \int_0^1 y_i(\xi_i) \mathrm{d}\xi_i. \tag{5.19}$$

We formulate an exact mixed-integer conic representation of (5.18). We set up $U = 1 + 2k$ interpolant basis points $o_u \in [0,1], \forall u \in [\![U]\!]$ and the collection of matrices P parametrizing $\mathcal{K}_{\mathrm{SOS}(P)}$. The new variables $y_i, z_i \in \mathbb{R}^U, \forall i \in [\![n]\!]$ represent the polynomials via the interpolant basis. Using the integral representation of the expectation (5.19), we can use the quadrature weights w to formulate the expectations in the objective as linear functions of y_i, z_i. The conic form model is:

$$\min_{x,y,z} \quad a'x + \sum_{i\in[\![n]\!]} w'(b_i y_i - c_i z_i) : \tag{5.20a}$$

$$o x_i + y_i - z_i - d_i e = 0 \qquad \forall i \in [\![n]\!], \tag{5.20b}$$

$$m - e'x \in \mathbb{R}^{\geq}, \tag{5.20c}$$

$$x \in \mathbb{R}^n_{\geq}, \tag{5.20d}$$

$$y_i \in \mathcal{K}_{\mathrm{SOS}(P)} \qquad \forall i \in [\![n]\!], \tag{5.20e}$$

$$z_i \in \mathcal{K}_{\mathrm{SOS}(P)} \qquad \forall i \in [\![n]\!], \tag{5.20f}$$

$$x \in \mathbb{Z}^n. \tag{5.20g}$$

The EF for the $2n$ $\mathcal{K}_{\mathrm{SOS}(P)}$ constraints in Section 2.5.2 uses $\mathcal{K}_{\geq}$ cones and auxiliary variables.

We generate random instances of (5.20) with $n = 3$ crops, varying the degree $2k$ of the polynomials. The instances are feasible since there is no capacity limit on the purchasing quantities. The instances are bounded because the purchasing costs exceed the selling costs, i.e. $0 < c_i < b_i$. Our results are summarized in Table 5.3. The NF converges in only one or two iterations on all instances up to half-degree $k = 512$ within the time limit, but the EF hits a time limit or encounters numerical convergence difficulties on all instances. We verify feasibility of the solutions returned by solving a small polynomial minimization problem for each SOS constraint.

We include the optimal objective values (obtained by *nat*) in the *obj* column. As expected, these values decrease as the polynomial half-degree k increases, with rapidly diminishing returns. If we had a problem with more crops, we could decrease k in order to balance the solve time.

Table 5.3: Polynomial two-stage stochastic problem solver statistics.

		nat			ext		
k	obj	st	it	time	st	it	time
8	16.73477	co	2	0.1	er	2	0.7
16	16.70516	co	2	0.2	er	2	4.1
32	16.69616	co	2	0.5	er	1	67
64	16.69364	co	2	1.6	tl	0	868
128	16.69301	co	2	6.8	*	*	*
256	16.69284	co	1	33	*	*	*
512	16.69280	co	1	234	*	*	*
1024	*	tl	0	673	*	*	*

We note we could probably solve (5.20) faster by decomposing the continuous conic subproblems by crop, i.e. solving n smaller subproblems instead of one at each iteration. This Benders-like algorithm is not implemented in MOIPajarito currently. Our example code allows optionally adding a linking constraint across the crops in the second stage, in which case we let the polynomial variables be multivariate functions of $(\xi_i)_{i \in [\![n]\!]}$. This problem is not decomposable and is much higher-dimensional. In future could also try non-uniform or non-independent distributions for the crop yields, as sum-of-squares modeling is quite versatile.

5.3.3.4 Polynomial regression

Suppose we have m observations $(X_i, Y_i), \forall i \in [\![m]\!]$, where $X_i \in [0,1]^n$ and $Y_i \in [0,1]$. We assume there are two populations and we do not know which observations belong to which population. Our goal is to estimate a multivariate polynomial regression function for each population and assign the observations to these populations. We let $p_1, p_2 \in \mathbb{R}_{n,2k}[x]$ be the regressor variables. Suppose that p_1 is greater than or equal to p_2 on the domain of interest, $[0,1]^n$. We let $b \in \{0,1\}^m$ be binary variables representing the assignment of the observations to the populations, and we assume that the each population contains at least 40% of the observations. We minimize the ℓ_2 norm of the residuals $z \in \mathbb{R}^m$. The high-level disjunctive polynomial model is:

$$\begin{aligned}
\min_{p_1,p_2,z,b} \quad & \|z\| : && && \text{(5.21a)}\\
& p_1(x) \leq 1 && \forall x \in [0,1]^n, && \text{(5.21b)}\\
& p_1(x) \geq p_2(x) && \forall x \in [0,1]^n, && \text{(5.21c)}\\
& p_2(x) \geq 0 && \forall x \in [0,1]^n, && \text{(5.21d)}\\
& b \in \{0,1\}^m, && && \text{(5.21e)}\\
& 0.4m \leq e'b \leq 0.6m, && && \text{(5.21f)}\\
b_i = 1 \quad \Rightarrow \quad & z_i = Y_i - p_1(X_i) && \forall i \in [\![m]\!], && \text{(5.21g)}\\
b_i = 0 \quad \Rightarrow \quad & z_i = Y_i - p_2(X_i) && \forall i \in [\![m]\!]. && \text{(5.21h)}
\end{aligned}$$

Since the polynomials are multivariate, we formulate a mixed-integer sum-of-squares relaxation of (5.21). As in Sections 2.3.6 and 2.3.7, we select $U = \binom{n+2k}{n}$ interpolant points in $[0,1]^n$, compute the collection of matrices P, and compute a matrix $B \in \mathbb{R}^{m \times U}$ by evaluating the U Lagrange polynomials (corresponding to the interpolant basis) at the observations $X_i, \forall i \in [\![m]\!]$. The new variables $p_1, p_2 \in \mathbb{R}^U, \forall i \in [\![n]\!]$ represent the values of the polynomials at the interpolant basis points, or equivalently the coefficients on the Lagrange basis polynomials. The linear function $Bp_1 \in \mathbb{R}^m$ of the polynomial variable p_1 represents its values at $X_i, \forall i \in [\![m]\!]$ (and similarly for p_2). Note the values of the residuals are bounded below by -1 and above by 1. For each $i \in [\![m]\!]$, we use big-M formulations for the implication constraints:

$$|z_i - Y_i + Bp_1| \leq 1 - b_i, \tag{5.22a}$$
$$|z_i - Y_i + Bp_2| \leq b_i. \tag{5.22b}$$

We reformulate these big-M constraints using linear constraints and auxiliary variables $r_1, r_2 \in \mathbb{R}^m$. Rewriting the objective function using an epigraph variable γ and a $\mathcal{K}_{\ell_2}$ (second order) constraint, the conic form model is:

$$\min_{p_1,p_2,z,b,r_1,r_2,\gamma} \quad \gamma : \tag{5.23a}$$
$$r_{1,i} = z_i - Y_i + Bp_1 \qquad \forall i \in [\![m]\!], \tag{5.23b}$$
$$r_{2,i} = z_i - Y_i + Bp_2 \qquad \forall i \in [\![m]\!], \tag{5.23c}$$
$$(\gamma, z) \in \mathcal{K}_{\ell_2}, \tag{5.23d}$$
$$r_1 + (e - b) \in \mathbb{R}^m_{\geq}, \tag{5.23e}$$
$$(e - b) - r_1 \in \mathbb{R}^m_{\geq}, \tag{5.23f}$$
$$r_2 + b \in \mathbb{R}^m_{\geq}, \tag{5.23g}$$
$$b - r_2 \in \mathbb{R}^m_{\geq}, \tag{5.23h}$$
$$e'b - 0.4m \in \mathbb{R}_{\geq}, \tag{5.23i}$$
$$0.6m - e'b \in \mathbb{R}_{\geq}, \tag{5.23j}$$
$$e - p_1 \in \mathcal{K}_{\mathrm{SOS}(P)}, \tag{5.23k}$$
$$p_1 - p_2 \in \mathcal{K}_{\mathrm{SOS}(P)}, \tag{5.23l}$$
$$p_2 \in \mathcal{K}_{\mathrm{SOS}(P)}, \tag{5.23m}$$
$$b \in \{0,1\}^m. \tag{5.23n}$$

The EF for the three $\mathcal{K}_{\mathrm{SOS}(P)}$ constraints in Section 2.5.2 uses $\mathcal{K}_{\geq}$ cones and auxiliary variables.

We generate random instances of (5.23), varying the number of observations m from $2U = 30$ to 80. We use bivariate quartic polynomials (i.e. $n = 2, 2k = 4$, and $U = 15$), hence the sum-of-squares relaxation (5.23) is exact (Blekherman, 2012). First we generate two random underlying polynomials representing the two populations and satisfying the three sum-of-squares constraints in

(5.23) on p_1 and p_2 (using Hypatia to solve small polynomial minimization problems). Then we use these underlying polynomials to generate the m noisy observations, with approximately half of the observations from each population, using a signal to noise ratio of 25. The instances are feasible and bounded.

Our results are summarized in Table 5.4. Note that the $\mathcal{K}_{\ell_2}$ constraint is handled with OA in MOIPajarito rather than passed to Gurobi. The NF converges with a modest number of iterations on all instances up to $m = 70$ observations within the time limit. The EF is slower to solve for all sizes and solves fewer large instances within the time limit. We verify SOS-feasibility of the solutions. We note that, due to the large number of variables and linear inequalities in these formulations, Hypatia would be able to solve the subproblems faster if many of these were able to be eliminated using advanced preprocessing techniques that are not implemented.

Table 5.4: Polynomial regression solver statistics.

	nat			ext		
m	st	it	time	st	it	time
30	co	9	6.0	co	9	16
35	co	6	8.3	co	8	33
40	co	7	25	co	8	54
45	co	7	22	co	9	74
50	co	10	47	co	9	107
55	co	17	259	tl	12	600
60	co	8	67	co	10	249
65	co	9	223	tl	9	600
70	co	11	288	tl	9	600
75	tl	8	600	tl	5	600
80	tl	4	600	tl	4	600

5.4 Infinity/spectral norm cones

In Sections 2.2.2 and 2.6 we consider a class of cones that can be characterized as epigraphs of vector ℓ_∞ or ℓ_1 norms or matrix spectral or nuclear norms. For the real vector domain case, the polyhedral cones $\mathcal{K}_{\ell_\infty}$ and $\mathcal{K}^*_{\ell_\infty}$ have fairly compact (polynomial-sized) linear EFs, hence we do not need to use polyhedral OA. For complex vector domains, these cones are nonpolyhedral but have compact EFs using multiple three-dimensional $\mathcal{K}_{\ell_2}$ cones. In the case of symmetric/Hermitian or rectangular matrix domains, we do not use EFs for OA.

We focus here on describing oracles for the rectangular real matrix domain case. Recall from Section 2.2.2.2 that for d rows and $s \geq d$ columns, the spectral and nuclear norm cones are:

$$\mathcal{K}_{\ell_{\text{spec}}(d,s)} \coloneqq \{(u, w) \in \mathbb{R}_{\geq} \times \mathbb{R}^{ds} : u \geq \sigma_1(W)\}, \tag{5.24a}$$
$$\mathcal{K}^*_{\ell_{\text{spec}}(d,s)} \coloneqq \left\{(u, w) \in \mathbb{R}_{\geq} \times \mathbb{R}^{ds} : u \geq \textstyle\sum_{i \in [\![d]\!]} \sigma_i(W)\right\}, \tag{5.24b}$$

where $W \coloneqq \text{mat}_{d,s}(w) \in \mathbb{R}^{d \times s}$, $\sigma_i(W) \geq 0$ is the ith largest singular value of W. The complex

matrix case is nearly identical, and we can easily specialize these oracles for the symmetric/Hermitian cases (e.g by using eigendecompositions instead of singular value decompositions). The primal and dual cones are implemented in MOIPajarito for all six domain types: real/complex vectors, symmetric/Hermitian matrices, and real/complex rectangular matrices.

5.4.1 Initial cuts

For $\mathcal{K}_{\ell_{\text{spec}}}$ and $\mathcal{K}^*_{\ell_{\text{spec}}}$, we add the initial cut $u \geq 0$, which is valid for any norm-epigraph cone. We can use some of the cheap primal or dual feasibility check conditions discussed in Section 2.6.2 to derive more initial cuts. For example, we can use the scaled ℓ_1 and ℓ_∞ lower bounds on the spectral norm along with EFs for $\mathcal{K}_{\ell_\infty}$ and $\mathcal{K}^*_{\ell_\infty}$ to impose a polyhedral relaxation of a $\mathcal{K}_{\ell_{\text{spec}}}$ constraint. If the OA solver supports MISOCP, we can use the scaled ℓ_2 norm bounds on the spectral or nuclear norm to add a $\mathcal{K}_{\ell_2}$ constraint that gives a tighter relaxation of the $\mathcal{K}_{\ell_{\text{spec}}}$ or $\mathcal{K}^*_{\ell_{\text{spec}}}$ constraint, though this is not currently implemented.

5.4.2 Cut strengthening

For a spectral norm cone constraint $(u, w) \in \mathcal{K}_{\ell_{\text{spec}}}$, suppose we have a dual point $(p, q) \in \mathcal{K}^*_{\ell_{\text{spec}}}$. Let $Q = \text{mat}(q) \in \mathbb{R}^{d \times s}$ and similarly let $W = \text{mat}(w)$ be the matrix of variables. Note that $q'w = \langle Q, W \rangle$. Let $Q = U \,\text{Diag}(\sigma) V'$ be the thin SVD of Q, where $\sigma \in \mathbb{R}^r_{\geq}$ are the singular values (ordered largest to smallest) and $U \in \mathbb{R}^{d \times d}$, $V \in \mathbb{R}^{s \times d}$ are orthogonal matrices i.e. $UU' = U'U = V'V = I(d)$. The cut $pu + \langle Q, W \rangle \geq 0$ is valid, but it can be strengthened to:

$$e'\sigma u + \langle Q, W \rangle \geq 0, \tag{5.25}$$

since $(e'\sigma, q)$ is on the boundary of $\mathcal{K}^*_{\ell_{\text{spec}}}$. We can decompose this cut into multiple cuts to tighten the polyhedral OA:

$$\sigma_i u + \langle \sigma_i U_i V_i', W \rangle \geq 0 \qquad \forall i \in [\![d]\!] : \sigma_i > 0. \tag{5.26}$$

These cuts are from extreme rays of $\mathcal{K}^*_{\ell_{\text{spec}}}$ (analogous to the extreme points of ℓ_1 norm ball), and they imply the strengthened dual cut (5.25) because:

$$\textstyle\sum_{i \in [\![d]\!] : \sigma_i > 0} (\sigma_i u + \langle \sigma_i U_i V_i', W \rangle) = e'\sigma u + \langle Q, W \rangle. \tag{5.27}$$

Now suppose we have a nuclear norm cone constraint $(u, w) \in \mathcal{K}^*_{\ell_{\text{spec}}}$ and the dual point is $(p, q) \in \mathcal{K}_{\ell_{\text{spec}}}$. The cut $pu + \langle Q, W \rangle \geq 0$ is valid, but it can be strengthened to:

$$\sigma_1 u + \langle Q, W \rangle \geq 0, \tag{5.28}$$

since (σ_1, q) is on the boundary of $\mathcal{K}_{\ell_{\text{spec}}}$. This cut is not extreme in general, since the extreme rays of $\mathcal{K}_{\ell_{\text{spec}}}$ have equal $\sigma_i = \sigma_j, \forall i, j \in [\![d]\!]$ (analogous to the extreme points of the ℓ_∞ norm ball). Since an extreme ray decomposition may require up to 2^{d-1} extreme rays, we do not attempt this. We could optionally add both the non-extreme cut and a single extreme cut $\sigma_1 u + \sigma_1 \langle Q, UV' \rangle \geq 0$

(obtained by letting $\sigma = \sigma_1 e$), but this is not currently implemented.

5.4.3 Separation cuts

Suppose we have a point $(\bar{u}, \bar{w}) \in \mathbb{R}^q$ and let $\bar{W} = \text{mat}(\bar{w}) \in \mathbb{R}^{d \times s}$. Similar to above, let $\bar{W} = U \,\text{Diag}(\sigma) V'$ be the thin SVD of $\bar{W}$. For a spectral norm cone constraint, the point is infeasible if $\bar{u} < \sigma_1$, in which case we add the separation cuts:

$$u - \langle U_i V_i', W \rangle \geq 0 \qquad \forall i \in [\![d]\!] : \bar{u} < \sigma_i. \tag{5.29}$$

These extreme cuts are violated because $\langle U_i V_i', \bar{W} \rangle = \sigma_i$.

For a nuclear norm cone constraint, the point is infeasible if $\bar{u} < e'\sigma$, in which case we add the separation cut:

$$u - \langle UV', W \rangle \geq 0. \tag{5.30}$$

This extreme cut is violated because $\langle UV', \bar{W} \rangle = e'\sigma$.

5.4.4 Examples for spectral and nuclear norm cones

5.4.4.1 Matrix completion

We find nonnegative integer values for the missing entries of a partially-completed symmetric matrix to minimize the nuclear norm (maximum absolute value of the eigenvalues) of the completed symmetric matrix, subject to some linear constraints. Let $X \in \mathbb{S}^d$ be a sparse symmetric matrix of the missing entry variables, with sparsity pattern $\mathcal{S}$. Let $Y \in \mathbb{S}^d$ be a sparse symmetric matrix with the inverse sparsity pattern of $\mathcal{S}$ containing the known entries, so that $X + Y$ is the completed dense matrix. The high-level model is:

$$\begin{aligned}
\min_X \quad & \|X + Y\|_{\text{nuc}} : && && \text{(5.31a)}\\
& 0 \leq X_{i,j} \leq 9 && \forall (i,j) \in \mathcal{S}, && \text{(5.31b)}\\
& \textstyle\sum_{(i,j) \in \mathcal{S}} X_{i,j} \geq d, && && \text{(5.31c)}\\
& X_{i,j} \in \mathbb{Z} && \forall (i,j) \in \mathcal{S}. && \text{(5.31d)}
\end{aligned}$$

The equivalent conic formulation simply uses nonnegative cones for the linear constraints and introduces an epigraph variable for the objective along with a dual symmetric spectral norm cone ($\mathcal{K}^*_{\ell_{\text{sspec}}}$) constraint. Note we rescale $X + Y$ by $1/d$ to improve the scaling of the objective. The standard conic EF for $\mathcal{K}^*_{\ell_{\text{sspec}}}$ is described in Section 2.2.2; it uses two $\mathcal{K}_{\succeq}$ cones and introduces a large number of auxiliary variables.

We generate random instances of the conic formulation for (5.31) for various side dimensions d. First, we generate a random PSD matrix with entries between 0 and 9, then we choose a fraction of approximately $(n-1)/n$ of these entries to be the known values. All instances are feasible and

bounded, and our results are summarized in Table 5.5. The NF converges on all instances up to size $d = 180$ within the time limit, but the EF hits a time limit or encounters numerical convergence difficulties on all instances but one. Note that larger instances solve in one iteration.

Table 5.5: Matrix completion solver statistics.

	nat			ext		
d	st	it	time	st	it	time
20	co	59	4.1	er	4	0.7
40	co	110	25	er	3	8.3
60	co	38	16	er	2	29
80	co	14	25	tl	3	600
100	co	1	38	co	1	438
120	co	1	72	tl	0	607
140	co	1	152	tl	0	1095
160	co	1	282	tl	0	2448
180	co	1	475	tl	0	4867
200	tl	0	610	*	*	*
220	tl	0	618	*	*	*

5.4.4.2 Matrix decomposition

We aim to decompose a rectangular matrix $C \in \mathbb{R}^{m \times n}$ into a sum $C = A + B$ of a binary matrix A and matrix B with minimal spectral norm. The number of nonzeros in A must equal k. The high-level model is:

$$\min_A \quad \|C - A\|_{\text{spec}} : \tag{5.32a}$$

$$\textstyle\sum_{i \in [\![m]\!], j \in [\![n]\!]} A_{i,j} = k, \tag{5.32b}$$

$$A_{i,j} \in \{0, 1\} \qquad \forall i \in [\![m]\!], j \in [\![n]\!]. \tag{5.32c}$$

The equivalent conic formulation simply introduces an epigraph variable for the objective along with a $\mathcal{K}_{\ell_{\text{spec}}}$ constraint. The standard conic EF for this constraint is described in Section 2.2.2; it involves a large $\mathcal{K}_{\succeq}$ cone of side dimension $m + n$.

We generate random instances of the conic formulation for (5.32) for row dimension $m = 15$, $k \approx m$, and various column dimensions n. The parameter matrix C is a sum of a random sparse binary matrix and a low-rank random matrix. All instances are feasible and bounded, and our results are summarized in Table 5.6. The NF and the EF converge on all instances up to size $n = 600$ and $n = 200$ respectively, beyond which they hit time limits. The NF is faster and usually takes a similar number of iterations as the EF, so there is no evidence that the EF tightens the polyhedral relaxations for this example.

Table 5.6: Matrix decomposition solver statistics.

	nat			ext		
n	st	it	time	st	it	time
50	co	53	23	co	71	37
100	co	14	17	co	14	32
150	co	55	49	co	55	140
200	co	111	164	co	109	386
250	co	59	70	tl	38	600
300	co	82	232	tl	0	602
350	co	138	440	tl	0	612
400	co	1	370	tl	0	612
450	co	8	219	tl	0	889
500	co	16	290	tl	0	1326
550	co	4	370	tl	0	1915
600	co	5	471	*	*	*
650	tl	0	610	*	*	*
700	tl	0	601	*	*	*

5.4.4.3 Matrix regression

Given a design matrix $X \in \mathbb{R}^{n\times p}$ and a response matrix $Y \in \mathbb{R}^{n\times m}$, we estimate a coefficient matrix $A \in \mathbb{R}^{p\times m}$, where n is the number of samples, p is the number of predictors, and m is the number of responses, and $n \geq p \geq m$. We assume all entries of A are in $[-1, 1]$. We use the ℓ_2 norm loss, with nuclear norm regularization (with parameter $\lambda > 0$) to encourage a low-rank solution. Furthermore, we restrict the number of nonzero rows of A to be at most $k \leq p$, i.e. we enforce sparsity with respect to the predictors. In the context of multi-response linear regression, *group-LASSO-row* (or GLR) regularization is sometimes used to encourage row sparsity (L. Chen and Huang, 2012), but our approach uses discrete constraints rather than convex regularization. The high-level model is:

$$\min_{A,z} \quad \|\mathrm{vec}(Y - XA)\| + \lambda\|A\|_{\mathrm{nuc}} : \tag{5.33a}$$

$$z \in \{0,1\}^p, \tag{5.33b}$$

$$e'z \leq k, \tag{5.33c}$$

$$-z_i \leq A_{i,j} \leq z_i \qquad \forall i \in [\![p]\!], j \in [\![m]\!]. \tag{5.33d}$$

The equivalent conic formulation introduces two epigraph variables for the objective terms along with a $\mathcal{K}_{\ell_2}$ constraint for the loss and a $\mathcal{K}^*_{\ell_{\mathrm{spec}}}$ constraint for the regularization. Note that for $n > p$, we use a QR factorization of X to reduce the dimension of the $\mathcal{K}_{\ell_2}$ constraint from $1 + nm$ to $1 + pm$; see Section 2.3.7 for details. The standard conic EF for $\mathcal{K}^*_{\ell_{\mathrm{spec}}}$ is described in Section 2.2.2; it involves a large $\mathcal{K}_{\succeq}$ cone of side dimension $p + m$ and $\mathrm{sd}(p) + \mathrm{sd}(m)$ auxiliary variables.

We generate random instances of the conic formulation for (5.33) for various values of p, with $k \approx 2p/3$, $n = 2p$, and $m = 5$. All instances are feasible and bounded, and our results are summarized in Table 5.7. The NF converges on all but one instance up to size $p = 100$, but the EF only converges

on the smallest instance and hits the time limit or encounters numerical convergence issues otherwise. The NF only takes at most 5 iterations to converge.

Table 5.7: Matrix regression solver statistics.

	nat			ext		
p	st	it	time	st	it	time
10	co	2	0.1	co	7	1.5
20	co	2	0.5	er	3	2.3
30	co	2	0.9	tl	2	600
40	co	4	14	er	2	38
50	co	4	37	er	1	74
60	co	5	363	tl	2	600
70	co	2	8.9	tl	1	600
80	co	2	35	tl	1	600
90	tl	4	600	tl	0	631
100	co	2	68	tl	0	632
110	tl	2	600	tl	0	637

5.5 Spectral function cones

These cones are discussed in Section 2.2.3 and Chapter 3. For the vector domains, MOIPajarito optionally uses internally managed EFs to accelerate OA, as discussed in Section 5.2.4. We show how to extend $\mathcal{K}^*$ cuts and solutions from the NF space into the EF space. For the matrix domains, we do not use internally managed EFs, as these are large and complex.

5.5.1 Geometric mean cone

For convenience, let $\text{geo} : \mathbb{R}^d_{\geq} \to \mathbb{R}_{\geq}$ be the geometric mean function $\text{geo}(w) = \prod_{i \in [\![d]\!]} w_i^{1/d}$. Note in Chapter 3, we refer to this as the root-determinant function in the more general context of Jordan algebras. The geometric mean cone is the hypograph of the geometric mean (see Section 2.2.3.1):

$$\mathcal{K}_{\text{geo}} \coloneqq \left\{(u, w) \in \mathbb{R} \times \mathbb{R}^d_{\geq} : u \leq \text{geo}(w)\right\}, \tag{5.34a}$$

$$\mathcal{K}^*_{\text{geo}} \coloneqq \left\{(u, w) \in \mathbb{R}_{\leq} \times \mathbb{R}^d_{\geq} : u \geq -d\,\text{geo}(w)\right\}. \tag{5.34b}$$

Since $\mathcal{K}^*_{\text{geo}}$ does not provide additional modeling capability over $\mathcal{K}_{\text{geo}}$, we only support $\mathcal{K}_{\text{geo}}$ in MOIPajarito.

Suppose the constraint is $(u, w) \in \mathcal{K}_{\text{geo}}$. We note that the OA techniques below can be easily adapted for the more general hypograph of power mean cone (see Section 1.8), which is implemented in Hypatia but not MOIPajarito currently. Since using the EF is optional in MOIPajarito, first we show the cut techniques we use without the EF, then we extend these techniques for the EF.

For initial cuts, we add the d variable bounds $w \geq 0$. By plugging $w = e$ into the nonlinear inequality in the $\mathcal{K}^*_{\text{geo}}$ definition, we see that $(-d, e)$ is a point on the boundary of the dual cone,

which defines a simple initial cut $-du + e'w \geq 0$. Indeed, given any point $(\bar{u}, \bar{w}) \in \mathbb{R}^{1+d}$ with $\bar{w} > 0$, we can ignore $\bar{u}$ and obtain an extreme cut:

$$-d\,\text{geo}(\bar{w})u + \bar{w}'w \geq 0. \tag{5.35}$$

To separate a point $(\bar{u}, \bar{w}) \notin \mathcal{K}_{\text{geo}}$ that satisfies $\bar{w} > 0$, we use the gradient cut:

$$-u + d^{-1}\,\text{geo}(\bar{w})\textstyle\sum_{i\in[\![d]\!]} w_i/\bar{w}_i \geq 0. \tag{5.36}$$

If we assume that $\bar{w}$ satisfies the initial cuts $w \geq 0$ but not strictly, i.e. some elements of $\bar{w}$ are zero, then we cannot use a simple gradient cut because the geometric mean function is not smooth at $\bar{w}$. In this case, a simple approach is to set any $\bar{w}_i = 0$ to $\bar{w}_i = \epsilon > 0$ for some small positive ϵ such that the gradient cut (5.36) separates $\bar{w}$. However, such cuts can have bad conditioning and may affect the numerical performance of the OA solver. It is possible to derive better separation cuts in this case (similar to our approach in Section 4.6.1 for the exponential cone in Pajarito), but we leave this for future work.

Now suppose we want to use an EF in the OA model (to tighten polyhedral relaxations), but we still want to use the NF in the conic subproblems (to avoid slowing down the conic solver). In terms of $1 + d$ auxiliary variables $\theta \in \mathbb{R}$ and $\lambda \in \mathbb{R}^d$, the EF for $(u, w) \in \mathcal{K}_{\text{geo}}$ is:

$$\theta \geq u, \tag{5.37a}$$
$$e'\lambda \geq 0, \tag{5.37b}$$
$$(\lambda_i, \theta, w_i) \in \mathcal{K}_{\log} \qquad \forall i \in [\![d]\!]. \tag{5.37c}$$

Note that this EF is slightly different (in terms of signs) to the EF (2.17) implemented in MathOptInterface bridges. Recall from the definition of the three-dimensional logarithm cone (i.e. the standard exponential cone) in Section 2.2.3.2 that $(\lambda_i, \theta, w_i) \in \mathcal{K}_{\log}$ is equivalent (modulo closures) to the variable bounds $\theta \geq 0$ and $w_i \geq 0$ and the convex constraint $\lambda_i \leq \theta \log(w_i/\theta)$. The three-dimensional dual logarithm cone is:

$$\mathcal{K}^*_{\log} = \text{cl}\{(p, q, r) : p \leq 0, r \geq 0, q \geq p(\log(-r/p) + 1)\}, \tag{5.38}$$

and any point $(p, p(\log(-r/p) + 1), r)$ with $p < 0$ and $r > 0$ is an extreme ray of $\mathcal{K}^*_{\log}$.

For each $\mathcal{K}_{\log}$ cone in the EF, we add the initial cuts $\theta \geq 0$ and $w_i \geq 0$. Since the point $(-1, -1, 1)$ is an extreme ray of $\mathcal{K}^*_{\log}$, we can add the d initial cuts:

$$-\lambda_i - \theta + w_i \geq 0 \qquad \forall i \in [\![d]\!]. \tag{5.39}$$

Together with the EF linear constraints in (5.37), these cuts imply our NF space initial cut $-du + e'w \geq 0$ because:

$$0 \leq \textstyle\sum_{i\in[\![d]\!]}(-\lambda_i - \theta + w_i) = -e'\lambda - d\theta + e'w \leq -du + e'w. \tag{5.40}$$

For an NF space dual point $(\bar{u}, \bar{w}) \in \mathcal{K}^*_{\text{geo}}$ with $\bar{w} > 0$, we can strengthen and extend the corresponding NF cut to the d extreme EF cuts:

$$p\lambda_i + p(\log(-\bar{w}_i/p) + 1)\theta + \bar{w}_i w_i \geq 0 \qquad \forall i \in [\![d]\!], \tag{5.41}$$

where we let $p = -\operatorname{geo}(\bar{w}) < 0$. Note that $\bar{u}$ is ignored, and we have:

$$\textstyle\sum_{i\in[\![d]\!]} \log(-\bar{w}_i/p) = \log\big(\prod_{i\in[\![d]\!]}(\bar{w}_i/\operatorname{geo}(\bar{w}))\big) = \log(1) = 0. \tag{5.42}$$

Therefore, together with the EF linear constraints in (5.37), these cuts imply:

$$0 \leq \textstyle\sum_{i\in[\![d]\!]}(p\lambda_i + p(\log(-\bar{w}_i/p) + 1)\theta + \bar{w}_i w_i) = p(e'\lambda + d\theta) + \bar{w}'w \leq dpu + \bar{w}'w. \tag{5.43}$$

Hence the disaggregated extreme cuts (5.41) are at least as strong as the NF space extreme cut (5.35).

Finally, suppose we have an NF space primal feasible solution $(\bar{u}, \bar{w}) \in \mathcal{K}_{\text{geo}}$ obtained by the subproblem solver, and we want to find values for the auxiliary variables θ and λ satisfying the EF space constraints (5.37). This allows MOIPajarito to give a feasible solution to the OA solver. We take $\theta = \operatorname{geo}(\bar{w}) \geq 0$. If $\theta = 0$, we let $\lambda_i = 0$, otherwise we let $\lambda_i = \theta \log(\bar{w}_i/\theta)$.

Figures 5.1 and 5.2 illustrate the impact of the geometric mean cone EF on the strength of polyhedral relaxations. In Figure 5.1, we consider a simple three-dimensional convex constraint $\operatorname{geo}(w) \geq 1$ for $w \in [0, 3]^3$. Note this is equivalent to $(1, w) \in \mathcal{K}_{\text{geo}}$, i.e. u is fixed to one, and in the EF we let $\theta = u = 1$. Consider seven points for $\bar{w} \in \mathbb{R}^3$: e and the three permutations of each of $(3^{-1}, 1, 1)$ and $(3^{-3}, 1, 1)$. At each point, we add the corresponding cuts (5.35) to the NF or (5.41) to the EF. In Figure 5.1 (right) we plot the EF cuts (5.41) for each $i \in [\![3]\!]$. We project the three-dimensional NF and the 6-dimensional EF onto w_1 and w_2. From Figure 5.1 (left), we see that the projected OA polyhedron is larger (and has fewer extreme points) for the NF than for the EF. We performed this analysis by setting up JuMP models and using Polyhedra.jl (Benoît Legat et al., 2021) (with CDDLib) to find the extreme points of the projected polyhedra.

Figure 5.2 shows the iteration counts and solve times for the separation-only algorithm (using only the initial fixed cuts and the separation cuts, and not solving any conic subproblems) for the continuous relaxation of the simple knapsack example in Section 5.5.4.1 with the negative geometric mean objective. We relax the integrality constraints because these cause a significant slowdown and make the plots very nonsmooth, which obscures the trends. The cone dimension and number of primal variables is $1 + n$. We only set up instances up to size $n = 200$ with *EF on* and $n = 70$ with *EF off* No instances fail to converge. Overall *EF on* requires much fewer iterations and much fewer cuts (which are only three-dimensional cuts instead n-dimensional) than *EF off*, and this translates to much faster solve times.

Figure 5.1: For a three-dimensional geometric mean ball constraint $\text{geo}(w) \geq 1$ with $w \in [0,3]^3$, projections of the NF and EF onto the first two variables w_1, w_2, given a fixed set of seven NF space cuts.

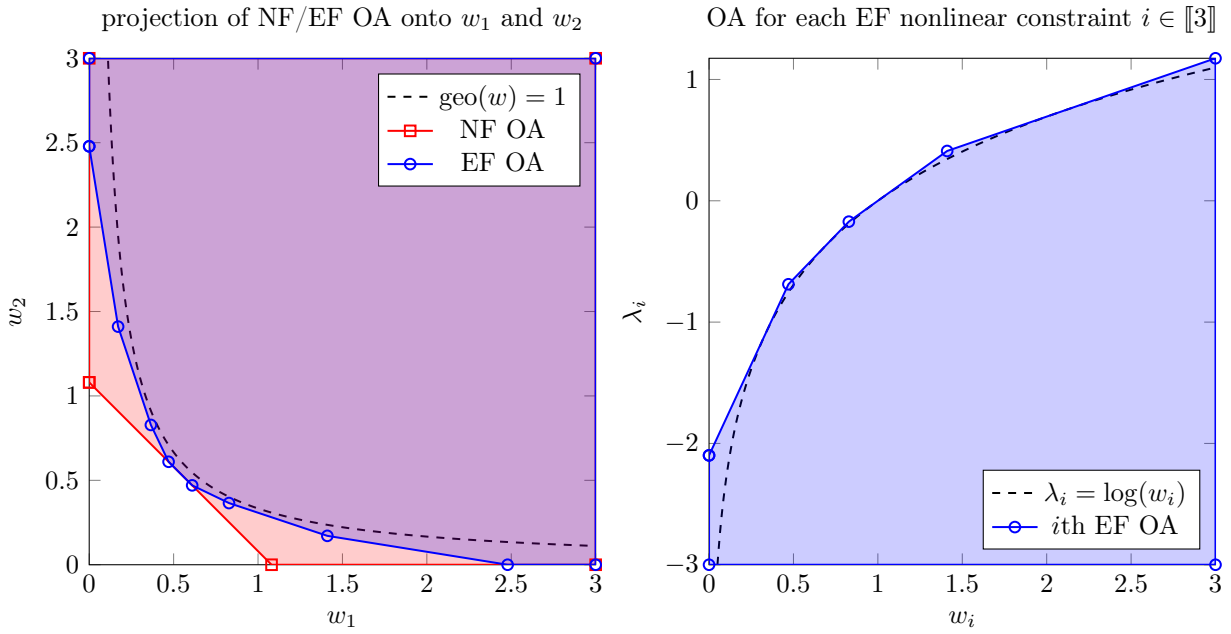
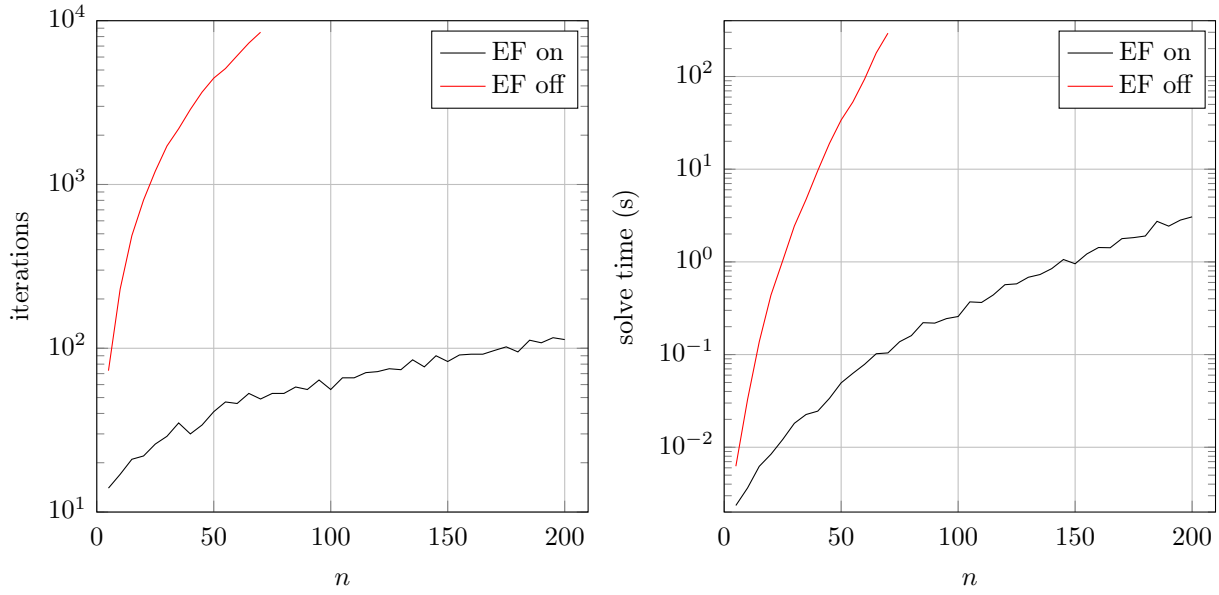


Figure 5.2: Results for the separation-only algorithm on the continuous relaxation of the knapsack example in Section 5.5.4.1 with the negative geometric mean objective.



5.5.2 Vector separable spectral function cones

Suppose $\varphi : \mathbb{R}^d \to \mathbb{R}$ is a convex separable spectral function defined on the positive reals, i.e. $\varphi(w) = \sum_{i \in [\![d]\!]} \varphi(w_i)$ for $w > 0$. Let φ^* be the convex conjugate function (see Section 3.4.3) of φ, which is also a convex separable spectral function (possibly infinite-valued). We define the vector separable spectral cone and its dual cone as:

$$\mathcal{K}_{\text{sep}} \coloneqq \text{cl}\{(u, v, w) \in \mathbb{R}^{2+d} : v > 0, w \geq 0, u \geq v\varphi(w/v)\}, \tag{5.44a}$$

$$\mathcal{K}^*_{\text{sep}} \coloneqq \text{cl}\{(u, v, w) \in \mathbb{R}^{2+d} : u > 0, v \geq u\varphi^*(w/u)\}. \tag{5.44b}$$

Examples of suitable φ functions belonging to the matrix monotone derivative (MMD) class are discussed in Section 3.6 and Table 3.1 (note $\mathcal{Q} = \mathbb{R}^d_{\geq}$ is the vector domain), including *NegLog*, *NegEntropy*, *NegSqrt*, *NegPower*, and *Power*. Note that for these MMD functions, the conjugates φ^* may or may not implicitly restrict $w \geq 0$ in $\mathcal{K}^*_{\text{sep}}$. We predefine this list of MMD functions in both Hypatia and MOIPajarito. Like Hypatia, MOIPajarito allows defining new φ functions through a small set of oracles, including those already needed by Hypatia. The additional φ oracles needed are (1) gradients of the conjugate function φ^*, and (2) a list of points from which we derive initial fixed cuts.

MOIPajarito supports both $\mathcal{K}_{\text{sep}}$ and $\mathcal{K}^*_{\text{sep}}$. For simplicity, here we only discuss oracles for the primal cone $\mathcal{K}_{\text{sep}}$, as these are simple to adapt for $\mathcal{K}^*_{\text{sep}}$ by e.g. swapping the epigraph and perspective variables. Suppose the constraint is $(u, v, w) \in \mathcal{K}_{\text{sep}}$. As in the geometric mean cone case, using the EF in MOIPajarito is optional, so first we show the cut techniques without the EF.

For initial cuts, we add the $1 + d$ variable bounds $v \geq 0$ and $w \geq 0$. Furthermore, by letting $w = re$ for some scalar r (such that $\varphi^*(r) < \infty$), we see from (5.44b) that $(1, d\varphi^*(r), re)$ is an extreme ray of $\mathcal{K}^*_{\text{sep}}$ that defines a simple initial cut:

$$u + d\varphi^*(r)v + re'w \geq 0. \tag{5.45}$$

Indeed, given a point $(\bar{u}, \bar{v}, \bar{w}) \in \mathbb{R}^{1+d}$ with $\bar{u} > 0, \varphi^*(\bar{w}) < \infty$, we can ignore $\bar{v}$ and obtain an extreme cut:

$$\bar{u}u + \bar{u}\varphi^*(\bar{w}/\bar{u})v + \bar{w}'w \geq 0. \tag{5.46}$$

To separate a point $(\bar{u}, \bar{v}, \bar{w}) \notin \mathcal{K}_{\text{sep}}$ that satisfies $\bar{v} > 0$, $\bar{w} > 0$, we add the gradient cut:

$$u + \varphi^*(r)v + r'w \geq 0, \tag{5.47}$$

where $r_i = -\nabla\varphi(\bar{w}_i/\bar{v}), \forall i \in [\![d]\!]$ and $\nabla\varphi$ is the derivative of univariate φ. Note this separation cut ignores $\bar{u}$. If we do not have $\bar{v} > 0$ and $\bar{w} > 0$, then we may be unable to compute the gradient. In this case it may be possible to derive a violated gradient cut from a nearby point; we leave the derivation of better separation cuts for future work.

Now suppose we use an EF in the OA model. In terms of auxiliary variables $\lambda \in \mathbb{R}^d$, the EF for

$(u, v, w) \in \mathcal{K}_{\text{sep}}$ is written in terms of a linear inequality and d $\mathcal{K}_{\text{sep}}$ cones of dimension three:

$$u \geq e'\lambda, \tag{5.48a}$$

$$(\lambda_i, v, w_i) \in \mathcal{K}_{\text{sep}} \qquad \forall i \in [\![d]\!]. \tag{5.48b}$$

For initial cuts we again add $v \geq 0$ and $w_i \geq 0$. For each r value from (5.45), we add d extreme initial cuts in EF space:

$$\lambda_i + \varphi^*(r)v + rw_i \geq 0 \qquad \forall i \in [\![d]\!]. \tag{5.49}$$

These cuts imply the corresponding NF space initial cuts, so our initial OA is at least as strong.

Indeed, for any NF space dual point $(\bar{u}, \bar{v}, \bar{w}) \in \mathcal{K}^*_{\text{sep}}$ with $\bar{u} > 0$ and $\varphi^*(\bar{w}) < \infty$, we can strengthen and extend the NF cut to d extreme EF cuts:

$$\bar{u}\lambda_i + \bar{u}\varphi^*(\bar{w}_i/\bar{u})v + \bar{w}_i w_i \geq 0 \qquad \forall i \in [\![d]\!]. \tag{5.50}$$

Note $\bar{v}$ is ignored. These cuts, together with (5.48a), imply the corresponding NF cut, since:

$$0 \leq \textstyle\sum_{i \in [\![d]\!]}(\bar{u}\lambda_i + \bar{u}\varphi^*(\bar{w}_i/\bar{u})v + \bar{w}_i w_i) = \bar{u}e'\lambda + \bar{u}\varphi^*(\bar{w}/\bar{u})v + \bar{w}'w \leq \bar{u}u + \bar{v}v + \bar{w}'w. \tag{5.51}$$

Finally, suppose we have an NF space primal feasible solution $(\bar{u}, \bar{v}, \bar{w}) \in \mathcal{K}_{\text{sep}}$. To extend this to a feasible solution for the auxiliary variables λ, we simply let $\lambda_i = \bar{v}\varphi(\bar{w}_i/\bar{v})$ if $\bar{v} > 0$, $\bar{w}_i > 0$, otherwise we let $\lambda_i = 0$.

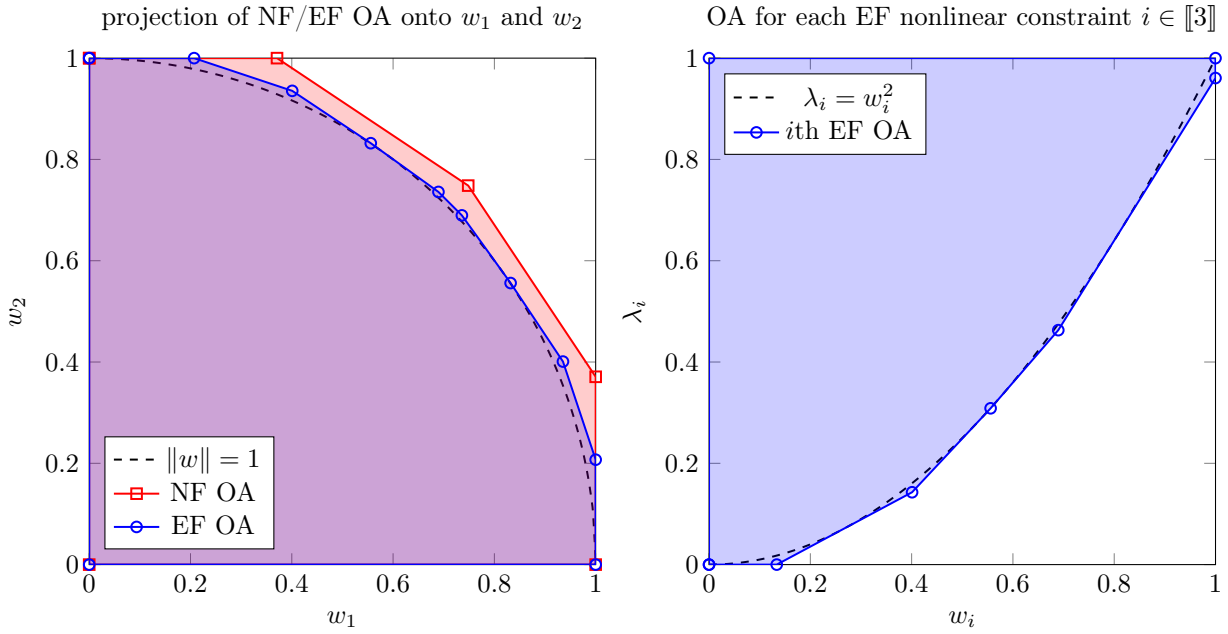
We illustrate the impact of the separable spectral vector cone EF on the strength of polyhedral relaxations. Similar to Figure 5.1, in Figure 5.3 we consider a simple three-dimensional convex constraint $\sum_{i \in [\![3]\!]} w^2 \leq 1$ for $w \in [0, 1]^3$. This convex constraint has the form $(1, 1, w) \in \mathcal{K}_{\text{sep}}(\text{Power}(2))$ with $d = 3$, and is equivalent to $\|w\| \leq 1$ (note it can also be expressed with $\mathcal{K}_{\ell_2}$ and linear constraints). Consider seven points for $\bar{w} \in \mathbb{R}^3$: e and the six permutations of $(1, 2, 3)$. At each point, we add the corresponding cuts (5.46) to the NF or (5.50) to the EF. In Figure 5.3 (right) we plot the EF cuts for each $i \in [\![3]\!]$. We project the three-dimensional NF and the 6-dimensional EF onto w_1 and w_2. From Figure 5.3 (left), we see that the projected OA polyhedron is larger (and has fewer extreme points) for the NF than for the EF.

Similar to Figure 5.2, in Figure 5.4 we plot the iteration counts and solve times for the separation-only algorithm for the continuous relaxation of the knapsack example in Section 5.5.4.1 with the sum-negative-logarithm (*NegLog*) objective or the sum-inverse (*NegSqrt* conjugate) objective. No instances fail to converge, and *EF on* requires much fewer iterations and cuts than *EF off*, translating into much faster solve times.

5.5.3 Matrix spectral function cones

The root-determinant cone $\mathcal{K}_{\text{rtdet}}$ defined in Section 2.2.3.1 and discussed in Section 3.7 is an analogy to the geometric mean cone in Section 5.5.1 for real symmetric or complex Hermitian domains.

Figure 5.3: For a three-dimensional Euclidean ball constraint $\|w\| \leq 1$ with $w \in [0,1]^3$, projections of the NF and EF onto the first two variables w_1, w_2, given a fixed set of seven NF space cuts.



Similarly, the real symmetric or complex Hermitian separable spectral function cones (and their dual cones) defined in Section 2.2.3.3 and discussed in Section 3.6 are analogies of the vector separable spectral cone in Section 5.5.2.

The $\mathcal{K}^*$ oracles we use for these matrix domain spectral function cones are in direct analogy to the NF oracles we derive in Sections 5.5.1 and 5.5.2 for the vector domain cases, so we do not describe the details here. Note we adapt techniques for the PSD cone from Section 5.3 to add initial cuts and to decompose certain separation cuts and subproblem cuts. For these matrix domain cones, we do not have simple EFs in terms of only three-dimensional cones, so we do not use internally managed EFs in MOIPajarito. However, MOIPajarito enables us to investigate whether the EFs we describe in Section 2.2.3, despite their size and complexity, could be helpful in accelerating OA for certain problems.

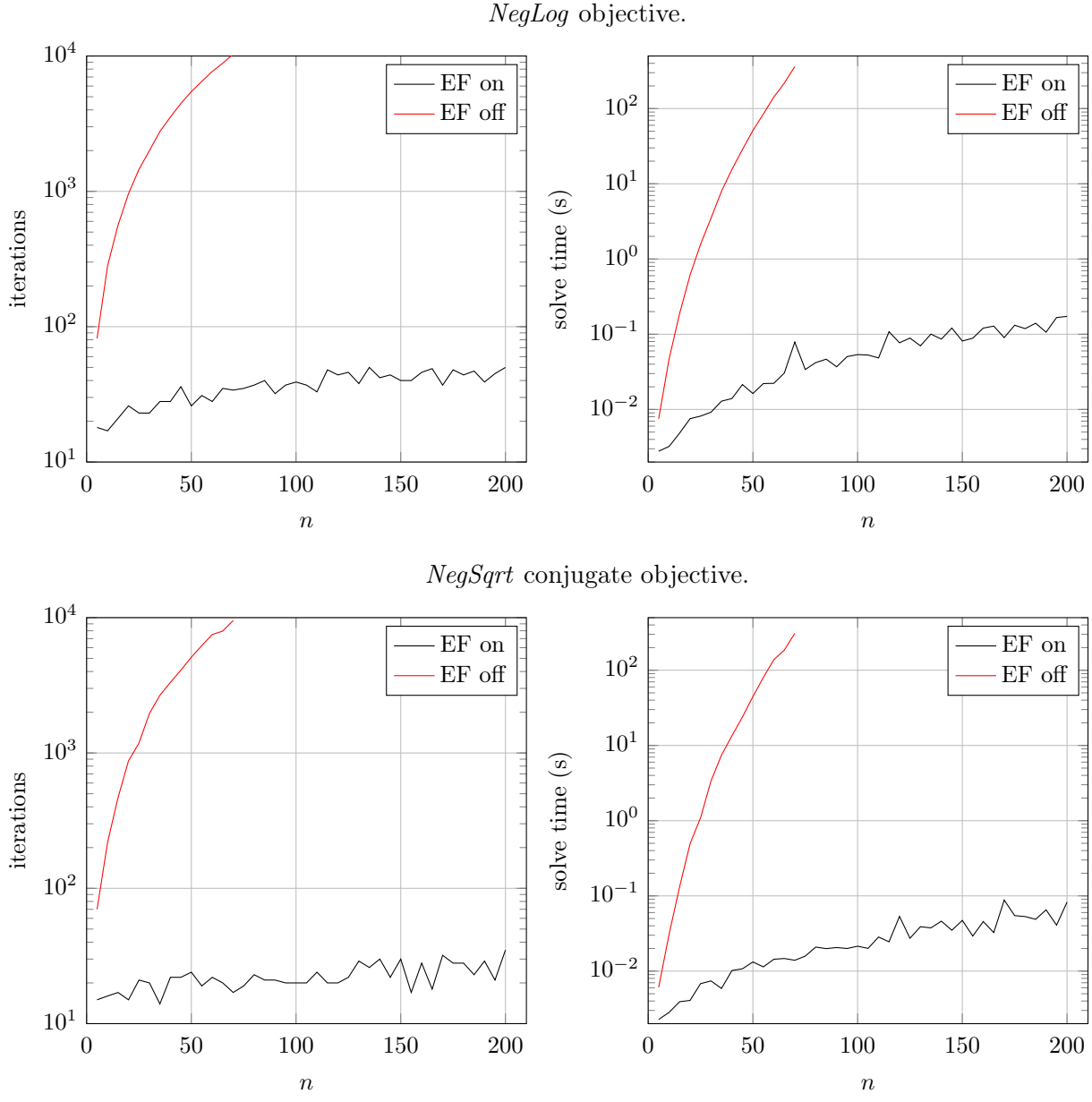
5.5.4 Examples for spectral function cones

5.5.4.1 Knapsack problem with convex objective

We formulate a simple integer knapsack problem with a convex objective. For each item $i \in [\![n]\!]$, the per-unit weight and value are $b_i > 0$ and $c_i > 0$, and the variable $x_i \geq 1$ represents the number of units we select. We have a weight budget of $B > 0$ and we minimize a convex spectral function $f : \mathbb{R}^n \to \mathbb{R}$ of the values $c_i x_i, \forall i \in [\![n]\!]$ of the units selected. The high-level model is:

$$\min_x \quad f\big((c_i x_i)_{i \in [\![n]\!]}\big) : \tag{5.52a}$$

Figure 5.4: Results for the separation-only algorithm on the continuous relaxation of the knapsack example in Section 5.5.4.1 with two different objective functions.



$$x \geq e, \tag{5.52b}$$
$$b'x \leq B, \tag{5.52c}$$
$$x \in \mathbb{Z}^n. \tag{5.52d}$$

The equivalent MI-conic formulation introduces an epigraph variable and a vector spectral function cone constraint (over $\mathcal{K}_{\text{geo}}$ or $\mathcal{K}_{\text{sep}}$) to represent the objective. The standard conic EFs for these types of constraints are discussed in Section 2.2.3.

We generate random instances of the conic formulation for (5.52) with the negative geometric mean objective for various numbers of items n. All instances are feasible and bounded, and our results are summarized in Table 5.8. *nat* is the standard algorithm that uses the NF for the continuous conic models and the EF for the OA model, *noext* uses the NF for both the conic and the OA models, and *ext* uses the EF for both the conic and the OA models. *nat* and *ext* solve similar numbers of instances in similar numbers of iterations, though *nat* solves one size larger and is typically faster, probably because Hypatia solves the NFs faster. *noext* takes many more iterations than *nat* or *ext* and only solves the smallest instances before hitting time limits. This suggests that the geometric mean cone EF generates tighter polyhedral relaxations.

Table 5.8: Knapsack problem with convex objective solver statistics.

	nat			noext			ext		
n	st	it	time	st	it	time	st	it	time
3	co	3	0.0	co	5	0.0	co	3	0.0
6	co	8	0.1	co	34	0.7	co	8	0.1
9	co	6	17	co	71	165	co	6	21
12	co	9	22	tl	30	600	co	7	34
15	co	6	4.3	tl	253	600	co	8	5.9
20	co	7	13	tl	76	600	co	8	34
25	co	8	15	tl	109	600	co	8	17
30	co	9	21	tl	175	600	co	8	19
35	co	7	9.7	tl	543	600	co	7	14
40	co	11	3.1	*	*	*	co	12	6.9
50	co	8	9.5	*	*	*	co	8	19
60	co	8	14	*	*	*	co	11	15
70	co	7	13	*	*	*	co	10	20
80	co	8	9.5	*	*	*	co	8	12
90	co	12	57	*	*	*	co	15	63
100	co	10	409	*	*	*	tl	7	600
110	tl	6	600	*	*	*	tl	6	600
120	tl	5	600	*	*	*	tl	5	600

5.5.4.2 Sparse regression with prior constraints

Given observations $(Y_i, X_i), \forall i \in [\![n]\!]$ with $Y_i \in \mathbb{R}$ and $X_i \in \mathbb{R}^m$, we formulate a linear regression problem with ℓ_2 norm loss and ℓ_0 norm regularization (with parameter $\lambda > 0$) to encourage sparsity. We are given some prior information expressed as L separable spectral function constraints on the

nonnegative parameter variable $\beta \in \mathbb{R}^m_{\geq}$. The high-level model is:

$$\min_\beta \quad \|Y - X\beta\| + \lambda\|\beta\|_0 : \tag{5.53a}$$

$$\beta \geq 0, \tag{5.53b}$$

$$f_l(e + \beta) \leq a_l \qquad \forall l \in [\![L]\!]. \tag{5.53c}$$

We let $L = 2$ and suppose f_1 is the *NegEntropy* function and f_2 is the *NegSqrt* conjugate function (sum-inverse). We use a big-M formulation for the ℓ_0 norm, assuming that $\hat{\beta}$ upper-bounds β. The conic form model is:

$$\min_{\beta,u,z} \quad u + \lambda e'z : \tag{5.54a}$$

$$(u, Y - X\beta) \in \mathcal{K}_{\ell_2}, \tag{5.54b}$$

$$\beta \in \mathbb{R}^m_{\geq}, \tag{5.54c}$$

$$\hat{\beta}z - \beta \in \mathbb{R}^m_{\geq}, \tag{5.54d}$$

$$(a_1, 1, e + \beta) \in \mathcal{K}_{\text{sep}}(\text{NegEntropy}), \tag{5.54e}$$

$$(1, a_2, e + \beta) \in \mathcal{K}^*_{\text{sep}}(\text{NegSqrt}), \tag{5.54f}$$

$$z \in \{0, 1\}^m. \tag{5.54g}$$

The standard conic EFs for $\mathcal{K}_{\text{sep}}$ and $\mathcal{K}^*_{\text{sep}}$ are discussed in Section 2.2.3.3; for the *NegEntropy*, the EF uses exponential cones, and for the *NegSqrt* conjugate, the EF uses second order cones.

We generate random instances of (5.54), varying the number of observations n and letting the parameter dimension be $m = n$. All instances are feasible and bounded, and our results are summarized in Table 5.9. *nat* is the standard algorithm and *noext* uses the NF for both the conic and the OA models (for $\mathcal{K}_{\ell_2}$, $\mathcal{K}_{\text{sep}}$, and $\mathcal{K}^*_{\text{sep}}$). Whereas *nat* solves instances reliably in fewer than ten iterations, *noext* takes many more iterations and fails to converge on all but the smallest instance.

Table 5.9: Sparse regression with prior constraints solver statistics.

	nat			noext		
d	st	it	time	st	it	time
10	co	4	0.1	co	19	0.3
20	co	5	0.6	er	57	4.5
30	co	5	0.7	er	62	34
40	co	8	3.3	er	109	319
50	co	7	6.5	tl	7	600
60	co	9	24	tl	7	600
70	co	9	27	*	*	*
80	co	9	38	*	*	*
90	tl	3	600	*	*	*
100	tl	1	600	*	*	*

5.5.4.3 Experiment design

We formulate a discrete experiment design problem similar to Boyd and Vandenberghe (2004, Section 7.5). The variable $x \in \mathbb{R}^k$ is the integer number of trials to run for each of k different experiments that are useful for estimating a vector in $\mathbb{R}^d$. The experiments are described by the columns of $V \in \mathbb{R}^{d \times k}$. A total of k experiments are to be performed, and each experiment can be performed at most twice. We minimize a convex spectral function f of the information matrix. The high-level model is:

$$\begin{aligned} \min_x \quad & f(V \operatorname{Diag}(x) V') : && \text{(5.55a)} \\ & 0 \leq x \leq 2e, && \text{(5.55b)} \\ & e'x = k, && \text{(5.55c)} \\ & x \in \mathbb{Z}^k. && \text{(5.55d)} \end{aligned}$$

If f is the negative root-determinant function, the equivalent conic form model for (5.55) is:

$$\begin{aligned} \min_{x,z} \quad & z : && \text{(5.56a)} \\ & e'x = k, && \text{(5.56b)} \\ & x \in \mathbb{R}^k_{\geq}, && \text{(5.56c)} \\ & 2e - x \in \mathbb{R}^k_{\geq}, && \text{(5.56d)} \\ & (-z, \operatorname{vec}(V \operatorname{Diag}(x) V')) \in \mathcal{K}_{\text{rtdet}}, && \text{(5.56e)} \\ & x \in \mathbb{Z}^k, && \text{(5.56f)} \end{aligned}$$

where vec is the svec operator. The hypograph of root-determinant cone $\mathcal{K}_{\text{rtdet}}$ and its standard conic EF are described in Section 2.2.3.1. Alternatively, if f is a matrix separable spectral function (e.g. *NegEntropy*, the negative entropy of the eigenvalues), we replace (5.56e) with:

$$(z, 1, \operatorname{vec}(V \operatorname{Diag}(x) V')) \in \mathcal{K}_{\text{sep}}. \tag{5.57}$$

We discuss matrix separable spectral function cones and standard conic EFs in Section 2.2.3.3.

We generate random instances of the root-determinant and *NegEntropy* variants of (5.56) for various dimensions d, with $k = 2d$ experiments. All instances are feasible and bounded, and our results are summarized in Tables 5.10 and 5.11. In both cases, *nat* is usually faster and solves larger instances within the time limit, and *ext* encounters numerical convergence issues on larger instances. However, on the instances that both algorithms converge on, *ext* usually takes fewer iterations. This suggests that the EFs for these cones are expensive but may yield tighter polyhedral relaxations (albeit with numerical issues), so these EFs are worth investigating further.

Table 5.10: Experiment design solver statistics for the root-determinant objective.

	nat			ext		
d	st	it	time	st	it	time
3	co	0	0.0	co	1	0.0
5	co	28	2.0	co	13	0.4
7	co	362	29	er	1	0.0
9	co	695	134	er	1	0.1
11	co	513	88	er	1	0.1
13	tl	1201	600	er	3	0.3
15	tl	3702	600	er	3	0.3

Table 5.11: Experiment design solver statistics for the negative entropy objective.

	nat			ext		
d	st	it	time	st	it	time
3	co	8	0.1	co	4	0.1
5	co	10	0.2	co	5	0.7
7	co	61	2.7	co	15	9.2
9	co	108	5.6	co	18	45
11	co	369	109	er	1	2.2
13	co	757	328	er	1	5.9
15	tl	703	600	er	1	14
17	tl	1699	600	er	2	57

5.5.4.4 Inverse covariance estimation

Given an empirical covariance matrix $\Sigma \in \mathbb{S}^d_{\succeq}$, we estimate an inverse covariance matrix; see Hsieh et al. (2012) and Bertsimas, Lamperski, and Pauphilet (2020) for related approaches. We use squared Frobenius norm regularization (with parameter $\lambda > 0$), and we constrain the number of rows of off-diagonal nonzeros to be at most k, since some predictors are assumed to be unrelated to others. The high-level model over the symmetric matrix variable P is:

$$\min_{P,y} \quad \langle \Sigma, P \rangle - \operatorname{logdet}(P) + \lambda \|P\|^2 : \tag{5.58a}$$

$$P \in \mathbb{S}^d_{\succeq}, \tag{5.58b}$$

$$y \in \{0,1\}^d, \tag{5.58c}$$

$$e'y = k, \tag{5.58d}$$

$$y_i = 0 \quad \Rightarrow \quad P_{i,j} = 0 \qquad \forall i, j \in [\![d]\!] : i \neq j. \tag{5.58e}$$

To construct an efficient conic formulation for (5.58), we formulate the implication constraints (5.58e) and the squared Frobenius norm terms jointly using $\mathcal{K}_{\text{sqr}}$ cones. We introduce auxiliary variables $u, z_0 \in \mathbb{R}$ and $z \in \mathbb{R}^d$. The variable $p \in \mathbb{R}^{\text{sd}(d)}$ is the svec transformation of the symmetric

matrix variable P in (5.58). Letting $P = \mathrm{mat}(p)$ for convenience below, the conic form model is:

$$\begin{aligned}
\min_{p,y,u,z_0,z} \quad & \langle \Sigma, P \rangle + u + \lambda(z_0 + e'z) : && && \text{(5.59a)}\\
& e'y = k, && && \text{(5.59b)}\\
& (u, 1, p) \in \mathcal{K}_{\mathrm{sep}}(\mathrm{NegLog}), && && \text{(5.59c)}\\
& (z_0/2, 1, \mathrm{diag}(P)) \in \mathcal{K}_{\mathrm{sqr}}, && && \text{(5.59d)}\\
& \left(z_i/2, y_i, (P_{i,j})_{j \in [\![d]\!]: j \neq i}\right) \in \mathcal{K}_{\mathrm{sqr}} && \forall i \in [\![d]\!], && \text{(5.59e)}\\
& y \in \{0,1\}^d. && && \text{(5.59f)}
\end{aligned}$$

For the $\mathcal{K}_{\mathrm{sep}}(\mathrm{NegLog})$ constraint, we use the standard conic EF in Section 2.2.3.2 for the log-determinant cone.

We generate random instances of (5.59) for various side dimensions d, with $k = d - \lfloor \sqrt{d} \rfloor$. All instances are feasible and bounded, and our results are summarized in Table 5.12. *nat* converges on all instances up to size $d = 14$ before hitting a time limit, but *ext* encounters numerical convergence issues on all sizes except $d = 9$. Note $\mathcal{K}_{\mathrm{sqr}}$ is a standard cone so *nat* and *ext* only differ in their treatment of $\mathcal{K}_{\mathrm{sep}}(\mathrm{NegLog})$.

Table 5.12: Inverse covariance estimation solver statistics.

	nat			ext		
d	st	it	time	st	it	time
5	co	6	0.2	er	2	0.0
6	co	14	0.7	er	2	0.1
7	co	13	1.4	er	2	0.1
8	co	9	1.0	co	9	3.1
9	co	7	0.9	er	2	0.2
10	co	52	13	er	2	0.4
11	co	103	55	er	2	0.6
12	co	51	20	er	2	0.3
13	co	132	162	er	1	0.5
14	co	118	188	er	2	0.5
15	tl	173	600	er	2	1.2
16	tl	132	600	er	2	10

5.6 Advanced mixed-integer conic formulations

5.6.1 Tight conic formulations for disjunctions of convex constraints

Suppose we have a finite collection of convex sets $(\mathcal{C}_i)_{i \in [\![n]\!]}$. For simplicity, assume each $\mathcal{C}_i$ is bounded. The union $\cup_{i \in [\![n]\!]} \mathcal{C}_i$ of these sets is nonconvex in general, and we denote its convex hull as $\mathrm{conv}_{i \in [\![n]\!]} \mathcal{C}_i$. Suppose we want an MI-convex formulation of the nonconvex constraint $y \in \cup_{i \in [\![n]\!]} \mathcal{C}_i$. Furthermore, we want its continuous relaxation to enforce $y \in \mathrm{conv}_{i \in [\![n]\!]} \mathcal{C}_i$.

For $i \in [\![n]\!]$, let $\mathcal{K}_i \subset \mathbb{R}^{1+d}$ be the convex cone generated by $\mathcal{C}_i \subset \mathbb{R}^d$:

$$\mathcal{K}_i \coloneqq \operatorname{cone}\{(1, y) : y \in \mathcal{C}_i\} = \{(\theta, \theta y) : \theta \in \mathbb{R}_{\geq}, y \in \mathcal{C}_i\}, \tag{5.60}$$

where cone is the conic hull operator. The standard *copies-of-variables* formulation for $y \in \cup_{i \in [\![n]\!]} \mathcal{C}_i$ uses, for each $i \in [\![n]\!]$, an auxiliary binary variable x_i and a 'copy' variable $z_i \in \mathbb{R}^d$:

$$\sum_{i \in [\![n]\!]} z_i = y, \tag{5.61a}$$
$$e'x = 1, \tag{5.61b}$$
$$(x_i, z_i) \in \mathcal{K}_i \qquad \forall i \in [\![n]\!], \tag{5.61c}$$
$$x \in \{0,1\}^n. \tag{5.61d}$$

Formulation (5.61) is discussed in Vielma (2018) and appears in, for example, Ben-Tal and Nemirovski (2001). It is a *unary* formulation, as the number of binary variables is linear in n. Any feasible solution has an index $i \in [\![n]\!]$ for which $x_i = 1$ and $z_i = y \in \mathcal{C}_i$, and $x_j = 0$ and $z_j = 0$ for all $j \neq i$. It is a tight formulation since the continuous relaxation has integral extreme points, and the projection onto y yields the convex hull formulation $y \in \operatorname{conv}_{i \in [\![n]\!]} \mathcal{C}_i$. This tightness comes at the cost of many auxiliary variables.

5.6.2 Piecewise linear relaxations of nonconvex equality constraints

Suppose $f : \mathbb{R} \to \mathbb{R}$ is a convex univariate function. Suppose we have a list of L increasing points on the real line, $(\omega_l)_{l \in [\![L]\!]}$, and we are only concerned with f on the domain $[\omega_1, \omega_L]$. We define the 2-dimensional graph of a piecewise linear (PWL) approximation of f as follows:

$$\operatorname{gr} \operatorname{pwl} f \coloneqq \cup_{l \in [\![L-1]\!]} \operatorname{conv}\{(\omega_l, f(\omega_l)), (\omega_{l+1}, f(\omega_{l+1}))\}. \tag{5.62}$$

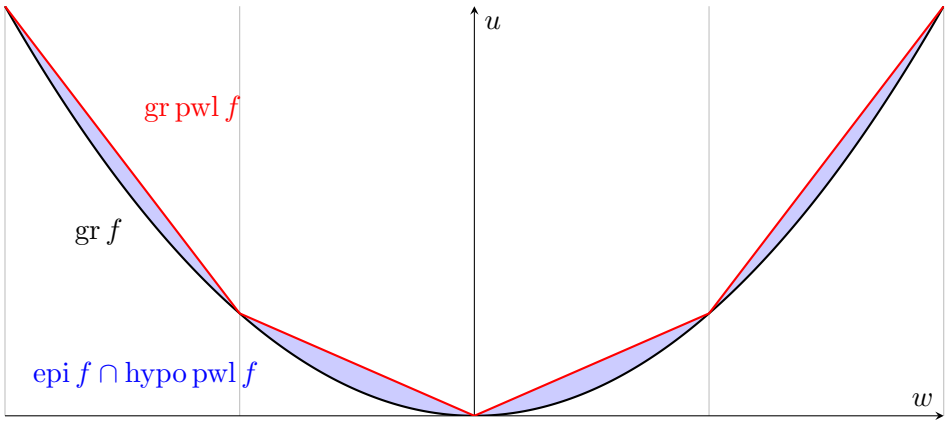
This set is a finite union of bounded polyhedra, so it is MILP-representable.

Consider the nonconvex nonlinear equality constraint $u = f(w)$, or equivalently $(u, w) \in \operatorname{gr} f$, for $w \in [\omega_1, \omega_L]$. See Figure 5.5 for an illustration. Since $\operatorname{gr} f \subset \operatorname{epi} f \cap \operatorname{hypo} \operatorname{pwl} f$ on the domain of interest, we can relax the nonconvex constraint to the convex constraint $u \geq f(w)$ and the PWL hypograph constraint $u \leq \operatorname{pwl} f(w)$, which sandwich the graph. Since the hypograph set $\operatorname{hypo} \operatorname{pwl} f$ is a finite union of bounded polyhedra with equal recession cones (the direction $(-1, 0)$), it is MILP-representable, like $\operatorname{gr} \operatorname{pwl} f$. Thus we can write an MI-convex formulation for this sandwiching relaxation.

For variables $\sigma \in \mathbb{R}^L$, a *type-2 special ordered set* (SOS2) constraint $\sigma \in \text{SOS2}$ enforces that all but two adjacent variables σ_l, σ_{l+1} are zero. We write an SOS2-linear formulation for the PWL hypograph constraint $(u, w) \in \operatorname{hypo} \operatorname{pwl} f$ for $w \in [\omega_1, \omega_L]$:

$$\sum_{l \in [\![L]\!]} \sigma_l f(\omega_l) \geq u, \tag{5.63a}$$
$$\sigma'\omega = w, \tag{5.63b}$$

Figure 5.5: A relaxation of the nonconvex set $\operatorname{gr} f$ for the convex function $f(w) = w^2$.



$$e'\sigma = 1, \tag{5.63c}$$

$$\sigma \geq 0, \tag{5.63d}$$

$$\sigma \in \text{SOS2}\,. \tag{5.63e}$$

Before turning our attention to mixed-integer formulations for SOS2 constraints in Section 5.6.4, we introduce a useful generalization of PWL constraints: *homogenized PWL constraints*.

5.6.3 Homogenized piecewise linear formulations

Now suppose we are concerned with the bivariate perspective function of f, $\operatorname{per} f : \mathbb{R}_> \times \mathbb{R} \to \mathbb{R}$:

$$\operatorname{per} f(v, w) \coloneqq vf(w/v). \tag{5.64}$$

This function is convex, since f is convex and the perspective function preserves convexity. The three-dimensional $\operatorname{gr} \operatorname{per} f$ corresponds to the cone generated by the 2-dimensional $\operatorname{gr} f$:

$$(u, v, w) \in \operatorname{cl} \operatorname{gr} \operatorname{per} f \quad \Leftrightarrow \quad (v, u, w) \in \operatorname{cl} \operatorname{cone} \operatorname{gr} f. \tag{5.65}$$

Initially, one might think to try to use a bivariate PWL formulation to approximate $\operatorname{gr} \operatorname{per} f$, but this set is unbounded so these techniques do not directly apply. Even if the perspective variable v is bounded by other constraints, the bivariate PWL formulations are significantly larger and more complicated than univariate PWL formulations. By leveraging the conic structure of this set, we can do better than a naive bivariate discretization. Indeed, the perspective function is 'effectively univariate' in the sense that every point in $\operatorname{gr} \operatorname{per} f$ is a ray that is in one-to-one correspondence with a point in $\operatorname{gr} f$.

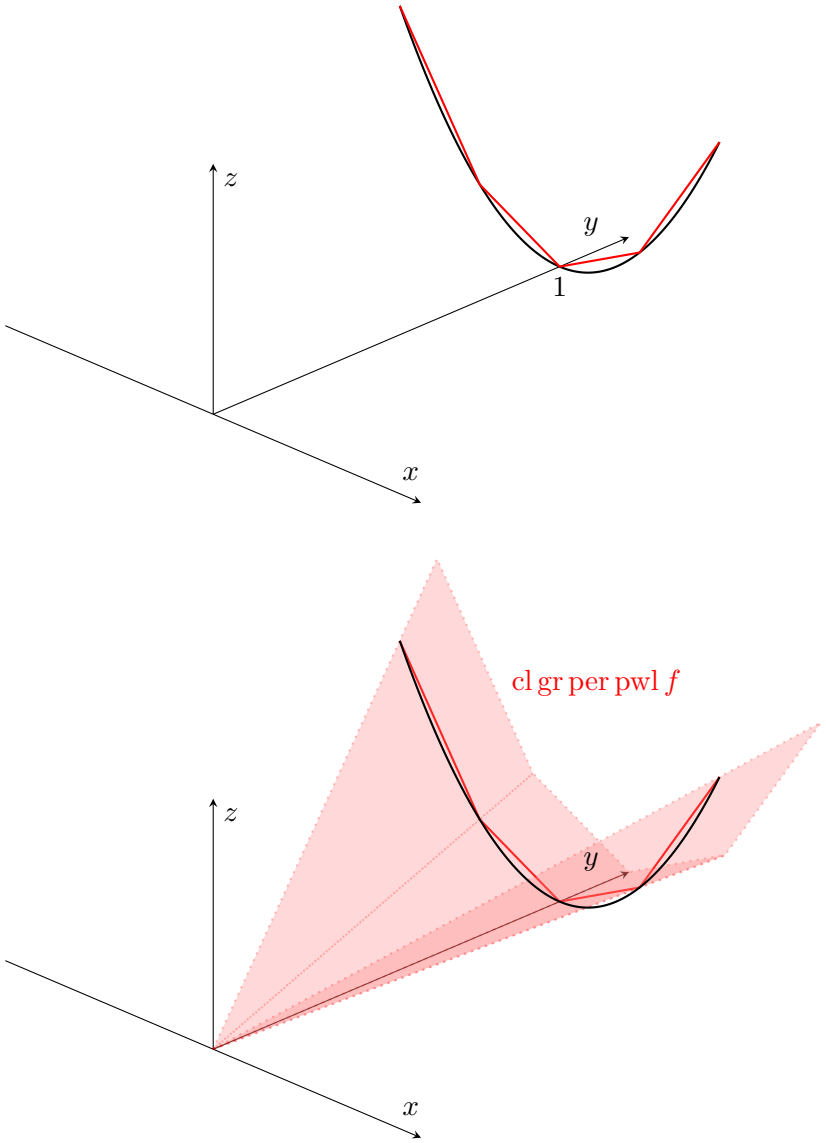
Suppose we have L increasing points on the real line, $(\omega_l)_{l \in [\![L]\!]}$, and we are only concerned with $\operatorname{per} f$ on the cone generated by the domain $[\omega_1, \omega_L]$, i.e. on the set of rays $\{(1, w) : w \in [\omega_1, \omega_L]\}$. On this domain, $\operatorname{gr} \operatorname{per} \operatorname{pwl} f$ corresponds to the cone generated by $\operatorname{gr} \operatorname{pwl} f$, and we can write:

$$\operatorname{cl} \operatorname{gr} \operatorname{per} \operatorname{pwl} f \coloneqq \operatorname{cone} \cup_{l \in [\![L-1]\!]} \operatorname{conv}\{(\omega_l, f(\omega_l)), (\omega_{l+1}, f(\omega_{l+1}))\} \tag{5.66a}$$

$$= \cup_{l \in [\![L-1]\!]} \operatorname{conv}\{\operatorname{cone}\{(\omega_l, f(\omega_l))\}, \operatorname{cone}\{(\omega_{l+1}, f(\omega_{l+1}))\}\}. \tag{5.66b}$$

This is a finite union of convex hulls of pairs of rays, i.e. a union of polyhedra with different recession cones, so it is not MILP-representable in general. See Figure 5.6 for an illustration.

Figure 5.6: The cone cl gr per pwl f generated by cl gr pwl f, for the convex function $f(w) = w^2$.



Similarly to the univariate graph case, we can write a sandwiching relaxation of the nonconvex constraint $(u, v, w) \in \text{cl gr per } f$, for $(v, w) \in \operatorname{cone}[\omega_1, \omega_L]$. Suppose cl epi per f is a proper cone. Since cl gr per $f \subset$ cl epi per $f \cap$ cl hypo per pwl f on the domain of interest, we can relax the nonconvex constraint to the conic constraint $(u, v, w) \in \text{cl epi per } f$ and the homogenized PWL hypograph constraint $(u, v, w) \in \text{cl hypo per pwl } f$.

We write an SOS2-linear formulation for $(u, v, w) \in \text{cl hypo per pwl } f$ on $(v, w) \in \operatorname{cone}[\omega_1, \omega_L]$ by homogenizing (with the perspective variable v) the SOS2-linear formulation for $(u, w) \in \text{hypo pwl } f$

in (5.63):

$$\sum_{l\in[\![L]\!]}\sigma_l f(\omega_l) \geq u, \tag{5.67a}$$
$$\sigma'\omega = w, \tag{5.67b}$$
$$e'\sigma = v, \tag{5.67c}$$
$$\sigma \geq 0, \tag{5.67d}$$
$$\sigma \in \text{SOS2}\,. \tag{5.67e}$$

Note that only constraint (5.63c) contains an affine constant (of 1), so this is the only constraint modified by homogenization with v.

Although this SOS2-linear formulation (5.67) does not have an MILP reformulation in general, we show in Section 5.6.4 that it is MI-conic-representable. Furthermore, if the perspective variable v is bounded above, it is MILP-representable. We can use the homogenized PWL formulations to formulate disjunctions of PWL constraints, by applying the copies-of-variables techniques in Section 5.6.1. In this case, v is a binary variable, like the x_i variables in (5.61).

5.6.4 Mixed-integer conic reformulations

There can be practical benefits to using mixed-integer formulations rather than SOS2 formulations. Performance often scales worse for SOS2 formulations than for good mixed-integer formulations (when they exist). We present seven MI-conic formulations for an SOS2 constraint $\sigma \in \text{SOS2}$.

These formulations modify/generalize either the *convex combinations* (CC) formulation or the *logarithmic independent branching* (AKA *LogIB*) formulation discussed in Vielma, Ahmed, and Nemhauser (2010) and Huchette and Vielma (2017). The CC formulations use $L-1$ auxiliary binary variables β and $L+1$ convex constraints. The LogIB formulations use only a logarithmic number $M = \lceil \log_2(L-1) \rceil$ of auxiliary binary variables β and $2M$ convex constraints. This is an optimal number of binary variables in the sense that we cannot formulate a union of $L-1$ convex sets with fewer than M binary variables in general. In practice for standard PWL constraints, LogIB formulations tend to have better performance than CC formulations, and we expect that this performance advantage should hold for the homogenized PWL formulations.

First, we present two formulations for $\sigma \in \text{SOS2}$ assuming σ is unconstrained. Since there exists no MILP formulation for a union of unbounded polyhedra, these formulations must involve non-polyhedral convex constraints. Similar to Hijazi and Liberti (2016) and Lubin, Vielma, and Zadik (2022), we use standard rotated second order conic constraints. Recall $\mathcal{K}_{\text{sqr}}$ is defined as:

$$\mathcal{K}_{\text{sqr}} := \left\{(u,v,w) \in \mathbb{R}_{\geq} \times \mathbb{R}_{\geq} \times \mathbb{R}^d : 2uv \geq \|w\|^2\right\}. \tag{5.68}$$

The formulations also introduce an auxiliary unbounded epigraph variable $t \in \mathbb{R}$.

The *free convex combinations* (or *FCC*) MISOCP formulation is:

$$e'\beta \leq 1, \tag{5.69a}$$

$$(t, \beta_1, \sigma_1) \in \mathcal{K}_{\text{sqr}}, \tag{5.69b}$$
$$(t, \beta_{l-1} + \beta_l, \sigma_l) \in \mathcal{K}_{\text{sqr}}, \qquad \forall l \in \{2, \dots, L-1\}, \tag{5.69c}$$
$$(t, \beta_{L-1}, \sigma_L) \in \mathcal{K}_{\text{sqr}}, \tag{5.69d}$$
$$\beta \in \{0,1\}^{L-1}. \tag{5.69e}$$

The *free log independent branching* (or *FLogIB*) MISOCP formulation is:

$$\left(t, \beta_o, (\sigma_l)_{l \in \mathcal{H}^1_o}\right) \in \mathcal{K}_{\text{sqr}}, \qquad \forall o \in [\![M]\!], \tag{5.70a}$$
$$\left(t, 1 - \beta_o, (\sigma_l)_{l \in \mathcal{H}^0_o}\right) \in \mathcal{K}_{\text{sqr}}, \qquad \forall o \in [\![M]\!], \tag{5.70b}$$
$$\beta \in \{0,1\}^M, \tag{5.70c}$$

where $\mathcal{H}^1_o, \mathcal{H}^0_o \subset [\![L]\!]$ for $o \in [\![M]\!]$ correspond to particular subsets of σ variables (from reflected Gray codes); see Vielma, Ahmed, and Nemhauser (2010). For each $o \in [\![M]\!]$, these constraints (in combination with the other constraints on σ) enforce:

$$\left(\sigma_l = 0, \forall l \in \mathcal{H}^1_o\right) \vee \left(\sigma_l = 0, \forall l \in \mathcal{H}^0_o\right). \tag{5.71}$$

Note the FCC formulation uses only three-dimensional $\mathcal{K}_{\text{sqr}}$ cones, but the FLogIB $\mathcal{K}_{\text{sqr}}$ cones may be higher dimensional.

Next, we assume that σ is nonnegative (as in the PWL formulations), which allows us to improve the FCC formulation by reducing the dimension of each $\mathcal{K}_{\text{sqr}}$ cone in (5.70) to three. The *nonnegative log independent branching* (or *NLogIB*) MISOCP formulation is:

$$\left(t, \beta_o, \textstyle\sum_{l \in \mathcal{H}^1_o} \sigma_l\right) \in \mathcal{K}_{\text{sqr}}, \qquad \forall o \in [\![M]\!], \tag{5.72a}$$
$$\left(t, 1 - \beta_o, \textstyle\sum_{l \in \mathcal{H}^0_o} \sigma_l\right) \in \mathcal{K}_{\text{sqr}}, \qquad \forall o \in [\![M]\!], \tag{5.72b}$$
$$\beta \in \{0,1\}^M. \tag{5.72c}$$

This is the first logarithmic-sized MI-convex formulation for a finite union of convex sets with different recession cones (we presented this formulation during the talk by Coey (2018)).

Next, we assume that σ is nonnegative and sums to the perspective variable $v \leq 1$ (as in the homogenized PWL formulation (5.67)). This implies $\sigma \in [0,1]^L$, so we have a finite union of bounded polyhedra, hence we can write MILP formulations. The *bounded convex combinations* (or *BCC*) MILP formulation is:

$$e'\beta = 1, \tag{5.73a}$$
$$\sigma_1 \leq \beta_1, \tag{5.73b}$$
$$\sigma_l \leq \beta_{l-1} + \beta_l \qquad \forall l \in \{2, \dots, L-1\}, \tag{5.73c}$$
$$\sigma_L \leq \beta_{L-1}, \tag{5.73d}$$
$$\beta \in \{0,1\}^{L-1}. \tag{5.73e}$$

The *bounded log independent branching* (or *BLogIB*) MILP formulation is:

$$\sum_{l \in \mathcal{H}_o^1} \sigma_l \leq \beta_o \qquad \forall o \in [\![M]\!], \tag{5.74a}$$

$$\sum_{l \in \mathcal{H}_o^0} \sigma_l \leq 1 - \beta_o \qquad \forall o \in [\![M]\!], \tag{5.74b}$$

$$\beta \in \{0,1\}^M. \tag{5.74c}$$

Finally, we impose the additional assumption that v is binary, i.e. we have $e'\sigma = v \in \{0,1\}$. We improve the BCC formulation (5.73) by homogenizing the constraint (5.73a) with v to get the *disjunctive convex combinations* (or *DCC*) MILP formulation:

$$e'\beta \leq v, \tag{5.75a}$$

$$\sigma_1 \leq \beta_1, \tag{5.75b}$$

$$\sigma_l \leq \beta_{l-1} + \beta_l \qquad \forall l \in \{2, \ldots, L-1\}, \tag{5.75c}$$

$$\sigma_L \leq \beta_{L-1}, \tag{5.75d}$$

$$\beta \in \{0,1\}^{L-1}. \tag{5.75e}$$

Note that $v = 1$ recovers the BCC formulation, and $v = 0$ forces all binary variables β to zero, hence no further branching is needed.

Similarly, for the BLogIB formulation in (5.74), we homogenize (5.74b) to get the *disjunctive log independent branching* (or *DLogIB*) MILP formulation:

$$\sum_{l \in \mathcal{H}_o^1} \sigma_l \leq \beta_o \qquad \forall o \in [\![M]\!], \tag{5.76a}$$

$$\sum_{l \in \mathcal{H}_o^0} \sigma_l \leq v - \beta_o \qquad \forall o \in [\![M]\!], \tag{5.76b}$$

$$\beta \in \{0,1\}^M. \tag{5.76c}$$

Note that $v = 1$ recovers the BLogIB formulation. Furthermore, (5.76b) and the nonnegativity of σ implies that $v \geq \beta_o, \forall o \in [\![M]\!]$. Thus both the DCC and DLogIB formulations have the nice branching property that $v = 0$ forces $\beta = 0$.

5.6.5 Examples for disjunctive formulations and nonconvex relaxations

5.6.5.1 Ball packing

We choose the centers and radii of m ℓ_2 norm balls in n-dimensional space, maximizing the sum of the radii such that the balls are non-overlapping and fit inside the unit box $[0,1]^n$. For ball $i \in [\![m]\!]$, we let the center be $c_i \in \mathbb{R}^n$ and the radius be $r_i \geq 0$. The high-level nonconvex model is:

$$\max_{r,c} \quad e'r : \tag{5.77a}$$

$$0 \leq r \leq e/2, \tag{5.77b}$$

$$r_i e \leq c_i \leq (1 - r_i)e \qquad \forall i \in [\![m]\!], \tag{5.77c}$$
$$c_{i,1} \leq c_{i+1,1} \qquad \forall i \in [\![m-1]\!], \tag{5.77d}$$
$$r_i + r_j \leq \|c_i - c_j\| \qquad \forall i, j \in [\![m]\!] : i < j. \tag{5.77e}$$

Note (5.77d) simply breaks some of the symmetry in the problem.

For each $i, j \in [\![m]\!] : i < j$, the nonconvex constraint (5.77e) is equivalent to the following EF in terms of auxiliary variables $x_{i,j} \in \mathbb{R}$ and $\lambda_{i,j} \in \mathbb{R}^n$:

$$x_{i,j} = r_i + r_j, \tag{5.78a}$$
$$x_{i,j} \leq e'\lambda_{i,j}, \tag{5.78b}$$
$$(\lambda_{i,j,k}, x_{i,j}, c_{i,k} - c_{j,k}) \in \text{hypo gr per } f \qquad \forall k \in [\![n]\!], \tag{5.78c}$$

where f is the univariate square function, i.e. $f(w) = w^2$. Now the nonconvexity only appears in the three-dimensional hypograph constraints (5.78c). Note $c_{i,k} - c_{j,k} \in [-1, 1], \forall k \in [\![n]\!]$, and the perspective variable is $x_{i,j} \in [0, 1]$.

We apply the homogenized PWL techniques introduced in Section 5.6.3 to formulate a relaxation of (5.78c) for each $k \in [\![n]\!]$. Consider L breakpoints $\omega \in [-1, 1]^L$ (e.g. linearly spaced). In terms of auxiliary SOS2 variables $\sigma_{i,j,k} \in \mathbb{R}^L$, the SOS2-linear relaxation is:

$$\omega'\sigma_{i,j,k} = c_{i,k} - c_{j,k}, \tag{5.79a}$$
$$\textstyle\sum_{l \in [\![L]\!]} \omega_l^2 \sigma_{i,j,k,l} = \lambda_{i,j,k}, \tag{5.79b}$$
$$e'\sigma_{i,j,k} = x_{i,j}, \tag{5.79c}$$
$$\sigma_{i,j,k} \geq 0, \tag{5.79d}$$
$$\sigma_{i,j,k} \in \text{SOS2}\,. \tag{5.79e}$$

To (optionally) obtain an MI-conic relaxation, we can replace the SOS2 constraints (5.79e) with an MI-conic formulation from Section 5.6.4. Since $\sigma_{i,j,k}$ is constrained by (5.79c) and (5.79d) and $x_{i,j} \in [0, 1]$, we can use the bounded MILP formulations *BCC* and *BLogIB*.

For our experiments, we generate random instances with $m = 4$ balls in $n = 3$ dimensions, with various numbers of breakpoints L (linearly spaced) in each of the $nm(m-1)/2 = 18$ three-dimensional homogenized piecewise linear constraints. Since the formulations are linear, there is no need for outer approximation (hence the number of iterations is zero), so we report results from Gurobi directly, including the number of branch-and-bound nodes Gurobi uses. We run the *SOS2* formulation itself (which is not an MILP but can be handled by Gurobi) and the *BCC* and *BLogIB* MILP formulations. All instances are feasible and bounded. We verify near-feasibility of the solutions returned by checking the original nonconvex norm constraints (5.77e) are almost satisfied.

Our results are summarized in Table 5.13. For each instance, Gurobi either converges or hits a time limit. *BCC* only solves the smallest two sizes, *SOS2* solves the first five, and *BLogIB* solves the first nine. Furthermore, *BLogIB* achieves the fastest solve times on all instances solved. The

number of nodes Gurobi explores stays relatively constant across the sizes for *BLogIB*, but generally seems to increase for the other two formulations (on the instances they converge on).

Table 5.13: Ball packing solver statistics.

	SOS2			BLogIB			BCC		
pts	st	nodes	time	st	nodes	time	st	nodes	time
7	co	123085	1.8	co	106705	1.6	co	171052	5.8
15	co	276230	8.1	co	167918	2.8	co	1785870	45
31	co	414870	17	co	120577	3.1	tl	15856119	600
63	co	1129736	131	co	98307	5.4	tl	5630500	600
127	co	617139	197	co	94581	12	tl	469856	600
255	tl	17987954	600	co	48040	14	tl	10950	600
511	tl	10183066	600	co	88807	38	tl	1041167	600
1023	tl	3675673	600	co	134039	161	tl	173870	600
2047	tl	1106535	600	co	132643	562	tl	11399	600
4095	tl	384702	600	tl	66445	600	tl	469	600
8191	tl	137973	601	tl	26286	601	tl	79	602

5.6.5.2 Modular design with convex constraints

We design a modular device (e.g. a robot) by selecting parts (e.g. a battery, actuators, sensors) from a catalog. For each module, we choose one corresponding part, satisfying constraints associated with that particular part. These constraints involve certain nonnegative global design variables (e.g. weight, torque, lifetime, wattage). We minimize the cost of the parts plus a linear function of the design variables. Suppose there are n design variables $y \in \mathbb{R}^n$, which are associated with costs vector d and are bounded by below by $y_{\min} \geq 0$ and above by $y_{\max}$. Suppose there are m modules and p different part choices per module. Let $x_{i,j}, \forall i \in [\![m]\!], j \in [\![p]\!]$ be a binary variable equal to one if and only if the jth part is selected for the ith module, at a cost of $c_{i,j}$. When part (i,j) is selected, it imposes certain constraints on the design variables; we represent these constraints as $y \in \mathcal{C}_{i,j} \subset \mathbb{R}^n$. In this section, we suppose that each set $\mathcal{C}_{i,j}$ is convex, and in the following section Section 5.6.5.3, we suppose $\mathcal{C}_{i,j}$ is nonconvex.

The high-level model is:

$$\min_{x,y} \quad d'y + \textstyle\sum_{i\in[\![m]\!], j\in[\![p]\!]} c_{i,j}x_{i,j} : \tag{5.80a}$$

$$x_{i,j} \in \{0,1\} \qquad \forall i \in [\![m]\!], j \in [\![p]\!], \tag{5.80b}$$

$$\textstyle\sum_{j\in[\![p]\!]} x_{i,j} = 1 \qquad \forall i \in [\![m]\!], \tag{5.80c}$$

$$y_{\min} \leq y \leq y_{\max}, \tag{5.80d}$$

$$x_{i,j} = 1 \quad \Rightarrow \quad y \in \mathcal{C}_{i,j} \qquad \forall i \in [\![m]\!], j \in [\![p]\!]. \tag{5.80e}$$

Suppose that for each $i \in [\![m]\!], j \in [\![p]\!]$, the set $\mathcal{C}_{i,j} \subset \mathbb{R}^n$ is convex and representable as follows:

$$y \in \mathcal{C}_{i,j} \quad \Leftrightarrow \quad b_{i,j} - a'_{i,j} y \geq f_{i,j}(h_{i,j} - G_{i,j} y), \tag{5.81}$$

where $f_{i,j} : \mathbb{R}^K \to \mathbb{R}$ is a convex function, $b_{i,j} \in \mathbb{R}$, $a_{i,j} \in \mathbb{R}^n$, $h_{i,j} \in \mathbb{R}^K$, and $G_{i,j} \in \mathbb{R}^{K \times n}$ for some $K \geq 1$. Suppose further that $f_{i,j}$ is a nonhomogeneous separable spectral function, such as one of the MMD functions discussed in Section 3.6, and let $\mathcal{K}_{i,j}$ represent the corresponding proper separable spectral vector function cone:

$$\mathcal{K}_{i,j} \coloneqq \mathcal{K}_{\text{sep}}(f_{i,j}) = \operatorname{cl} \operatorname{epi} \operatorname{per} f_{i,j}. \tag{5.82}$$

Then a conic formulation for $y \in \mathcal{C}_{i,j}$ is:

$$\left(b_{i,j} - a'_{i,j} y, 1, h_{i,j} - G_{i,j} y\right) \in \mathcal{K}_{i,j} \subset \mathbb{R}^{2+K}. \tag{5.83}$$

For the disjunctive implication constraints (5.80e), we apply the tight conic copies-of-variables formulation discussed in Section 5.6.1 with auxiliary copy variables $z_{i,j} \in \mathbb{R}^n$. Using the disjunctive variables $x_{i,j}$, we homogenize the bounds (5.80d) and the affine functions in (5.83). This yields the conic form model:

$$\begin{aligned}
\min_{x,y,z} \quad & d'y + \textstyle\sum_{i \in [\![m]\!], j \in [\![p]\!]} c_{i,j} x_{i,j} : && && \text{(5.84a)}\\
& \textstyle\sum_{j \in [\![p]\!]} x_{i,j} = 1 && \forall i \in [\![m]\!], && \text{(5.84b)}\\
& \textstyle\sum_{j \in [\![p]\!]} z_{i,j} = y && \forall i \in [\![m]\!], && \text{(5.84c)}\\
& z_{i,j} - y_{\min} x_{i,j} \in \mathbb{R}^n_{\geq} && \forall i \in [\![m]\!], j \in [\![p]\!], && \text{(5.84d)}\\
& y_{\max} x_{i,j} - z_{i,j} \in \mathbb{R}^n_{\geq} && \forall i \in [\![m]\!], j \in [\![p]\!], && \text{(5.84e)}\\
& \left(b_{i,j} x_{i,j} - a'_{i,j} z_{i,j}, x_{i,j}, h_{i,j} x_{i,j} - G_{i,j} z_{i,j}\right) \in \mathcal{K}_{i,j} && \forall i \in [\![m]\!], j \in [\![p]\!], && \text{(5.84f)}\\
& x_{i,j} \in \{0,1\} && \forall i \in [\![m]\!], j \in [\![p]\!]. && \text{(5.84g)}
\end{aligned}$$

We generate random instances of (5.84) with $m = 6$ modules and $p = 5$ part options per module, with various numbers of design variables n and convex function dimension $K = \lfloor n/2 \rfloor$. For $i \in [\![m]\!], j \in [\![p]\!]$, we select each convex function $f_{i,j} : \mathbb{R}^K_{\geq} \to \mathbb{R}$ randomly from a list of MMD functions (see Section 3.6): *NegEntropy*, *NegSqrt*, *NegPower*(0.8), *Power*(2.0). The cost parameters $d, c_{i,j}$, variable bounds $y_{\min}, y_{\max}$, and affine data for the conic constraints (including sparse matrices $G_{i,j}$) are generated to ensure the instances are feasible and bounded. We verify that solutions returned by MOIPajarito are feasible for the disjunctions of convex constraints.

Our results are summarized in Table 5.14. *nat* uses the NF for Hypatia and the EF in the OA model, and *noext* differs in that it uses the NF in the OA model. The timings and iteration counts are very similar across all sizes, indicating that for these instances, there is no benefit to using EFs. The very low iteration counts indicate that the continuous relaxations of our MI-conic copies-of-variables formulations are very strong (as the theory predicts). For example, for $n = 20$,

MOIPajarito needs no OA iterations, as Hypatia's continuous relaxation solution is detected to be integral to the desired tolerances. For several instances, we notice that MOIPajarito iterations repeat the same integral solution, simply adding rounds of separation cuts to improve the objective bound until the desired optimality gap is reached.

Table 5.14: Modular design with convex constraints solver statistics.

	nat			noext		
n	st	it	time	st	it	time
20	co	0	0.3	co	0	0.2
40	co	1	5.2	co	1	1.9
60	co	1	5.4	co	1	5.3
80	co	2	18	co	2	18
100	co	3	79	co	3	81
120	co	4	167	co	4	143
140	co	1	99	co	1	89
160	co	5	358	co	6	571
180	co	1	164	co	1	136
200	co	1	373	co	1	375
220	tl	3	602	tl	3	604

5.6.5.3 Modular design with nonconvex constraints

As in Section 5.6.5.2, we start with the high-level modular design model (5.80). However, now we assume that for each $i \in [\![m]\!], j \in [\![p]\!]$, the set $\mathcal{C}_{i,j} \subset \mathbb{R}^n$ is nonconvex. Specifically, we change the epigraph constraint (5.81) to a graph constraint:

$$y \in \mathcal{C}_{i,j} \quad \Leftrightarrow \quad b_{i,j} - a'_{i,j} y = f_{i,j}(h_{i,j} - G_{i,j} y). \tag{5.85}$$

Since $f_{i,j} : \mathbb{R}^K \to \mathbb{R}$ is still assumed to be a convex separable spectral function, we can write a disaggregated formulation for (5.85) in terms of auxiliary variables $\lambda_{i,j} \in \mathbb{R}^K$ and $w_{i,j} \in \mathbb{R}^K$:

$$w_{i,j} = h_{i,j} - G_{i,j} y, \tag{5.86a}$$
$$b_{i,j} - a'_{i,j} y = e' \lambda_{i,j}, \tag{5.86b}$$
$$\lambda_{i,j,k} = f_{i,j}(w_{i,j,k}) \qquad \forall k \in [\![K]\!]. \tag{5.86c}$$

Now the nonconvexity only appears in each 2-dimensional graph constraint (5.86c).

Using the PWL techniques in Section 5.6.2, we can formulate a sandwiching relaxation for each graph constraint (5.86c). We first compute lower and upper bounds on each $w_{i,j,k}$, which are finite since we have finite bounds on y. We intersect these bounds with the natural domain of $f_{i,j}$ and select L breakpoints $\omega_{i,j,k,l}, \forall l \in [\![L]\!]$ between these bounds (e.g. linearly spaced). For $k \in [\![K]\!]$, the SOS2-convex sandwiching relaxation of (5.86c) is:

$$\lambda_{i,j,k} \geq f_{i,j}(w_{i,j,k}), \tag{5.87a}$$

$$\lambda_{i,j,k} \leq \sum_{l \in L} f_{i,j}(\omega_{i,j,k,l}) \sigma_{i,j,k,l}, \tag{5.87b}$$
$$\omega'_{i,j,k} \sigma_{i,j,k} = w_{i,j,k}, \tag{5.87c}$$
$$e' \sigma_{i,j,k} = 1, \tag{5.87d}$$
$$\sigma_{i,j,k} \geq 0, \tag{5.87e}$$
$$\sigma_{i,j,k} \in \text{SOS2}\,. \tag{5.87f}$$

So we can write an SOS2-convex relaxation of $y \in \mathcal{C}_{i,j}$, but we have yet to formulate the disjunctions over these constraints for each module/part. We again apply the copies-of-variables formulation from Section 5.6.1, introducing auxiliary copy variables $z_{i,j}$ of y. We homogenize the bound constraints and the sandwiching relaxation constraints (5.87). Following the homogenized PWL techniques introduced in Section 5.6.3, we get three-dimensional proper cone constraints from (5.87a) and SOS2-linear constraints from the PWL hypograph constraints in (5.87). The resulting SOS2-conic model is:

$$\min_{x,y,z,w,\lambda,\sigma} \quad d'y + \sum_{i \in [\![m]\!], j \in [\![p]\!]} c_{i,j} x_{i,j} : \tag{5.88a}$$
$$\sum_{j \in [\![p]\!]} x_{i,j} = 1 \qquad \forall i \in [\![m]\!], \tag{5.88b}$$
$$\sum_{j \in [\![p]\!]} z_{i,j} = y \qquad \forall i \in [\![m]\!], \tag{5.88c}$$
$$b_{i,j} x_{i,j} - a'_{i,j} z_{i,j} = e' \lambda_{i,j} \qquad \forall i \in [\![m]\!], j \in [\![p]\!], \tag{5.88d}$$
$$h_{i,j} x_{i,j} - G_{i,j} z_{i,j} = w_{i,j} \qquad \forall i \in [\![m]\!], j \in [\![p]\!], \tag{5.88e}$$
$$e' \sigma_{i,j,k} = x_{i,j} \qquad \forall i \in [\![m]\!], j \in [\![p]\!], k \in [\![K]\!], \tag{5.88f}$$
$$\omega'_{i,j,k} \sigma_{i,j,k} = w_{i,j,k} \qquad \forall i \in [\![m]\!], j \in [\![p]\!], k \in [\![K]\!], \tag{5.88g}$$
$$z_{i,j} - y_{\min} x_{i,j} \in \mathbb{R}^n_{\geq} \qquad \forall i \in [\![m]\!], j \in [\![p]\!], \tag{5.88h}$$
$$y_{\max} x_{i,j} - z_{i,j} \in \mathbb{R}^n_{\geq} \qquad \forall i \in [\![m]\!], j \in [\![p]\!], \tag{5.88i}$$
$$(\lambda_{i,j,k}, x_{i,j}, w_{i,j,k}) \in \mathcal{K}_{i,j} \qquad \forall i \in [\![m]\!], j \in [\![p]\!], k \in [\![K]\!], \tag{5.88j}$$
$$\sum_{l \in L} f_{i,j}(\omega_{i,j,k,l}) \sigma_{i,j,k,l} - \lambda_{i,j,k} \in \mathbb{R}_{\geq} \qquad \forall i \in [\![m]\!], j \in [\![p]\!], k \in [\![K]\!], \tag{5.88k}$$
$$\sigma_{i,j,k} \in \mathbb{R}^L_{\geq} \qquad \forall i \in [\![m]\!], j \in [\![p]\!], k \in [\![K]\!], \tag{5.88l}$$
$$\sigma_{i,j,k} \in \text{SOS2} \qquad \forall i \in [\![m]\!], j \in [\![p]\!], k \in [\![K]\!], \tag{5.88m}$$
$$x_{i,j} \in \{0,1\} \qquad \forall i \in [\![m]\!], j \in [\![p]\!]. \tag{5.88n}$$

Note that in (5.88j), the proper cones $\mathcal{K}_{i,j}$ (defined in (5.82)) are three-dimensional, whereas in the convex case in Section 5.6.5.2 they were $(2+K)$-dimensional. To (optionally) obtain an MI-conic formulation of (5.88), we can replace the SOS2 constraints (5.88m) with an MI-conic formulation from Section 5.6.4. Note that for each $i \in [\![m]\!], j \in [\![p]\!], k \in [\![K]\!]$, the $\sigma_{i,j,k}$ variables are nonnegative and sum to the binary variable $x_{i,j}$ by (5.88f). Thus we can use either of the disjunctive SOS2 MILP formulations - *DCC* and *DLogIB* - from (5.88).

We generate random instances with $m = 3$ modules and $p = 3$ part options per module, with

various numbers of design variables n. As in the convex constraints case in Section 5.6.5.2, we let $K = \lfloor n/2 \rfloor$ and select the functions $f_{i,j} : \mathbb{R}^K_{\geq} \to \mathbb{R}$ randomly from the same list of four MMD functions. We use $L = 512$ breakpoints (linearly spaced) in each of the $mpK = 9\lfloor n/2 \rfloor$ three-dimensional homogenized piecewise linear constraints.

We run three formulations for (5.88): the *SOS2* formulation itself, and the *DCC* and *DLogIB* MI-conic formulations. To enable MOIPajarito to handle SOS2 constraints, we turn off conic subproblem solves and use separation cuts instead. We do this for all three formulations to ensure fair comparisons, though it is not necessary for the MI-conic formulations. All instances are feasible and bounded. We verify that solutions returned by MOIPajarito are near-feasible for the disjunctions of nonconvex constraints.

Our results are summarized in Table 5.15. For each instance, MOIPajarito either converges or hits a time limit. *DCC* only solves five small instances, and is the slowest. *SOS2* and *DLogIB* converge up to size $n = 22$, and *DLogIB* additionally converges on $n = 26$. Up to $n = 12$, *SOS2* is generally faster than *DLogIB*, but *DLogIB* is usually faster for larger sizes. The number of iterations varies across the formulations without a clear pattern. We would expect this to be more consistent because the formulations are equivalent, but since Gurobi does not solve the OA model exactly at each iteration of MOIPajarito, we can get slightly different OA solutions from Gurobi.

Table 5.15: Modular design with nonconvex constraints solver statistics.

	SOS2			DLogIB			DCC		
n	st	it	time	st	it	time	st	it	time
4	co	7	4.2	co	8	9.9	co	7	95
6	co	15	7.6	co	14	13	co	14	166
8	co	26	27	co	15	30	tl	18	600
10	co	13	17	co	16	28	co	21	483
12	co	36	268	co	53	395	co	13	144
14	co	28	92	co	24	84	co	25	475
16	co	21	447	co	23	333	tl	10	600
18	co	27	195	co	27	206	tl	12	600
20	co	40	475	co	28	273	*	*	*
22	co	35	480	co	33	219	*	*	*
24	tl	13	600	tl	21	600	*	*	*
26	tl	61	600	co	23	149	*	*	*
28	tl	36	600	tl	28	600	*	*	*
30	tl	43	600	tl	61	600	*	*	*

Bibliography

Agrawal, Akshay, Steven Diamond, and Stephen Boyd (2019). "Disciplined geometric programming". In: *Optimization Letters* 13.5, pp. 961–976.

Ahmadi, Amir Ali and Georgina Hall (2015). "Sum of squares basis pursuit with linear and second order cone programming". In: *arXiv preprint arXiv:1510.01597*. eprint: 1510.01597.

Andersen, Erling D, Cornelis Roos, and Tamas Terlaky (2003). "On implementing a primal-dual interior-point method for conic quadratic optimization". In: *Mathematical Programming* 95.2, pp. 249–277.

Andersen, Martin et al. (2011). "Interior-point methods for large-scale cone programming". In: *Optimization for Machine Learning* 5583.

Andersen, Martin S, Joachim Dahl, and Lieven Vandenberghe (2013). "Logarithmic barriers for sparse matrix cones". In: *Optimization Methods and Software* 28.3, pp. 396–423.

Aylward, Erin M, Pablo A Parrilo, and Jean-Jacques E Slotine (2008). "Stability and robustness analysis of nonlinear systems via contraction metrics and SOS programming". In: *Automatica* 44.8, pp. 2163–2170.

Baes, Michel (2007). "Convexity and differentiability properties of spectral functions and spectral mappings on Euclidean Jordan algebras". In: *Linear algebra and its applications* 422.2-3, pp. 664–700. DOI: 10.1016/j.laa.2006.11.025.

Belotti, Pietro et al. (May 2013). "Mixed-integer nonlinear optimization". In: *Acta Numerica* 22, pp. 1–131. ISSN: 1474-0508. DOI: 10.1017/s0962492913000032.

Ben-Tal, Ahron and Arkadi Nemirovski (2001). *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Vol. 2. SIAM.

Bertsimas, Dimitris, Jourdain Lamperski, and Jean Pauphilet (2020). "Certifiably optimal sparse inverse covariance estimation". In: *Mathematical Programming* 184.1, pp. 491–530.

Bezanson, Jeff et al. (2017). "Julia: A fresh approach to numerical computing". In: *SIAM Review* 59.1, pp. 65–98.

Blekherman, Grigoriy (2012). "Nonnegative polynomials and sums of squares". In: *Journal of the American Mathematical Society* 25.3, pp. 617–635.

Bonami, Pierre, Lorenz T. Biegler, et al. (2008). "An algorithmic framework for convex mixed integer nonlinear programs". In: *Discrete Optimization* 5.2, pp. 186–204. ISSN: 1572-5286. DOI: 10.1016/j.disopt.2006.10.011.

Bonami, Pierre, Mustafa Kılınç, and Jeff Linderoth (2012). "Algorithms and Software for Convex Mixed Integer Nonlinear Programs". English. In: *Mixed Integer Nonlinear Programming*. Ed. by Jon Lee and Sven Leyffer. Vol. 154. The IMA Volumes in Mathematics and its Applications. Springer New York, pp. 1–39. ISBN: 978-1-4614-1926-6. DOI: 10.1007/978-1-4614-1927-3_1.

Borchers, Brian (1999). "CSDP, A C library for semidefinite programming". In: *Optimization Methods and Software* 11.1-4, pp. 613–623.

Boyd, Stephen (2009). *EE363 Review Session 4: Linear Matrix Inequalities*. University Lecture. URL: https://stanford.edu/class/ee363/sessions/s4notes.pdf.

Boyd, Stephen, Laurent El Ghaoui, et al. (1994). *Linear matrix inequalities in system and control theory*. Vol. 15. SIAM.

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex optimization*. Cambridge University Press.

Burer, Samuel (2003). "Semidefinite programming in the space of partial positive semidefinite matrices". In: *SIAM Journal on Optimization* 14.1, pp. 139–172.

Burkardt, John (2016). *Polynomials for Global Optimization Tests*. Accessed: 2021-03-22. URL: https://people.sc.fsu.edu/~jburkardt/py_src/polynomials/polynomials.html.

Carlen, Eric (2010). "Trace inequalities and quantum entropy: an introductory course". In: *Entropy and the quantum* 529, pp. 73–140. DOI: 10.1090/conm/529/10428.

Chandrasekaran, Venkat and Parikshit Shah (2016). "Relative entropy relaxations for signomial optimization". In: *SIAM Journal on Optimization* 26.2, pp. 1147–1173.

— (2017). "Relative entropy optimization and its applications". In: *Mathematical Programming* 161.1-2, pp. 1–32.

Chares, Robert (2009). "Cones and interior-point algorithms for structured convex optimization involving powers andexponentials". PhD thesis. Ph. D. Thesis, UCL-Université Catholique de Louvain, Louvain-la-Neuve, Belgium.

Chen, Lisha and Jianhua Z Huang (2012). "Sparse reduced-rank regression for simultaneous dimension reduction and variable selection". In: *Journal of the American Statistical Association* 107.500, pp. 1533–1545.

Chen, Yanqing et al. (2008). "Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate". In: *ACM Transactions on Mathematical Software (TOMS)* 35.3, pp. 1–14.

Coey, Chris (July 2018). "Using algebraic structure to accelerate polyhedral approximation". Unpublished; talk slides, International Symposium on Mathematical Programming (Bordeaux).

Coey, Chris, Lea Kapelevich, and Juan Pablo Vielma (2021a). *Conic optimization with spectral functions on Euclidean Jordan algebras*. arXiv: 2103.04104 [math.OC].

— (2021b). *Hypatia cones reference*. Online; accessed 1-June-2021. URL: https://github.com/chriscoey/Hypatia.jl/wiki/.

— (2021c). *Hypatia documentation*. Online; accessed 7-June-2021. URL: https://chriscoey.github.io/Hypatia.jl/dev/.

— (2021d). *Performance enhancements for a generic conic interior point algorithm*. arXiv: 2107.04262 [math.OC].

Coey, Chris, Lea Kapelevich, and Juan Pablo Vielma (2021e). *Solving natural conic formulations with Hypatia.jl*. arXiv: 2005.01136 [math.OC].

Coey, Chris, Miles Lubin, and Juan Pablo Vielma (2020). "Outer approximation with conic certificates for mixed-integer convex problems". In: *Mathematical Programming Computation* 12, pp. 249–293.

d'Aspremont, Alexandre et al. (2007). "A direct formulation for sparse PCA using semidefinite programming". In: *SIAM review* 49.3, pp. 434–448.

Dahl, Joachim and Erling D Andersen (2021). "A primal-dual interior-point algorithm for nonsymmetric exponential-cone optimization". In: *Mathematical Programming*, pp. 1–30.

Davis, Chandler (1957). "All convex invariant functions of Hermitian matrices". In: *Archiv der Mathematik* 8.4, pp. 276–278. DOI: 10.1007/bf01898787.

Deng, Chun Yuan (2011). "A generalization of the Sherman–Morrison–Woodbury formula". In: *Applied Mathematics Letters* 24.9, pp. 1561–1564. DOI: 10.1016/j.aml.2011.03.046.

Diamond, Steven and Stephen Boyd (2016). "CVXPY: A Python-embedded modeling language for convex optimization". In: *The Journal of Machine Learning Research* 17.1, pp. 2909–2913.

Dolan, Elizabeth D and Jorge J Moré (2002). "Benchmarking optimization software with performance profiles". In: *Mathematical programming* 91.2, pp. 201–213.

Domahidi, Alexander, Eric Chu, and Stephen Boyd (2013). "ECOS: An SOCP solver for embedded systems". In: *2013 European Control Conference (ECC)*. IEEE, pp. 3071–3076.

Drewes, Sarah and Stefan Ulbrich (2012). "Subgradient Based Outer Approximation for Mixed Integer Second Order Cone Programming". English. In: *Mixed Integer Nonlinear Programming*. Ed. by Jon Lee and Sven Leyffer. Vol. 154. The IMA Volumes in Mathematics and its Applications. Springer New York, pp. 41–59. ISBN: 978-1-4614-1926-6. DOI: 10.1007/978-1-4614-1927-3_2.

Dunning, Iain, Joey Huchette, and Miles Lubin (2017). "JuMP: A modeling language for mathematical optimization". In: *SIAM Review* 59.2, pp. 295–320.

Faraut, Jacques and Adam Koranyi (1998). "Analysis on symmetric cones". In: *Bull. Amer. Math. Soc* 35, pp. 77–86.

Fawzi, Hamza and Omar Fawzi (2018). "Efficient optimization of the quantum relative entropy". In: *Journal of Physics A: Mathematical and Theoretical* 51.15, p. 154003.

Fawzi, Hamza, James Saunderson, and Pablo A Parrilo (2019). "Semidefinite approximations of the matrix logarithm". In: *Foundations of Computational Mathematics* 19.2, pp. 259–296.

Faybusovich, Leonid and Takashi Tsuchiya (2017). "Matrix monotonicity and self-concordance: how to handle quantum entropy in optimization problems". In: *Optimization Letters* 11.8, pp. 1513–1526. DOI: 10.1007/s11590-017-1145-6.

Faybusovich, Leonid and Cunlu Zhou (2021). "Long-step path-following algorithm for quantum information theory: Some numerical aspects and applications". In: *Numerical Algebra, Control & Optimization*. DOI: 10.3934/naco.2021017. eprint: 1906.00037.

Fleming, Philip J and John J Wallace (1986). "How not to lie with statistics: the correct way to summarize benchmark results". In: *Communications of the ACM* 29.3, pp. 218–221.

Friberg, Henrik A (2016). "CBLIB 2014: A benchmark library for conic mixed-integer and continuous optimization". In: *Mathematical Programming Computation* 8.2, pp. 191–214.

Furuta, Takayuki (2008). "Concrete examples of operator monotone functions obtained by an elementary method without appealing to Löwner integral representation". In: *Linear algebra and its applications* 429.5-6, pp. 972–980. DOI: 10.1016/j.laa.2006.11.023.

Gally, Tristan, Marc E. Pfetsch, and Stefan Ulbrich (2018). "A framework for solving mixed-integer semidefinite programs". In: *Optimization Methods and Software* 33.3, pp. 594–632. DOI: 10.1080/10556788.2017.1322081.

Gould, Nicholas and Jennifer Scott (2016). "A note on performance profiles for benchmarking software". In: *ACM Transactions on Mathematical Software (TOMS)* 43.2, pp. 1–5.

Grant, Michael and Stephen Boyd (2014). *CVX: MATLAB software for disciplined convex programming, version 2.1*.

Grant, Michael, Stephen Boyd, and Yinyu Ye (2006). "Disciplined convex programming". In: *Global optimization*. Springer, pp. 155–210.

Güler, Osman (1996). "Barrier functions in interior point methods". In: *Mathematics of Operations Research* 21.4, pp. 860–885.

Güler, Osman and Levent Tunçel (1998). "Characterization of the barrier parameter of homogeneous convex cones". In: *Mathematical programming* 81.1, pp. 55–76.

Gurobi (2022). *Documentation: Choosing the right algorithm*. URL: https://www.gurobi.com/documentation/9.5/refman/choosing_the_right_algorit.html.

Hall, Georgina (2019). "Engineering and business applications of sum of squares polynomials". In: *arXiv preprint arXiv:1906.07961*.

Henrion, Didier and Milan Korda (2013). "Convex computation of the region of attraction of polynomial control systems". In: *IEEE Transactions on Automatic Control* 59.2, pp. 297–312.

Hijazi, Hassan, Pierre Bonami, and Adam Ouorou (2014). "An outer-inner approximation for separable mixed-integer nonlinear programs". In: *INFORMS Journal on Computing* 26.1, pp. 31–44.

Hijazi, Hassan and Leo Liberti (2016). "Constraint qualification failure in action". In: *Operations Research Letters* 44.4, pp. 503–506.

Hsieh, Cho-Jui et al. (2012). "A divide-and-conquer procedure for sparse inverse covariance estimation". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2330–2338.

Huchette, Joey and Juan Pablo Vielma (2017). "Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools". In: *arXiv preprint arXiv:1708.00050*.

Kapelevich, Lea, Chris Coey, and Juan Pablo Vielma (2021). *Sum of squares generalizations for conic sets*. arXiv: 2103.11499 [math.OC].

Karimi, Mehdi and Levent Tunçel (2020a). *Domain-Driven Solver (DDS) Version 2.0: a MATLAB-based Software Package for Convex Optimization Problems in Domain-Driven Form*. arXiv: 1908.03075 [math.OC].

Karimi, Mehdi and Levent Tunçel (2020b). "Primal–Dual Interior-Point Methods for Domain-Driven Formulations". In: *Mathematics of Operations Research* 45.2, pp. 591–621.

Kim, Sunyoung, Masakazu Kojima, and Makoto Yamashita (2003). "Second Order Cone Programming Relaxation of a Positive Semidefinite Constraint". In: *Optimization Methods and Software* 18.5, pp. 535–541. DOI: 10.1080/1055678031000148696.

Korda, Milan, Didier Henrion, and Colin N Jones (2016). "Controller design and value function approximation for nonlinear dynamical systems". In: *Automatica* 67, pp. 54–66.

Kwong, Man Kam (1989). "Some results on matrix monotone functions". In: *Linear Algebra and Its Applications* 118, pp. 129–153. DOI: 10.1016/0024-3795(89)90577-6.

Lasserre, Jean B (1998). "Homogeneous functions and conjugacy". In: *Journal of Convex Analysis* 5, pp. 397–404.

Laurent, Monique and Teresa Piovesan (2015). "Conic approach to quantum graph parameters using linear optimization over the completely positive semidefinite cone". In: *SIAM Journal on Optimization* 25.4, pp. 2461–2493.

Legat, Benoit et al. (2020). "MathOptInterface: a data structure for mathematical optimization problems". In: *arXiv preprint arXiv:2002.03447*.

Legat, Benoît et al. (June 2021). *JuliaPolyhedra/Polyhedra.jl: v0.6.16*. Version v0.6.16. DOI: 10.5281/zenodo.4993670.

Leyffer, Sven (Dec. 1993). "Deterministic Methods for Mixed Integer Nonlinear Programming". PhD thesis. University of Dundee.

Löwner, Karl (1934). "Über monotone matrixfunktionen". In: *Mathematische Zeitschrift* 38.1, pp. 177–216. DOI: 10.1007/bf01170633.

Lubin, Miles, Juan Pablo Vielma, and Ilias Zadik (2022). "Mixed-integer convex representability". In: *Mathematics of Operations Research* 47.1, pp. 720–749.

Lubin, Miles, Emre Yamangil, et al. (2016). "Extended Formulations in Mixed-Integer Convex Programming". In: *Integer Programming and Combinatorial Optimization: 18th International Conference, IPCO 2016, Liège, Belgium, June 1-3, 2016, Proceedings*. Ed. by Quentin Louveaux and Martin Skutella. Cham: Springer International Publishing, pp. 102–113. ISBN: 978-3-319-33461-5. DOI: 10.1007/978-3-319-33461-5_9.

— (Sept. 2017). "Polyhedral approximation in mixed-integer convex optimization". In: *Mathematical Programming*. ISSN: 1436-4646. DOI: 10.1007/s10107-017-1191-y.

Lubin, Miles, Ilias Zadik, and Juan Pablo Vielma (2017). "Regularity in mixed-integer convex representability". In: *arXiv preprint arXiv:1706.05135*.

M.S. Andersen, J. Dahl, L. Vandenberghe (2021). *CVXOPT User's Guide - Cone Programming - Algorithm Parameters*. URL: https://cvxopt.org/userguide/coneprog.html#algorithm-parameters.

Mazumder, Rahul et al. (2019). "A computational framework for multivariate convex regression and its variants". In: *Journal of the American Statistical Association* 114.525, pp. 318–331.

Mehrotra, Sanjay (1992). "On the implementation of a primal-dual interior point method". In: *SIAM Journal on Optimization* 2.4, pp. 575–601.

MOSEK ApS (2020a). *Modeling Cookbook revision 3.2.1*. URL: https://docs.mosek.com/modeling-cookbook/index.html.

— (2020b). *MOSEK Fusion API for Python*. URL: https://docs.mosek.com/9.1/pythonfusion/index.html.

— (2022). *MOSEK Optimizer API for Java 9.3.14*. URL: https://docs.mosek.com/latest/javaapi/index.html.

Murray, Riley, Venkat Chandrasekaran, and Adam Wierman (2020). "Signomial and polynomial optimization via relative entropy and partial dualization". In: *Mathematical Programming Computation*, pp. 1–39.

Myklebust, Tor and Levent Tunçel (2014). *Interior-point algorithms for convex optimization based on primal-dual metrics*. arXiv: 1411.2129 [math.OC].

Nesterov, Yuri (2012). "Towards non-symmetric conic optimization". In: *Optimization Methods and Software* 27.4-5, pp. 893–917.

Nesterov, Yuri et al. (2018). *Lectures on convex optimization*. Vol. 137. Springer.

Nesterov, Yuri and Arkadi Nemirovski (1994). *Interior-point polynomial algorithms in convex programming*. Studies in Applied Mathematics. Society for Industrial and Applied Mathematics.

Nesterov, Yuri and Michael J Todd (1998). "Primal-dual interior-point methods for self-scaled cones". In: *SIAM Journal on Optimization* 8.2, pp. 324–364.

Nesterov, Yuri, Michael J Todd, and Yinyu Ye (1996). *Infeasible-start primal-dual methods and infeasibility detectors for nonlinear programming problems*. Tech. rep. Cornell University Operations Research and Industrial Engineering.

— (1997). "Self-scaled barriers and interior-point methods for convex programming". In: *Mathematics of Operations Research* 22.1, pp. 1–42.

O'Donoghue, Brendan et al. (2016). "Conic optimization via operator splitting and homogeneous self-dual embedding". In: *Journal of Optimization Theory and Applications* 169.3, pp. 1042–1068.

Orban, Dominique (2019). *BenchmarkProfiles.jl*. Version v0.3.3. DOI: 10.5281/zenodo.4630955. URL: https://doi.org/10.5281/zenodo.4630955.

Papp, Dávid and Farid Alizadeh (2013). "Semidefinite characterization of sum-of-squares cones in algebras". In: *SIAM Journal on Optimization* 23.3, pp. 1398–1423.

— (2014). "Shape-constrained estimation using nonnegative splines". In: *Journal of Computational and Graphical Statistics* 23.1, pp. 211–231.

Papp, Dávid and Sercan Yıldız (2017). "On "A Homogeneous Interior-Point Algorithm for Non-Symmetric Convex Conic Optimization"". In: *arXiv preprint arXiv:1712.00492*.

— (2019). "Sum-of-squares optimization without semidefinite programming". In: *SIAM Journal on Optimization* 29.1, pp. 822–851.

— (2020). *alfonso: ALgorithm FOr Non-Symmetric Optimization*. URL: https://github.com/dpapp-github/alfonso.

Papp, Dávid and Sercan Yıldız (2021). "Alfonso: Matlab package for nonsymmetric conic optimization". In: *INFORMS Journal on Computing*. URL: https://doi.org/10.1287/ijoc.2021.1058.

Parrilo, Pablo A (2012). "Chapter 3: Polynomial Optimization, Sums of Squares, and Applications". In: *Semidefinite Optimization and Convex Algebraic Geometry*. SIAM, pp. 47–157. DOI: 10.1137/1.9781611972290.ch3.

Permenter, Frank, Henrik A Friberg, and Erling D Andersen (2017). "Solving conic optimization problems via self-dual embedding and facial reduction: a unified approach". In: *SIAM Journal on Optimization* 27.3, pp. 1257–1282.

Quesada, I. and I.E. Grossmann (1992). "An LP/NLP based branch and bound algorithm for convex MINLP optimization problems". In: *Computers & Chemical Engineering* 16.10, pp. 937–947. ISSN: 0098-1354. DOI: 10.1016/0098-1354(92)80028-8. URL: https://www.sciencedirect.com/science/article/abs/pii/0098135492800288.

Recht, Benjamin, Maryam Fazel, and Pablo A Parrilo (2010). "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization". In: *SIAM Review* 52.3, pp. 471–501.

Renegar, James (2001). *A mathematical view of interior-point methods in convex optimization*. SIAM.

Rockafellar, Ralph Tyrell (2015). *Convex analysis*. Princeton university press. DOI: 10.1515/9781400873173.

Roy, Scott and Lin Xiao (2021). "On self-concordant barriers for generalized power cones". In: *Optimization Letters*, pp. 1–14.

Sendov, Hristo S (2007). "The higher-order derivatives of spectral functions". In: *Linear algebra and its applications* 424.1, pp. 240–281. DOI: 10.1016/j.laa.2006.12.013.

Serrano, Santiago Akle (Mar. 2015). "Algorithms for unsymmetric cone optimization and an implementation for problems with the exponential cone". PhD thesis. Stanford, CA: Stanford University.

Skajaa, Anders, Erling D Andersen, and Yinyu Ye (2013). "Warmstarting the homogeneous and self-dual interior point method for linear and conic quadratic problems". In: *Mathematical Programming Computation*, pp. 1–25.

Skajaa, Anders and Yinyu Ye (2015). "A homogeneous interior-point algorithm for nonsymmetric convex conic optimization". In: *Mathematical Programming* 150.2, pp. 391–422.

Sun, Defeng and Jie Sun (2008). "Löwner's operator and spectral functions in Euclidean Jordan algebras". In: *Mathematics of Operations Research* 33.2, pp. 421–445. DOI: 10.1287/moor.1070.0300.

Sun, Yifan and Lieven Vandenberghe (2015). "Decomposition methods for sparse matrix nearness problems". In: *SIAM Journal on Matrix Analysis and Applications* 36.4, pp. 1691–1717.

Sutter, David et al. (2015). "Efficient approximation of quantum channel capacities". In: *IEEE Transactions on Information Theory* 62.1, pp. 578–598. DOI: 10.1109/tit.2015.2503755. eprint: 1407.8202.

Tawarmalani, Mohit and Nikolaos V Sahinidis (2005). "A polyhedral branch-and-cut approach to global optimization". In: *Mathematical programming* 103.2, pp. 225–249.

Tunçel, Levent et al. (2004). "Geometry of homogeneous convex cones, duality mapping, and optimal self-concordant barriers". In: *Mathematical programming* 100.2, pp. 295–316.

Udell, Madeleine et al. (2014). "Convex optimization in Julia". In: *Proceedings of the 1st First Workshop for High Performance Technical Computing in Dynamic Languages*. IEEE Press, pp. 18–28.

Vandenberghe, Lieven (2010). *The CVXOPT linear and quadratic cone program solvers*. URL: https://www.seas.ucla.edu/~vandenbe/publications/coneprog.pdf.

Vieira, Manuel V.C. (Nov. 2007). "Jordan Algebraic approach to symmetric optimization". PhD thesis. NL 2628 CD, Delft, The Netherlands: Faculty of Electrical Engineering, Mathematics and Computer Science, TU Delft. ISBN: 978-90-6464-189-3.

— (2016). "Derivatives of eigenvalues and Jordan frames". In: *Numerical Algebra, Control & Optimization* 6.2, p. 115. DOI: 10.3934/naco.2016003.

Vielma, Juan Pablo (Mar. 2018). "Small and strong formulations for unions of convex sets from the Cayley embedding". In: *Mathematical Programming*. ISSN: 1436-4646. DOI: 10.1007/s10107-018-1258-4.

Vielma, Juan Pablo, Shabbir Ahmed, and George Nemhauser (2010). "Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions". In: *Operations research* 58.2, pp. 303–315.

Vielma, Juan Pablo, Iain Dunning, et al. (Sept. 2017). "Extended formulations in mixed integer conic quadratic programming". In: *Mathematical Programming Computation* 9.3, pp. 369–418. ISSN: 1867-2957. DOI: 10.1007/s12532-016-0113-y.

Vigerske, Stefan (2018). *MINLPLIB2 Library*. Ed. by GAMS. URL: http://www.minlplib.org.

Witzig, Jakob, Timo Berthold, and Stefan Heinz (2017). "Experiments with conflict analysis in mixed integer programming". In: *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer, pp. 211–220. DOI: 10.1007/978-3-319-59776-8_17.

Xu, Xiaojie, Pi-Fang Hung, and Yinyu Ye (1996). "A simplified homogeneous and self-dual linear programming algorithm and its implementation". In: *Annals of Operations Research* 62.1, pp. 151–171.

Yamashita, Makoto, Katsuki Fujisawa, and Masakazu Kojima (2003). "Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0)". In: *Optimization Methods and Software* 18.4, pp. 491–505.

Yang, Jian et al. (2016). "Nuclear norm based matrix regression with applications to face recognition with occlusion and illumination changes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.1, pp. 156–171.

Zhang, Shuzhong (2004). "A new self-dual embedding method for convex programming". In: *Journal of Global Optimization* 29.4, pp. 479–496. DOI: 10.1023/b:jogo.0000047915.10754.a1.