Techniques for handling nonsymmetric cones in interior point algorithms

by

Lea Kapelevich

B.E. (Hons), University of Auckland (2016)

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

Author .....
Sloan School of Management
April 28, 2022

Certified by .....
Bart P.G. Van Parys
Assistant Professor
Thesis Supervisor

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

Techniques for handling nonsymmetric cones in interior point algorithms

by

Lea Kapelevich

Submitted to the Sloan School of Management
on April 28, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

Conic programs that seek to minimize a linear function over an intersection of *symmetric* (self-dual and homogeneous) cones are amenable to highly e□cient primal-dual interior point methods, which are implemented by many popular o□-the-shelf conic solvers. On the other hand, many useful conic sets cannot be modeled exactly or can be modeled more e□ciently using cones that are not symmetric. Algorithms for *nonsymmetric* cones have been implemented in significantly fewer solvers. Practically efficient, self-concordant barrier functions have not been previously suggested for many useful nonsymmetric cones. For the nonsymmetric cones with known barriers, there is little published work on how to implement numerically stable and computationally fast barrier oracles for interior point methods.

We begin this thesis by describing the interior point algorithm we implement in the solver Hypatia for *exotic* cones. The exotic cones are a broad class of cones (including symmetric and nonsymmetric cones) that admit a small set of oracles needed by Hypatia's algorithm. We justify a number of practical algorithmic enhancements from an empirical standpoint. We derive new logarithmically-homogeneous, self-concordant barrier functions for several useful nonsymmetric cones. In Chapter 3, these are barriers for cones derived from spectral functions on Euclidean Jordan algebras while in Chapter 5, these are barriers related to sum-of-squares polynomial cones. We show that using these cones with our new barriers is computationally favorable in comparison to alternative formulations. We show how to evaluate the oracles needed by Hypatia's algorithm for these barriers and others in a computationally e□cient and numerically stable fashion throughout Chapters 3 to 5.

In the final two chapters, we derive e□cient techniques for calculating information related to convex conjugates of the barriers for seven nonsymmetric cones. This information is not used by Hypatia, but is necessary for the alternative algorithm by Dahl and Andersen [2021] that is implemented by the solver MOSEK. We implement the stepping procedure described by Dahl and Andersen [2021] in Hypatia and make some empirical comparisons between MOSEK's algorithm and Hypatia's.

Thesis Supervisor: Bart P.G. Van Parys
Title: Assistant Professor

Acknowledgments

A big first and foremost thanks to Juan Pablo. Thank you for all the energy and time you dedicate to helping us. Thank you for taking interest in anything we propose, for being patient, and for making our work and meetings fun. I would like to thank my committee (Bart Van Parys and Rahul Mazumder) for their engagement in this work. I appreciate their willingness to delve into nonsymmetric cones and the thoughtful questions they asked. I am grateful to Erling and the team at MOSEK for hosting me last summer. The work at MOSEK was the basis for chapters 6 and 7 of this thesis. Chris, I will always be grateful for the mentorship, especially when I was new to the group. Thank you for being a fun partner for discovery. A special thank you to all the housemates I've lived with over the last five years, ORC friends, and especially Holly, Rebecca, and Michaela for their friendships.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Any convex optimization problem can be expressed in conic form:

$$\inf_x \quad c^\top x : \tag{1.1a}$$

$$b - Ax = 0, \tag{1.1b}$$

$$h - Gx \in K. \tag{1.1c}$$

Here x is a decision variable, b, c, and h are vectors of appropriate size, while A and G are linear operators. K is a conic set, i.e. a set such that for any $w \in K$, we have that $\theta w \in K$ for all nonnegative constants θ.

Under certain (typically nonrestrictive) conditions, a conic problem admits a simple and easily checkable certificate of optimality, primal infeasibility, or dual infeasibility [Permenter et al., 2017]. Practitioners have access to a variety of off-the-shelf solvers, which are able to return such certificates for (1.1). Most of these solvers implement *primal-dual interior point methods* (PDIPMs). Some examples are CSDP [Borchers, 1999], CVXOPT [Andersen et al., 2011], ECOS [Serrano, 2015], MOSEK [MOSEK ApS, 2020], SDPA [Yamashita et al., 2003], SeDuMi [Labit et al., 2002], and SDPT3 [Toh et al., 1999]. The PDIPMs implemented by these solvers generate a series of primal and dual iterates approximately following a *central path*. The central path can be thought of a set of primal-dual points that satisfy a set of nonlinear equations that depend on the problem data of (1.1) and a *barrier function* of K.

The majority of this thesis is dedicated to improving PDIPMs for a broad class of problems in the form of (1.1).

Many popular PDIPM solvers are specialized to support *symmetric* cones. A symmetric cone is self-dual (the dual cone is equal to the primal) and homogeneous (the set of automorphisms of the cone act transitively in the cone). The symmetric cones useful for optimization are the nonnegative orthant, the second order cone, and the positive semidefinite (PSD) matrices. In their seminal work, Nesterov and Todd [1997, 1998] show that these cones (which the authors refer to as the *self-scaled cones*) contain special points (the so-called *Nesterov-Todd scaling points*), which are used to derive good approximations of the barrier far from the current iterate. In addition, the symmetric cones admit simple barrier functions with highly e□cient oracles for various parts of PDIPMs. These properties are usually exploited by the PDIPM solvers described.

Many useful conic sets are nonsymmetric. For example, the set $f(u, v, w) : u \leq v \log(w/v)g$ (when permuted, commonly referred to as the *exponential cone*) is useful for modeling the logarithm function, and cannot be represented exactly using symmetric cones. An alternative example is the cone of *sum-of-squares* polynomials that we describe in Chapter 5. This cone can be represented with symmetric cones, but the representation requires a lifting into a higher dimensional space.

A longstanding approach to optimizing over nonsymmetric cones has been to either approximate the nonsymmetric cone with symmetric cones (e.g. implemented by the modeling tool CVXQUAD [Fawzi et al., 2018]), or, when possible, to build exact higher dimensional representations using symmetric cones. A variety of modeling techniques are described by Ben-Tal and Nemirovski [2001], and modeling tools such as disciplined convex programming packages (see CVX [Grant et al., 2006], CVXPY [Diamond and Boyd, 2016], and Convex.jl [Udell et al., 2014]) and MathOptInterface's bridges [Legat et al., 2020] are designed to facilitate such transformations. On the other hand, o□-the shelf solvers that handle nonsymmetric cones directly within a PDIPM have not existed until recently.

Early algorithmic frameworks for optimizing over nonsymmetric cones have been

suggested by Nesterov et al. [1999] and Nesterov [2012], but have some limiting factors in comparison to their symmetric counterparts. The high-level algorithms in both papers require derivative information from the conjugates of the barrier functions of the cones in the problem, and it is not clear when this can be evaluated e□ciently. Specifically, the only nonsymmetric cones for which a conjugate of the barrier function has been written in closed form are the three-dimensional exponential cone [Serrano, 2015], the three-dimensional power cone [Nesterov, 2012], and the cone of sparse positive semidefinite matrices with chordal sparsity [Andersen et al., 2013]. Additionally, Nesterov et al. [1999] requires solving a linear system in each iteration that is twice the size of the linear systems that arise in symmetric algorithms.

More recent nonsymmetric algorithms bypass some of the aforementioned issues. The algorithm by Skajaa and Ye [2015] does so by requiring that iterates remain close to the central path. A key advantage of this algorithm is that it requires very few oracles for each primal cone in the problem, and doesn't require oracles relating to the conjugate of the primal barrier. This algorithm is the basis of the PDIPM implemented in our interior point solver Hypatia, which we discuss in detail in Chapter 2.

An alternative algorithm, implemented by the state-of-the-art solver MOSEK is based on a technique by Tunçel [2001], Myklebust and Tunçel [2014] which generalizes the concept of scaling matrices from symmetric cones. Like the algorithm by Skajaa and Ye [2015], the linear systems solved in each iteration are equal in size to those arising in symmetric algorithms. Unlike the algorithm by Skajaa and Ye [2015], information about the conjugate gradient is required to compute search directions.

1.1 Choosing natural over extended formulations

From hereon we refer to the cones supported by MOSEK as the *standard* cones (to reflect that they are supported by an accessible, state-of-the-art conic solver). These are the symmetric cones, the three-dimensional exponential cone, and the three-dimensional power cone. On the other hand, we use the term *exotic cone* to refer to any cone that Hypatia could support- requiring only that it admits e□-

ciently computable oracles for Hypatia's algorithm. Exotic cones can be symmetric or nonsymmetric.

In Coey et al. [2021d], we provide numerous computational arguments for supporting exotic cones. 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, or higher dimensional conic constraints. By supporting exotic cones, Hypatia can be used to directly solve simpler, smaller *natural formulations* (NFs). In Coey et al. [2021d, Section 5], a number of NFs are solved using Hypatia and equivalent EFs are solved using MOSEK. Significant computational advantages are shown from solving the NFs with Hypatia compared to solving the EFs with either Hypatia or MOSEK, especially in terms of solve time and memory usage. The NFs are also typically faster and less memory-intensive to generate. These results motivate us to develop enhancements for PDIPMs, such as the PDIPM implemented in Hypatia, for a broad variety of exotic cones. Such enhancements can be in the form of of algorithmic innovations, or improved numerical techniques for evaluating oracles of cones. This thesis studies both.

1.2 The Hypatia solver

Hypatia [Coey et al., 2021d] is an open-source, extensible conic primal-dual interior point solver.[1] Hypatia is written in the Julia language [Bezanson et al., 2017] and is accessible through a flexible, low-level native interface or the modeling tool JuMP [Dunning et al., 2017]. A key feature of Hypatia is a generic cone interface that allows users to define new exotic cones. Adding an exotic cone to Hypatia amounts to implementing the oracles we describe in Section 2.2 for either the cone or its dual. Defining a new cone through Hypatia's cone interface makes both the cone and its dual available for use in conic formulations. We have already predefined 23 useful exotic cone types (some with multiple variants) in Hypatia.

[1] Hypatia is available at github.com/chriscoey/Hypatia.jl under the MIT license.

1.3 Notation

For a natural number d, we define the index set $\mathbb{J}d\mathbb{K} := \{1, 2, \ldots, d\}$. Often we construct vectors with round parentheses, e.g. (a, b, c), and matrices with square brackets, e.g. $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$. We use $s_{i \in \mathbb{J}d\mathbb{K}}$ for the product of elements $(s_1, \ldots, s_d)$. For a set C, cl(C) and int(C) denote the closure and interior of 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 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 $\text{mat}_{d_1,d_2} : \mathbb{R}^{d_1 d_2} \to \mathbb{R}^{d_1 \times d_2}$.

$\mathbb{S}^d$ is the space of symmetric matrices with side dimension d, and $\mathbb{S}^d_+$ and $\mathbb{S}^d_{++}$ denote the positive semidefinite and positive definite symmetric matrices. The inequality $S \succeq Z$ is equivalent to $S - Z \in \mathbb{S}^d_+$ (and similarly for the strict inequality $\succ$ and $\mathbb{S}^d_{++}$). We let $\text{sd}(d) := d(d+1)/2$ be the dimension of the vectorized upper triangle of $\mathbb{S}^d$. We overload the vectorization operator vec : $\mathbb{S}^d \to \mathbb{R}^{\text{sd}(d)}$ 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 $S \in \mathbb{S}^3$ we have $\text{sd}(3) = 6$ and $\text{vec}(S) = (S_{1,1}, \sqrt{2}S_{1,2}, S_{2,2}, \sqrt{2}S_{1,3}, \sqrt{2}S_{2,3}, S_{3,3}) \in \mathbb{R}^{\text{sd}(3)}$. The inverse mapping $\text{mat} : \mathbb{R}^{\text{sd}(d)} \to \mathbb{S}^d$ is well-defined.

For a vector or matrix A, the transpose is $A^\top$ and the trace is $\text{tr}(A)$. We use the standard inner product on $\mathbb{R}^d$, i.e. $s^\top z = \sum_{i \in \mathbb{J}d\mathbb{K}} s_i z_i$ for $s, z \in \mathbb{R}^d$, which equips $\mathbb{R}^d$ with the standard norm $\|s\| = (s^\top s)^{1/2}$. The linear operators vec and mat preserve inner products, e.g. $\text{vec}(S)^\top \text{vec}(Z) = \text{tr}(S^\top Z)$ for $S, Z \in \mathbb{R}^{d_1 \times d_2}$ or $S, Z \in \mathbb{S}^d$. For a linear operator $M : A \to B$, the adjoint $M^* : B \to A$ is the unique operator satisfying $\langle s, Mz \rangle_A = \langle z, M^* s \rangle_B$ for all $s \in A$ and $z \in B$. $\text{Diag} : \mathbb{R}^d \to \mathbb{S}^d$ is the diagonal matrix of a given vector, and $\text{diag} : \mathbb{S}^d \to \mathbb{R}^d$ is the vector of the diagonal of a given matrix. For dimensions implied by context, e is a vector of 1s, e_i is the ith unit vector, and 0 is a vector or matrix of 0s. $I(d)$ is the identity in $\mathbb{R}^{d \times d}$.

$|x|$ is the absolute value of $x \in \mathbb{R}$ and $\log(x)$ is the natural logarithm of $x > 0$.

$\det(X)$ is the determinant of $X \in \mathbb{S}^d$, and $\log\det(X)$ is the log-determinant of $X \succ 0$. For a vector $x \in \mathbb{R}^d$, $\|x\|_1 = \max_{i=1,\dots,d} |x_i|$ is the ℓ_1 norm and $\|x\|_1 = \sum_{i=1,\dots,d} |x_i|$ is the ℓ_1 norm.

The kth derivative of a function f evaluated at a point w is $\nabla^k f(w)$, which may be interpreted as an operator. For example, the second directional derivative of f at w applied in the direction r twice is $\nabla^2 f(w)[r,r] = \langle \nabla^2 f(w)[r], r \rangle$. Often we omit the point at which the derivative is evaluated if this is clear from context. We use subscripts for partial derivatives, for example $\nabla_w f$.

1.4 Cones and barrier functions

Let 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 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. The dual cone of K is K^*, which is also a proper cone in $\mathbb{R}^q$:

$$K^* := \{z \in \mathbb{R}^q : s^\top z \geq 0, \forall s \in K\}. \tag{1.2}$$

Analysis of conic interior point methods relies heavily on the notion of *logarithmically-homogeneous, self-concordant barriers* (LHSCBs). Following Nesterov and Nemirovskii [1994, Sections 2.3.1 and 2.3.3], $f : \text{int}(K) \to \mathbb{R}$ is a ν-LHSCB for K, where $\nu \geq 1$ is the *LHSCB parameter*, if it is three times continuously differentiable, strictly convex, satisfies $f(w_i) \to 1$ along every sequence $w_i \in \text{int}(K)$ converging to the boundary of K, and:

$$\left|\nabla^3 f(w)[r,r,r]\right| \leq 2\left(\nabla^2 f(w)[r,r]\right)^{3/2} \quad \forall w \in \text{int}(K), r \in \mathbb{R}^d, \tag{1.3a}$$

$$f(\theta w) = f(w) - \nu \log(\theta) \quad \forall w \in \text{int}(K), \theta > 0. \tag{1.3b}$$

Complexity analysis of idealized PDIPMs shows that they converge to ε tolerance in $O(\sqrt{\nu}\log(1/\varepsilon))$ iterations, where ν is the barrier parameter of the LHSCB. A key result by Skajaa and Ye [2015] is the affirmation that their proposed non-symmetric

interior point method matches this best-known complexity.

A Cartesian product $K = K_1 \times \cdots \times K_K$ of K proper cones is a proper cone, and its dual cone is $K^* = K_1^* \times \cdots \times K_K^*$. In this case, if f_k is a ν_k-LHSCB for K_k, then $\sum_{k2\mathtt{J}K\mathtt{K}} f_k$ is an LHSCB for K with parameter $\sum_{k2\mathtt{J}K\mathtt{K}} \nu_k$ [Nesterov and Nemirovskii, 1994, Proposition 2.3.3]. We call 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 1). Note K^* is primitive if and only if K is primitive. Primitive proper cones are the fundamental building blocks of conic formulations.

Let $f : \mathrm{dom}(f) \; ! \; \mathsf{R}$ be an arbitrary function with domain $\mathrm{dom}(f)$. We define $f^* : \mathrm{dom}(f^*) \; ! \; \mathsf{R}[\; 1$ of f as the modified Legendre-Fenchel transformation (similar to Zhang [2004, page 483]):

$$f^*(r) = \sup_{w2\mathrm{dom}(f)} f - hw, rii - f(w)g, \tag{1.4}$$

If f is an LHSCB for K, then f^* is an LHSCB for K^* [Nesterov and Todd, 1997, (2.6)] and we refer to it as the *conjugate barrier*.

1.5 Thesis contributions and outline

In Chapter 2 we describe the PDIPM implemented in our interior point solver Hypatia, which allows optimizing over exotic cones. We introduce a number of practical performance enhancements and provide computational evidence that these enhancements improve iteration counts and solve times on a variety of test problems. We also introduce a benchmark set of problems that we reuse in Chapter 7 . The algorithm from this chapter requires a specific set of oracles that are derived for various classes of cones in later sections. In Chapter 3, we focus on the class of nonsymmetric cones that can be described using epigraphs of spectral functions. We show that this is a useful class of cones and provide cases of cones in this class that admit simple LHSCBs with small parameters. We show that the oracles for those barriers are e□cient to compute. Chapter 4 is focused on deriving oracles for cones that can be described as

linear slices of the PSD cone. This class encompasses a number of cones implemented in Hypatia, including several polynomial-like cones that we describe in Chapter 5. In Chapter 5 we describe several polynomial cones that are inspired by the cone of sum-of-squares polynomials. These can be thought of as polynomial generalizations of three conic sets. We provide new LHSCBs for three new cones, and provide some computational justification for using these three specialized cones in place of alternative formulations. In Chapter 6 and Chapter 7, we shift our focus from the algorithm implemented in Hypatia to MOSEK's algorithm. MOSEK's algorithm requires conjugate gradient information. In Chapter 6 we show how to calculate this information e□ciently for seven useful nonsymmetric cones. These conjugate gradients have not been written explicitly before. Chapter 7 provides a computational analysis and discussion of the value of including conjguate gradient information.

Chapter 2

Hypatia's algorithm: practical algorithmic enhancements for a nonsymmetric PDIPM

This chapter is based on the submitted paper Coey et al. [2021c].

2.1 Introduction

2.1.1 The Skajaa-Ye algorithm

As described in Chapter 1, the algorithm by Skajaa and Ye [2015], henceforth referred to as *SY*, is the earliest PDIPM for non-symmetric cones that matches the best known worst-case complexity of symmetric PDIPMs and does not require any conjugate barrier information. The algorithm approximately traces the central path of the homogeneous self-dual embedding (HSDE) [Andersen et al., 2003, Xu et al., 1996], allowing for infeasible starts and detection of primal or dual infeasiblity certificates. 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. 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. 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.

2.1.2 Practical algorithmic developments

For many proper cones of interest, including most of Hypatia's non-symmetric cones, we are aware of LHSCBs with tractable oracles for either the cone or its dual cone but not both. Suppose a 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 Coey et al. [2021d]). In this case, SY can solve neither the primal problem nor its conic dual, as SY requires primal oracles. Hypatia's 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 PDIPMs. This complexity analysis was corrected by Papp and Yıldız [2017], who implemented SY in their recent MATLAB solver Alfonso [Papp and Yıldız, 2020, 2021]. It is well known that performant PDIPM 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 large 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 PDIPMs 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], Dahl and 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 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 Andersen [2021] test their technique only for the standard exponential cone, we implement and test our TOO for all of Hypatia's 23 predefined cones. In our experience, requiring a tractable TOO is only as restrictive

[1] To avoid confusion, we do not use the term 'corrector'. In the terminology of Mehrotra [1992], Dahl and 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], Papp and Yıldız [2017] as the 'corrector' direction.

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. To illustrate, we describe efficient and numerically stable TOO procedures for several cones that can be characterized as intersections of slices of the PSD cone.

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. We only describe and test one set of (default) options for these components.

2.1.3 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 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.

2.1.4 Chapter overview

In Section 2.2, we define exotic cones, LHSCBs, and our required cone oracles (including the TOO). In Section 2.3, we describe Hypatia's general primal-dual conic form, associated conic certificates, and the HSDE. In Section 2.4, 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 Section 2.5, we briefly introduce Hypatia's predefined exotic cones and show that our TOO is relatively cheap to compute. In Section 2.6, we discuss advanced procedures for preprocessing and initial point finding, and solving structured linear systems for directions. In Section 2.7, we summarize our applied examples and exotic conic benchmark instances, and finally we present our computational results demonstrating the practical e□cacy of our stepping enhancements.

2.2 Exotic cones and oracles

Suppose we have tractable oracles for $K \subset \mathbb{R}^q$ and let $f : \text{int}(K) \to \mathbb{R}$ denote the ν-LHSCB for K. The oracles for 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 $s \in \text{int}(K)$.

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

$$g(s) := \nabla f(s) \in \mathbb{R}^q, \tag{2.1a}$$

$$H(s) := \nabla^2 f(s) \in \mathbb{S}^q. \tag{2.1b}$$

Third order directional derivative. Given a point $s \in \text{int}(K)$ 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:

$$\mathrm{T}(s, \delta_s) := -\tfrac{1}{2} r^3 f(s)[\delta_s, \delta_s] \in \mathbb{R}^q. \tag{2.2}$$

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

In Section 2.5, we introduce Hypatia's predefined cones and discuss the time complexity of computing the feasibility check, gradient, Hessian, and TOO oracles. 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 K. This is the unique point satisfying $t \in \text{int}(K) \setminus \text{int}(K^*)$ and $t = -g(t)$ [Dahl and Andersen, 2021]. It can also be characterized as the solution to the following strictly convex problem:

$$\arg\min_{s \in \text{int}(K)} \left(f(s) + \tfrac{1}{2}\|s\|^2 \right). \tag{2.3}$$

For the nonnegative cone $K = \mathbb{R}^n_+$, $f(s) = -\log(s)$ 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.

2.3 General conic form and certificates

In Sections 2.3.1 and 2.3.2, we describe our general conic primal-dual form and the associated conic certificates. In Section 2.3.3, we introduce the homogeneous self-dual

embedding (HSDE) conic feasibility problem, a solution of which may provide a conic certificate.

2.3.1 General conic form

Recall (1.1), which is Hypatia's primal conic form:

$$\inf_x \quad c^\top x : \tag{2.4a}$$

$$b - Ax = 0, \tag{2.4b}$$

$$h - Gx \in \mathcal{K}, \tag{2.4c}$$

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 $k \in \mathtt{J}K\mathtt{K}$, we let $q_k = \dim(\mathcal{K}_k)$, so $\sum_{k \in \mathtt{J}K\mathtt{K}} q_k = q = \dim(\mathcal{K})$. 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 (2.4c). The primal form (2.4) matches CVXOPT's form, however CVXOPT only recognizes symmetric cones [Vandenberghe, 2010]. Unlike the conic form used by Skajaa and Ye [2015], 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 (2.4), over variables $y \in \mathbb{R}^p$ and $z \in \mathbb{R}^q$ associated with (2.4b) and (2.4c), is:

$$\sup_{y,z} \quad -b^\top y - h^\top z : \tag{2.5a}$$

$$c + A^\top y + G^\top z = 0, \tag{2.5b}$$

$$z \in \mathcal{K}^*, \tag{2.5c}$$

where (2.5b) is associated with the primal variable $x \in \mathbb{R}^n$.

2.3.2 Conic certificates

Under certain conditions, there exists a simple conic certificate providing an easily verifiable proof of infeasibility of the primal (2.4) or dual (2.5) 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:

$$c^\top x < 0, \tag{2.6a}$$

$$-Ax = 0, \tag{2.6b}$$

$$-Gx \in \mathcal{K}, \tag{2.6c}$$

and hence it certifies dual infeasibility.

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

$$-b^\top y - h^\top z > 0, \tag{2.7a}$$

$$A^\top y + G^\top z = 0, \tag{2.7b}$$

$$z \in \mathcal{K}^*, \tag{2.7c}$$

and hence it certifies primal infeasibility.

A complementary solution (x, y, z) satisfies the primal-dual feasibility conditions (2.4b), (2.4c), (2.5b) and (2.5c), and has equal and attained primal and dual objective values:

$$c^\top x = -b^\top y - h^\top z, \tag{2.8}$$

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*. 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 MOSEK ApS [2020, Section 7.2] and Permenter et al. [2017] for more details).

2.3.3 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 (2.4b), (2.4c), (2.5b), (2.5c) and (2.8):

$$\begin{bmatrix} 0 \\ 0 \\ s \\ \kappa \end{bmatrix} = \begin{bmatrix} 0 & A^\top & G^\top & c \\ -A & 0 & 0 & b \\ -G & 0 & 0 & h \\ -c^\top & -b^\top & -h^\top & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \tau \end{bmatrix}, \tag{2.9a}$$

$$(z, \tau, s, \kappa) \in \left(\mathcal{K}^* \times \mathbb{R}_+ \times \mathcal{K} \times \mathbb{R}_+\right). \tag{2.9b}$$

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 $E \in \mathbb{R}^{(n+p+q+1)\times \dim(\mathcal{K})}$ such that (2.9a) is equivalent to:

$$E\omega = 0. \tag{2.10}$$

Here we assume E has full row rank; in Section 2.6 we discuss preprocessing techniques that handle linearly dependent rows. Note that $\omega = 0$ satisfies (2.9), so the HSDE is always feasible. A point ω is an interior point if it is strictly feasible for the conic constraints (2.9b), i.e. ω satisfies $(z, \tau, s, \kappa) \in \operatorname{int}\left(\mathcal{K}^* \times \mathbb{R}_+ \times \mathcal{K} \times \mathbb{R}_+\right)$.

Suppose a point ω is feasible for the HSDE (2.9). From skew symmetry of the square 4×4 block matrix in (2.9a), we have $s^\top z + \kappa\tau = 0$. From the conic constraints (2.9b) and the dual cone inequality (1.2) we have $s^\top z \geq 0$ and $\kappa\tau \geq 0$. Hence $s^\top z = \kappa\tau = 0$. 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 (2.4b), (2.4c), (2.5b), (2.5c) and (2.8).

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

Of primal. If $b^\top y + h^\top z < 0$, then (y, z) is a primal infeasibility certificate satisfying (2.7).

Of dual. If $c^\top x < 0$, then x is a dual infeasibility certificate satisfying (2.6).

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]).

If the primal and dual problems are both feasible and have zero duality gap, SY 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) [Skajaa and Ye, 2015, Section 2]. 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 et al., 2017]. The algorithm we describe in Section 2.4 is an extension of SY that inherits these properties.

2.4 Central path following algorithm

In Section 2.4.1, we describe the central path of the HSDE, and in Section 2.4.2 we define central path proximity measures. In Section 2.4.3, we outline a high level PDIPM that maintains iterates close to the central path, and we give numerical convergence criteria for detecting approximate conic certificates. In Section 2.4.4, we derive prediction and centering directions and our corresponding TOA directions using the TOO. Finally in Section 2.4.5, we summarize an SY-like stepping procedure and describe our sequence of four enhancements to this procedure.

2.4.1 Central path of the HSDE

We define the HSDE in (2.9). Recall that K in our primal conic form (2.4) is a Cartesian product $K = K_1 \times \cdots \times K_K$ of K exotic cones. We partition the exotic cone indices $\llbracket K \rrbracket$ into two sets: K_{pr} for cones with primal oracles (i.e. for K_k) and K_{du} for cones with dual oracles (i.e. for K_k^*). For convenience, we append the τ and κ variables onto the s and z variables. Letting $K = K + 1$, we define for $k \in \llbracket K \rrbracket$:

$$K_k := \begin{cases} K_k & k \in K_{\text{pr}}, \\ K_k^* & k \in K_{\text{du}}, \\ \mathbb{R}_+ & k = K, \end{cases} \tag{2.11a}$$

$$(z_k, s_k) := \begin{cases} (z_k, s_k) & k \in K_{\text{pr}}, \\ (s_k, z_k) & k \in K_{\text{du}}, \\ (\kappa, \tau) & k = K. \end{cases} \tag{2.11b}$$

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 $\omega = (x, y, z, \tau, s, \kappa)$, parameterized by $\mu > 0$, satisfying:

$$E\omega = \mu E\omega^0, \tag{2.12a}$$

$$z_k + \mu g_k(s_k) = 0 \quad \forall k \in \llbracket K \rrbracket, \tag{2.12b}$$

$$(z, s) \in \text{int}(K^* \times K). \tag{2.12c}$$

When all exotic cones have primal oracles (i.e. K_{du} is empty), our definition (2.12) 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 (2.12b) allows cones with dual oracles (K_{du} may be nonempty).

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

$$(z_k^0, s_k^0) = (-g_k(t_k), t_k) \quad \forall k \in \mathbb{J}K\mathbb{K}, \tag{2.13}$$

where $t_k \in \text{int}(K_k)$ 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:

$$\min_{x \in \mathbb{R}^n} \quad \|x\|^2 / 2 \tag{2.14a}$$

$$-Ax + b\tau^0 = 0, \tag{2.14b}$$

$$-Gx + h\tau^0 - s^0 = 0, \tag{2.14c}$$

and we let y^0 be the solution of:

$$\min_{y \in \mathbb{R}^p} \quad \|y\|^2 / 2 \tag{2.15a}$$

$$A^\top y + G^\top z^0 + c\tau^0 = 0. \tag{2.15b}$$

In Section 2.6, 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:

$$\mu(\omega) := s^\top z / \textstyle\sum_{k \in \mathbb{J}\bar{K}\mathbb{K}} \nu_k, \tag{2.16}$$

where ν_k is the LHSCB parameter of the LHSCB f_k for K_k (see (1.3b)). Note that $\mu(\omega) > 0$ if $(z, s) \in \text{int}(K^*) \times \text{int}(K)$, by a strict version of the dual cone inequality (1.2). From (2.13), $\mu(\omega^0) = 1$, since in (2.16) we have $(s^0)^\top z^0 = \sum_{k \in \mathbb{J}\bar{K}\mathbb{K}} t_k^\top(-g_k(t_k))$, and $t_k^\top(-g_k(t_k)) = \nu_k$ by logarithmic homogeneity of f_k [Nesterov and Nemirovskii, 1994, Proposition 2.3.4]. Hence ω^0 satisfies the central path conditions (2.12) 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.

2.4.2 Central path proximity

Given a point ω, we define the central path *proximity* π_k for exotic cone $k \in \mathtt{JK}$ as:

$$\pi_k(\omega) := \begin{cases} \left\|(H_k(s_k))^{-1/2}(z_k/\mu(\omega) + g_k(s_k))\right\| & \text{if } \mu(\omega) > 0, s_k \in \operatorname{int}(K_k), \\ 1 & \text{otherwise.} \end{cases} \tag{2.17}$$

Hence π_k is a measure of the distance from s_k and z_k to the surface defined by the central path condition (2.12b) (compare to Skajaa and Ye [2015, Equation 9] and Nesterov and Todd [1998, Section 4]).

In Lemma 2.4.1, we show that for exotic cone $k \in \mathtt{JK}$, if $\pi_k(\omega) < 1$, then $s_k \in \operatorname{int}(K_k)$ and $z_k \in \operatorname{int}(K_k^*)$. This condition is sufficient but not necessary for strict cone feasibility. If it holds for all $k \in \mathtt{JK}$, then ω is an interior point (by definition) and (2.12c) is satisfied. From (2.17), $\pi_k(\omega)$ can be computed by evaluating the feasibility check, gradient, and Hessian oracles for K_k at s_k.

Lemma 2.4.1. *Given a point ω, for each $k \in \mathtt{JK}$, $\pi_k(\omega) < 1$ implies $s_k \in \operatorname{int}(K_k)$ and $z_k \in \operatorname{int}(K_k^*)$.*

Proof. We adapt Papp and Yıldız [2017, Lemma 15]. Fix $\mu = \mu(\omega)$ for convenience, and suppose $\pi_k(\omega) < 1$ for exotic cone $k \in \mathtt{JK}$. Then by (2.17), $\mu > 0$ and $s_k \in \operatorname{int}(K_k)$. By Papp and Yıldız [2017, Theorem 8], $s_k \in \operatorname{int}(K_k)$ implies $-g_k(s_k) \in \operatorname{int}(K_k^*)$. Let f_k be the LHSCB for K_k, and let $H_k^* := \nabla^2 f_k^*$ denote the Hessian operator for the conjugate f_k^* (see (1.4)) of f_k. By Papp and Yıldız [2017, Equation 13], $H_k^*(-g_k(s_k)) = (H_k(s_k))^{-1}$, so:

$$\left\|(H_k^*(-g_k(s_k)))^{1/2}(z_k/\mu + g_k(s_k))\right\| \tag{2.18a}$$

$$= \left\|(H_k(s_k))^{-1/2}(z_k/\mu + g_k(s_k))\right\| \tag{2.18b}$$

$$= \pi_k(\omega) < 1. \tag{2.18c}$$

So by Papp and Yıldız [2017, Definition 1], $z_k/\mu \in \operatorname{int}(K_k^*)$, hence $z_k \in \operatorname{int}(K_k^*)$. □

We now define a proximity function that aggregates the exotic cone central path

proximity values $\pi_k(\omega) \geq 0, \forall k \in [K]$. SY aggregates by taking the ℓ_2 norm:

$$\pi_{SY_2}(\omega) := \left\| (\pi_k(\omega))_{k \in [\bar{K}]} \right\|. \tag{2.19}$$

An alternative aggregated proximity uses the ℓ_1 norm (maximum):

$$\pi_{SY_1}(\omega) := \left\| (\pi_k(\omega))_{k \in [\bar{K}]} \right\|_1. \tag{2.20}$$

Clearly, $0 \leq \pi_k(\omega) \leq \pi_{SY_1}(\omega) \leq \pi_{SY_2}(\omega), \forall k \in [K]$. Both conditions $\pi_{SY_2}(\omega) < 1$ and $\pi_{SY_1}(\omega) < 1$ guarantee by Lemma 2.4.1 that ω is an interior point, however using π_{SY_2} leads to a more restrictive condition on ω.

2.4.3 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 (2.12) through a series of iterations. It maintains feasibility for the linear equality conditions (2.12a) and strict cone feasibility conditions (2.12c), but allows violation of the nonlinear equality conditions (2.12b). On the ith iteration, the current interior point is ω^{i-1} satisfying $\pi_k(\omega^{i-1}) < 1, \forall k \in [K]$, and the complementarity gap is $\mu(\omega^{i-1})$. The method searches for a new point ω^i that maintains the proximity condition $\pi_k(\omega^i) < 1, \forall k \in [K]$ (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_{SY_2} or π_{SY_1}), or both. As the complementarity gap decreases towards zero, the RHS of (2.12a) approaches the origin, so the iterates approach a solution of the HSDE (2.9).

To detect an approximate conic certificate and terminate the iterations, we check if 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 (see Section 2.7.2 for the tolerance values we use in computational testing).

Optimality. We terminate with a complementary solution $(x, y, z)/\tau$ approximately satisfying the primal-dual optimality conditions (2.4b), (2.4c), (2.5b), (2.5c) and (2.8) if:

$$\max\left(\frac{\|A^\top y + G^\top z + c\tau\|_1}{1 + \|c\|_1}, \frac{\|-Ax + b\tau\|_1}{1 + \|b\|_1}, \frac{\|-Gx + h\tau - s\|_1}{1 + \|h\|_1}\right) \leq \varepsilon_f \tau, \tag{2.21a}$$

and at least one of the following two conditions holds:

$$s^\top z \leq \varepsilon_a, \tag{2.21b}$$

$$\min(s^\top z/\tau, |c^\top x + b^\top y + h^\top z|) \leq \varepsilon_r \max(\tau, \min(|c^\top x|, |b^\top y + h^\top z|)). \tag{2.21c}$$

Note that (2.21b) and (2.21c) are absolute and relative optimality gap conditions respectively.

Primal infeasibility. We terminate with a dual improving ray (y, z) approximately satisfying (2.7) if:

$$b^\top y + h^\top z < 0, \qquad \|A^\top y + G^\top z\|_1 \leq -\varepsilon_i (b^\top y + h^\top z). \tag{2.22}$$

Dual infeasibility. We terminate with a primal improving ray x approximately satisfying (2.6) if:

$$c^\top x < 0, \qquad \max(\|Ax\|_1, \|Gx + s\|_1) \leq -\varepsilon_i c^\top x. \tag{2.23}$$

Ill-posed primal or dual. If τ and κ are approximately 0, the primal and dual problem statuses cannot be determined (see Section 2.3.3). We terminate with an ill-posed status if:

$$\mu(\omega) \leq \varepsilon_p, \qquad \tau \leq \varepsilon_p \min(1, \kappa). \tag{2.24}$$

The high level path following algorithm below computes an approximate solution

to the HSDE. In Section 2.4.5, 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 (2.21) to (2.24) do
5: ω^i Step(ω^{i-1})
6: i $i + 1$
7: end while
8: return ω^i
9: end procedure

2.4.4 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 2.4.5 first compute one or more search directions, which depend on ω. We derive the *centering* direction in Section 2.4.4 and the *prediction* direction in Section 2.4.4. 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 6 × 6 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 $r := (r_E, r_1, \ldots, r_{\bar{K}}) \in \mathbb{R}^{\dim(\mathcal{K})}$

represent an RHS, where $r_E \in \mathbb{R}^{n+p+q+1}$ corresponds to the linear equalities (2.12a) and $r_k \in \mathbb{R}^{q_k}, \forall k \in \mathcal{K}$ corresponds to the nonlinear equalities (2.12b). The direction $\delta := (\delta_x, \delta_y, \delta_z, \delta_\tau, \delta_s, \delta_\kappa) \in \mathbb{R}^{\dim(\mathcal{K})}$ corresponding to r is the solution to:

$$E\delta = r_E, \tag{2.25a}$$

$$\delta_{\bar{z}_k} + \mu H_k(s_k)\delta_{\bar{s}_k} = r_k \quad \forall k \in \mathcal{K}. \tag{2.25b}$$

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 2.6, we describe a particular method for solving (2.25).

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 (2.12b) (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 (2.12a), we ensure $E\delta^c = E\delta^{ct} = 0$ in (2.25a).

Dropping the index $k \in \mathcal{K}$ for conciseness, recall that (2.12b) expresses $z + \mu g(s) = 0$. A first order approximation of this condition gives:

$$z + \delta_{\bar{z}} + \mu(g(s) + H(s)\delta_{\bar{s}}) = 0 \tag{2.26a}$$

$$\iff \quad \delta_{\bar{z}} + \mu H(s)\delta_{\bar{s}} = -z - \mu g(s), \tag{2.26b}$$

which matches the form of (2.25b). Hence we let the centering direction δ^c be the solution to:

$$E\delta = 0, \tag{2.27a}$$

$$\delta_{\bar{z}_k} + \mu H_k(s_k)\delta_{\bar{s}_k} = -z_k - \mu g_k(s_k) \quad \forall k \in \mathcal{K}. \tag{2.27b}$$

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

$$z + \delta_{\bar{z}} + \mu\big(g(s) + H(s)\delta_{\bar{s}} + \tfrac{1}{2} r^3 f(s)[\delta_{\bar{s}}, \delta_{\bar{s}}]\big) = 0 \tag{2.28a}$$

$$\Rightarrow \quad \delta_{\bar{z}} + \mu H(s)\delta_{\bar{s}} = -z - \mu g(s) + \mu \mathsf{T}(s, \delta_{\bar{s}}), \tag{2.28b}$$

where (2.28b) uses the definition of the TOO in (2.2). Note that the RHSs of (2.26b) and (2.28b) differ only by $\mu \mathsf{T}(s, \delta_{\bar{s}})$, 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 (2.28b). Hence we let the centering TOA direction δ^{ct}, which adjusts the centering direction, be the solution to:

$$E\delta = 0, \tag{2.29a}$$

$$\delta_{\bar{z};k} + \mu H_k(s_k)\delta_{\bar{s};k} = \mu \mathsf{T}_k\big(s_k, \delta^c_{\bar{s};k}\big) \quad \forall k \in \mathcal{K}. \tag{2.29b}$$

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).

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 (2.12) as a dynamical system parametrized by $\mu > 0$, and differentiating the linear and nonlinear equalities (2.12a) and (2.12b).

Differentiating (2.12a) once gives:

$$E\dot{\omega} = E\omega^0. \tag{2.30}$$

Rescaling (2.30) by $-\mu$ and substituting (2.12a) gives:

$$E(-\mu\dot{\omega}) = -\mu E\omega^0 = -E\omega . \tag{2.31}$$

Dropping the index $k \in [K]$ for conciseness, we differentiate $z = \mu g(s)$ from (2.12b) once to get:

$$\dot{z} = g(s) + \mu H(s)\dot{s} = 0. \tag{2.32}$$

Rescaling (2.32) by $-\mu$ and substituting $z = -\mu g(s)$ from (2.12b) gives:

$$-\mu\dot{z} + \mu H(s)(-\mu\dot{s}) = -z. \tag{2.33}$$

The direction ω is tangent to the central path. Like SY, we interpret the prediction direction as $\delta^p = -\mu\omega$, so (2.31) and (2.33) become:

$$E\delta^p = -E\omega, \tag{2.34a}$$

$$\delta^p_{\bar{z}} + \mu H(s)\delta^p_{\bar{s}} = -z, \tag{2.34b}$$

which matches the form (2.25). So we let δ^p be the solution to:

$$E\delta = -E\omega, \tag{2.35a}$$

$$\delta_{\bar{z};k} + \mu H_k(s_k)\delta_{\bar{s};k} = -z_k \quad \forall k \in [K]. \tag{2.35b}$$

Differentiating (2.12a) twice and rescaling by $\frac{1}{2}\mu^2$ gives:

$$E\left(\tfrac{1}{2}\mu^2\ddot{w}\right) = 0. \tag{2.36}$$

Differentiating $\dot{z} = \mu g(s)$ twice gives:

$$\ddot{z} + 2H(s)\dot{s} + \mu r^3 f(s)[\dot{s}, \dot{s}] + \mu H(s)\ddot{s} = 0. \tag{2.37}$$

Rescaling (2.37) by $\frac{1}{2}\mu^2$ and substituting the TOO definition (2.2), we have:

$$\tfrac{1}{2}\mu^2\ddot{z} + \mu H(s)\left(\tfrac{1}{2}\mu^2\ddot{s}\right) = \mu H(s)(-\mu\dot{s}) - \tfrac{1}{2}\mu r^3 f(s)[-\mu\dot{s}, -\mu\dot{s}] \tag{2.38a}$$

$$= \mu H(s)(-\mu\dot{s}) + \mu T(s, -\mu\dot{s}). \tag{2.38b}$$

We interpret the prediction TOA direction, which adjusts the prediction direction, as $\delta^{pt} = \frac{1}{2}\mu^2\omega$. The RHS of (2.38b) depends on s, so we remove this dependency by substituting the prediction direction $\delta^p = -\mu\omega$, which we assume is already computed. Hence using (2.36) and (2.38b), we let δ^{pt} be the solution to:

$$E\delta = 0, \tag{2.39a}$$

$$\delta_{z;k} + \mu H_k(s_k)\delta_{s;k} = \mu H_k(s_k)\delta^p_{s;k} + \mu \mathsf{T}_k\left(s_k, \delta^p_{s;k}\right) \quad 8k \in \mathtt{JK}. \tag{2.39b}$$

We note that the RHS in (2.39b) differs from the 'higher order corrector' RHS of Dahl and Andersen [2021, (16)], which has the form $\frac{1}{2} r^3 f_k\left[\delta^p_{s;k}, (H_k(s_k))^{-1}\delta^p_{z;k}\right]$.

2.4.5 Stepping procedures

A stepping procedure computes one or more directions from Section 2.4.4 and uses the directions to search for a new interior point. Recall from Line 5 of the high level PDIPM in Section 2.4.3 that on iteration i with current iterate ω^{i-1}, Step 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 2.4.5, we describe a stepping procedure similar to that of Alfonso [Papp and Yıldız, 2021], which is a practical implementation of SY. This procedure, which we call *basic*, alternates between prediction and centering steps and does not use the TOA directions. In Sections 2.4.5 to 2.4.5, we describe a sequence of four cumulative enhancements to the *basic* procedure, with the goal of improving iteration counts and per-iteration computational efficiency in practice. The main purpose of our computational testing in Section 2.7 is to assess the value of these enhancements on a diverse set of benchmark instances.

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 (2.19) does not exceed $\eta = 0.0332$ (from Alfonso [Papp and Yıldız, 2020]), or if the most recent $N = 4$ steps have all been centering steps, then we

compute the prediction direction δ^p from (2.35). Otherwise, we compute the centering direction δ^c from (2.27). Letting j be the number of consecutive centering steps taken immediately before the current ith iteration, the search direction is:

$$\delta := \begin{cases} \delta^p & \text{if } \pi_{\backprime_2}(\omega^{i-1}) \le \eta \text{ or } j \ge N, \\ \delta^c & \text{otherwise.} \end{cases} \tag{2.40}$$

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 $A = \{\alpha_l\}_{l=1,\dots,L}$, 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 $\pi_{\backprime_2}(\omega^i) \le \beta_1$ for $\beta_1 = 0.2844$ (from Alfonso [Papp and Yıldız, 2020]), which guarantees interiority by Lemma 2.4.1. If the backtracking search terminates without a step length satisfying the proximity condition (i.e. α_L is too large), the PDIPM algorithm terminates without a solution.

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.

1: procedure BasicStep(ω^{i-1}, j)
2: if $\pi_{\backprime_2}(\omega^{i-1}) \le \eta$ or $j \ge N$ then ▷ choose predict or center
3: $\delta \leftarrow \delta^p$ from (2.35) ▷ compute prediction direction
4: $j \leftarrow 0$
5: else
6: $\delta \leftarrow \delta^c$ from (2.27) ▷ compute centering direction
7: $j \leftarrow j + 1$
8: end if
9: $\hat{\alpha} \leftarrow \max\{\alpha \in A : \pi_{\backprime_2}(\omega^{i-1} + \alpha\delta) \le \beta_1\}$ ▷ compute step length by backtracking search
10: $\omega^i \leftarrow \omega^{i-1} + \hat{\alpha}\delta$ ▷ update current iterate
11: end procedure

Less restrictive proximity

The *basic* stepping procedure in Section 2.4.5 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_{c1} proximity measure from (2.20) instead of π_{c2} to compute the proximity of ω^{i-1}, though we do not change the proximity bound η. To allow longer step lengths, we also use π_{c1} instead of π_{c2} for the backtracking search proximity checks, and we increase this proximity bound to $\beta_2 = 0.99$ (by Lemma 2.4.1, $\beta_2 < 1$ guarantees interiority).

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

1: procedure ProxStep(ω^{i-1}, j)
2: if $\pi_{c1}(\omega^{i-1}) \leq \eta$ or $j \geq N$ then ▷ use less restrictive proximity measure π_{c1}
3: $\delta \leftarrow \delta^p$ from (2.35)
4: $j \leftarrow 0$
5: else
6: $\delta \leftarrow \delta^c$ from (2.27)
7: $j \leftarrow j + 1$
8: end if
9: $\hat{\alpha} \leftarrow \max\{\alpha \in A : \pi_{c1}(\omega^{i-1} + \alpha\delta) \leq \beta_2\}$ ▷ use π_{c1} and larger proximity bound β_2
10: $\omega^i \leftarrow \omega^{i-1} + \hat{\alpha}\delta$
11: end procedure

Third order adjustments

We modify the *prox* stepping procedure in Section 2.4.5 to incorporate the new TOA directions associated with the prediction and centering directions. 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, just like *prox*, and we use this step length $\hat{\alpha}^u \in (0,1)$ to scale down the TOA direction. We let the final direction be $\delta^u + \hat{\alpha}^u\delta^t$. The rescaling of δ^t helps to prevent over-adjustment. Finally, we perform a second backtracking line search, using the same techniques and proximity condition as the first line search.

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

procedure TOAStep(ω^{i-1}, j)
  if $\pi_1(\omega^{i-1}) \le \eta$ or $j \ge N$ then
    $\delta^u \leftarrow \delta^p$ from (2.35)
    $\delta^t \leftarrow \delta^{pt}$ from (2.39) ▷ compute prediction TOA direction
    $j \leftarrow 0$
  else
    $\delta^u \leftarrow \delta^c$ from (2.27)
    $\delta^t \leftarrow \delta^{ct}$ from (2.29) ▷ compute centering TOA direction
    $j \leftarrow j + 1$
  end if
  $\hat{\alpha}^u \leftarrow \max\{\alpha \in A : \pi_1(\omega^{i-1} + \alpha\delta^u) \le \beta_2\}$ ▷ perform line search for unadjusted direction
  $\delta \leftarrow \delta^u + \hat{\alpha}^u\delta^t$ ▷ compute final direction
  $\hat{\alpha} \leftarrow \max\{\alpha \in A : \pi_1(\omega^{i-1} + \alpha\delta) \le \beta_2\}$
  $\omega^i \leftarrow \omega^{i-1} + \hat{\alpha}\delta$
end procedure

Curve search

The *TOA* stepping procedure in Section 2.4.5 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 12 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 $\pi_1(\omega^{i-1}) \leq \eta$ or $j \geq N$ then
3: $\delta^u \leftarrow \delta^p$ from (2.35)
4: $\delta^t \leftarrow \delta^{pt}$ from (2.39)
5: $j \leftarrow 0$
6: else
7: $\delta^u \leftarrow \delta^c$ from (2.27)
8: $\delta^t \leftarrow \delta^{ct}$ from (2.29)
9: $j \leftarrow j + 1$
10: end if
11: let $\hat{\omega}(\alpha) := \omega^{i-1} + \alpha(\delta^u + \alpha\delta^t)$ ▷ use curved trajectory
12: $\hat{\alpha} \leftarrow \max\{\alpha \in A : \pi_1(\hat{\omega}(\alpha)) \leq \beta_2\}$
13: $\omega^i \leftarrow \hat{\omega}(\hat{\alpha})$
14: end procedure

Combined directions

Unlike Skajaa and Ye [2015], Papp and Yıldız [2021], most conic PDIPMs combine the prediction and centering phases (e.g. Vandenberghe [2010], Dahl and 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 2.4.5. 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 11 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 11 to 13 of the *curve* procedure.

1: procedure CombStep(ω^{i-1})
2: compute $\delta^c, \delta^{ct}, \delta^p, \delta^{pt}$ from (2.27), (2.29), (2.35) and (2.39) ▷ use four directions instead of two
3: let $\hat{\omega}(\alpha) := \omega^{i-1} + \alpha(\delta^p + \alpha\delta^{pt}) + (1-\alpha)(\delta^c + (1-\alpha)\delta^{ct})$ ▷ use combined trajectory
4: $\hat{\alpha} \leftarrow \max\{\alpha \in A : \pi_{\mathcal{K}_1}(\hat{\omega}(\alpha)) \leq \beta_2\}$
5: $\omega^i \leftarrow \hat{\omega}(\hat{\alpha})$
6: end procedure

2.5 Oracles for predefined exotic cones

Below we list 23 exotic cone types that we have predefined through Hypatia's generic cone interface (see Section 2.2). Each of these cones is represented in the benchmark set of conic instances that we introduce in Section 2.7.1. Recall that we write any

exotic cone K in vectorized form, i.e. as a subset of $\mathbb{R}^q$, where $q = \dim(K) \geq 1$ is the cone dimension. For cones typically defined using symmetric matrices, we use the standard svec vectorization (see Section 1.3) to ensure the vectorized cone is proper, to preserve inner products, and to simplify the dual cone definition. 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 several cones, we have implemented additional variants over complex numbers (for example, a Hermitian PSD cone), but we omit these definitions here for simplicity. We defer a more complete description of Hypatia's exotic cones and LHSCBs to Coey et al. [2021b,a], Kapelevich et al. [2021].

Nonnegative cone. $K_{\geq} := \mathbb{R}^d_{\geq}$ is the (self-dual) nonnegative real vectors (note for $d > 1$, $K_{\geq}$ is not a primitive cone).

PSD cone. $K_{} := \{w \in \mathbb{R}^{\text{sd}(d)} : \text{mat}(w) \in \mathbb{S}^d\}$ is the (self-dual) PSD matrices of side dimension d.

Doubly nonnegative cone. $K_{\text{DNN}} := K_{\geq} \cap K_{}$ is the PSD matrices with all nonnegative entries of side dimension d.

Sparse PSD cone. K_{sPSD} is the PSD matrices of side dimension s with a fixed sparsity pattern S containing $d \geq s$ nonzeros (including all diagonal elements); see Section 4.4. The dual cone K^*_{sPSD} is the symmetric matrices with pattern S for which there exists a PSD completion, i.e. an assignment of the elements not in S such that the full matrix is PSD. For simplicity, the complexity estimates in Table 2.1 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 [Andersen et al., 2013].

Linear matrix inequality cone. $K_{\text{LMI}} := \{w \in \mathbb{R}^d : \sum_{i \in [d]} w_i P_i \in \mathbb{S}^s\}$ 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.

Infinity norm cone. $K_{\ell_1} := \{(u,w) \in \mathbb{R}_+ \times \mathbb{R}^d : u \geq \|w\|_1\}$ is the epigraph of the ℓ_1 norm on $\mathbb{R}^d$. The dual cone $K^*_{\ell_1}$ is the epigraph of the ℓ_1 norm.

Euclidean norm cone. $K_{\ell_2} := \{(u,w) \in \mathbb{R}_+ \times \mathbb{R}^d : u \geq \|w\|\}$ is the (self-dual) epigraph of the ℓ_2 norm on $\mathbb{R}^d$ (AKA second-order cone).

Euclidean norm square cone. $K_{\text{sqr}} := \{(u,v,w) \in \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}^d : 2uv \geq \|w\|^2\}$ 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).

Spectral norm cone. $K_{\ell_{\text{spec}}} := \{(u,w) \in \mathbb{R}_+ \times \mathbb{R}^{rs} : u \geq \sigma_{\max}(\text{mat}(w))\}$, where $\sigma_{\max}$ is the largest singular value function, is the epigraph of the spectral norm on $\mathbb{R}^{r\times s}$, assuming $r \leq s$ without loss of generality. Similarly, $K^*_{\ell_{\text{spec}}}$ is the epigraph of the matrix nuclear norm (i.e. the sum of singular values).

Matrix square cone. $K_{\text{matsqr}} := \left\{(u,v,w) \in \mathbb{R}^{\text{sd}(r)} \times \mathbb{R}_+ \times \mathbb{R}^{rs} : U \in \mathbb{S}^r, 2Uv \succeq WW^\top\right\}$, where $U := \text{mat}(u)$ and $W := \text{mat}(w) \in \mathbb{R}^{r\times s}$, is the homogenized symmetric matrix epigraph of the symmetric outer product, assuming $r \leq s$ without loss of generality [Güler and Tunçel, 1998].

Generalized power cone. $K_{\text{gpow}} := \left\{(u,w) \in \mathbb{R}^r_+ \times \mathbb{R}^s : \prod_{i\in[r]} u_i^{\alpha_i} \geq \|w\|\right\}$, parametrized by exponents $\alpha \in \mathbb{R}^r_{++}$ with $e^\top\alpha = 1$, is the generalized power cone [Chares, 2009, Section 3.1.2].

Power mean cone. $K_{\text{pow}} := \left\{(u,w) \in \mathbb{R} \times \mathbb{R}^d_+ : u \leq \prod_{i\in[d]} w_i^{\alpha_i}\right\}$, parametrized by exponents $\alpha \in \mathbb{R}^d_{++}$ with $e^\top\alpha = 1$, is the hypograph of the power mean on $\mathbb{R}^d_+$.

Geometric mean cone. K_{geo} is the hypograph of the geometric mean on $\mathbb{R}^d_+$, a special case of K_{pow} with equal exponents.

Root-determinant cone. $K_{\text{rtdet}} := \left\{(u,w) \in \mathbb{R}\times\mathbb{R}^{\text{sd}(d)} : W \in \mathbb{S}^d, u \leq (\det(W))^{1/d}\right\}$, where $W := \text{mat}(w)$, is the hypograph of the dth-root-determinant on $\mathbb{S}^d$.

Logarithm cone. $K_{\log} := \text{cl}\left\{(u,v,w) \in \mathbb{R} \times \mathbb{R}_{++} \times \mathbb{R}^d_{++} : u \leq \sum_{i\in[d]} v\log(w_i/v)\right\}$ is the hypograph of the perspective of the sum of logarithms on $\mathbb{R}^d_{++}$.

Log-determinant cone. $K_{\text{logdet}} := \text{cl}\{(u, v, w) \in \mathbb{R} \times \mathbb{R}_> \times \mathbb{R}^{\text{sd}(d)} : W \in \mathbb{S}^d, u \leq v \log\det(W/v)\}$, where $W := \text{mat}(w)$, is the hypograph of the perspective of the log-determinant on $\mathbb{S}^d$.

Separable spectral function cone. $K_{\text{sepspec}} := \text{cl}\{(u, v, w) \in \mathbb{R} \times \mathbb{R}_> \times \text{int}(Q) : u \geq v\varphi(w/v)\}$, where Q is $K_{\geq}$ or K (a cone of squares of a Jordan algebra), is the epigraph of the perspective of a convex separable spectral function $\varphi : \text{int}(Q) \to \mathbb{R}$, such as the sum or trace of the negative logarithm, negative entropy, or power in $(1, 2]$ (see Coey et al. [2021a] for more details). The complexity estimates in Table 2.1 depend on whether Q is $K_{\geq}$ or K.

Relative entropy cone. $K_{\text{relent}} := \text{cl}\left\{(u, v, w) \in \mathbb{R} \times \mathbb{R}^d_> \times \mathbb{R}^d_> : u \geq \sum_{i \in [d]} w_i \log(w_i/v_i)\right\}$ is the epigraph of vector relative entropy.

Matrix relative entropy cone. $K_{\text{matrelent}} := \text{cl}\{(u, v, w) \in \mathbb{R} \times \mathbb{R}^{\text{sd}(d)} \times \mathbb{R}^{\text{sd}(d)} : V \in \mathbb{S}^d, W \in \mathbb{S}^d, u \geq \text{tr}(W(\log(W) - \log(V)))\}$, where $V := \text{mat}(v)$ and $W := \text{mat}(w)$, is the epigraph of matrix relative entropy.[2]

Weighted sum-of-squares (WSOS) 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 WSOS 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 Yildiz [2019]. For simplicity, the complexity estimates in Table 2.1 assume $s_l = s, \forall l \in [r]$. Note that $s < d \leq s^2$. More detailed descriptions of these cones are given in Chapter 5.

Scalar WSOS cone. K_{SOS} is a cone of polynomials that are guaranteed to be nonnegative pointwise on the domain.

Symmetric matrix WSOS cone. K_{matSOS} is a cone of polynomial symmetric matrices (in an svec-like format) of side dimension t that are guaranteed

[2] The logarithmically homogeneous barrier for $K_{\text{matrelent}}$ that Hypatia uses is conjectured by Karimi and Tunçel [2020] to be self-concordant.

to belong to S pointwise on the domain. We let $m := st + d$ in Table 2.1 for succinctness.

ℓ_1 epigraph WSOS cone. $K_{\text{`}_{1\text{SOS}}}$ is a cone of polynomial vectors of length $1 + t$ that are guaranteed to belong to $K^*_{\text{`}_1}$ pointwise on the domain.

ℓ_2 epigraph WSOS cone. $K_{\text{`}_{2\text{SOS}}}$ is a cone of polynomial vectors of length $1 + t$ that are guaranteed to belong to $K_{\text{`}_2}$ pointwise on the domain.

For each cone, we have an analytic form for the feasibility check, gradient, Hessian, and TOO oracles defined in Section 2.2. 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 2.1, 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 2.1 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 2.7.3 demonstrate that the TOO is very rarely an algorithmic bottleneck in practice.

Our TOO in (2.2) is distinct from the 'higher order corrector' terms proposed by Mehrotra [1992], Dahl and Andersen [2021]. The method by Mehrotra [1992] only applies to symmetric cones, and Dahl and Andersen [2021] test their technique only for the standard exponential cone. Compared to the third order term proposed by Dahl and 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 K_{DNN} and K_{SOS}) that are defined as intersections of other cones. We leverage these

cone	dim(K)	ν	feasibility	gradient	Hessian	TOO
$K_{\geq}$	d	d	d	d	d	d
$K_{\text{}}$	sd(d)	d	d^3	d^3	d^4	d^3
K_{DNN}	sd(d)	sd(d)	d^3	d^3	d^4	d^3
K_{sPSD}	d	s	JD^2l	JD^2l	dJD^2l	JD^2l
K_{LMI}	d	s	$ds^2 + s^3$	ds^3	d^2s^2	$ds^2 + s^3$
$K_{\text{`}_1}$	$1 + d$	$1 + d$	d	d	d	d
$K_{\text{`}_2}$, K_{sqr}	$1 + d$	2	d	d	d^2	d
$K_{\text{`}_{\text{spec}}}$	$1 + rs$	$1 + r$	$r^2s + r^3$	$r^2s + r^3$	r^2s^2	rs^2
K_{matsqr}	sd(r) $+ 1 + rs$	$1 + r$	$r^2s + r^3$	$r^2s + r^3$	r^2s^2	rs^2
K_{gpow}	$r + s$	$1 + r$	$r + s$	$r + s$	$r^2 + s^2$	$r + s$
K_{pow}, K_{geo}	$1 + d$	$1 + d$	d	d	d^2	d
K_{rtdet}	$1 +$ sd(d)	$1 + d$	d^3	d^3	d^4	d^3
K_{log}	$2 + d$	$2 + d$	d	d	d^2	d
K_{logdet}	$2 +$ sd(d)	$2 + d$	d^3	d^3	d^4	d^3
K_{sepspec}-$K_{\geq}$	$2 + d$	$2 + d$	d	d	d^2	d
K_{sepspec}-K	$2 +$ sd(d)	$2 + d$	d^3	d^3	d^5	d^3
K_{relent}	$1 + 2d$	$1 + 2d$	d	d	d^2	d
$K_{\text{matrelent}}$	$1 + 2$ sd(d)	$1 + 2d$	d^3	d^3	d^5	d^4
K_{SOS}	d	sr	ds^2r	ds^2r	d^2sr	ds^2r
K_{matSOS}	d sd(t)	str	ms^2t^2r	ds^2t^2r	d^2st^3r	ms^2t^2r
$K_{\text{`}_1\text{SOS}}$	$d(1 + t)$	str	ds^2tr	ds^2tr	d^2str	ds^2tr
$K_{\text{`}_2\text{SOS}}$	$d(1 + t)$	$2sr$	ds^2tr	ds^2tr	d^2st^2r	ds^2t^2r

Table 2.1: Cone dimension dim(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 2.5.

properties to obtain fast and numerically stable TOO implementations.

This is illustrated in Section 4.1, where we define LHSCBs and derive e□cient TOO procedures for a class of cones that can be characterized as intersections of slices of the PSD cone K . In Chapter 3, we derive e□cient TOO procedures for a class of spectral function cones on positive domains (K_{sepspec}, K_{log}, K_{logdet}, K_{geo}, K_{rtdet}).

2.6 Preprocessing and solving for search directions

We discuss preprocessing and initial point finding procedures and 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 2.7, to give context for these results. These techniques are likely to be useful for other conic PDIPM implementations.

Given a conic model specified in the general primal conic form (2.4), we first rescale the primal and dual equality constraints (2.4b) and (2.5b) to improve the conditioning of the a□ne data. Next, we perform a QR factorization of $A^\top$ 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 (2.14c). Since y is eliminated, we do not need to solve (2.15b) for y^0.

Starting from the initial interior point ω^0 defined in Section 2.4.1, we perform PDIPM iterations until the convergence conditions in Section 2.4.3 (in the preprocessed space) are met. Finally, we reuse 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 PDIPM iteration, we solve the linear system (2.25) 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 2.4.5. 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 (2.25), 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 Section 4.1 and Coey et al. [2021a], 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.

2.7 Computational testing

In Section 2.7.1, we introduce a diverse set of exotic conic benchmark instances generated from a variety of applied examples. In Section 2.7.2, we describe our methodology for comparing the stepping procedures from Section 2.4.5, and in Section 2.7.3 we examine our computational results.

2.7.1 Exotic conic benchmark set

We generate 379 instances (in our primal general form (2.4)) 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 2.5. 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 2-1 shows the distributions of instance dimensions and exotic cone counts. All instances have at least one cone (note any $K_{\geq}$ cones are concatenated together, so $K_{\geq}$ is counted at most once) and take at least one iteration to solve with Hypatia.

Below we briefly introduce each example. In Table 2.2, 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 $K^*_{\cdot_1}$ are only listed as using $K_{\cdot_1}$). For some examples, we describe a subset of formulations in Coey et al. [2021d] and Chapter 5. 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.

Classical-quantum capacity. Compute the capacity of a classical-to-quantum channel (adapted from Fawzi and Fawzi [2018, Section 3.1]).

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

Contraction analysis. Find a contraction metric that guarantees global stability of a dynamical system (adapted from Aylward et al. [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 (compare to [Papp and Alizadeh, 2014, Section 4.3], see Coey et al. [2021d, Section 5.6]).

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 Coey et al. [2021d, Section 5.4]).

Entanglement-assisted capacity. Compute the entanglement-assisted classical capacity of a quantum channel (adapted from Fawzi and 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]).

Linear program. Solve a simple linear program.

Lotka-Volterra. Find an optimal controller for a Lotka-Volterra model of population dynamics (adapted from Korda et al. [2016, Section 7.2]).

Lyapunov stability. Minimize an upper bound on the root mean square gain of a dynamical system (adapted from Boyd 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 (compare to Agrawal et al. [2019, Equation 8]; see Coey et al. [2021d, Section 5.2]).

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 Coey et al. [2021d, Section 5.3]).

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

Nearest correlation matrix. Compute the nearest correlation matrix in the quantum relative entropy sense (adapted from Fawzi et al. [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 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.

Norm cone polynomial. Given a vector of polynomials, check a sufficient condition for pointwise membership in $K_{n,2}$ or $K^*_{n,1}$. 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 (see Papp and Yildiz [2019, Section 7.2.1]).

Polynomial minimization. Compute a lower bound for a given polynomial over a given semialgebraic set (see Papp and Yildiz [2019, Section 7.3.1] and Coey et al. [2021d, Section 5.5]). 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 Chapter 5).

Portfolio. Maximize the expected returns of a stock portfolio and satisfy various risk constraints (see Coey et al. [2021d, Section 5.1]).

Region of attraction. Find the region of attraction of a polynomial control system (see 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 Fawzi and Fawzi [2018, Section 4]).

Robust geometric programming. Bound the worst-case optimal value of an uncertain signomial function with a given coe□cient uncertainty set (adapted from Chandrasekaran and Shah [2017, Equation 39]).

Semidefinite polynomial matrix. Check a su□cient 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 Coey et al. [2021d, Section 5.7]). Several instances use real datasets from Mazumder et al. [2019].

Signomial minimization. Compute a global lower bound for a given signomial function (see Murray et al. [2020]). Several instances use signomials with known optimal values from Murray et al. [2020], 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 (see 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]).

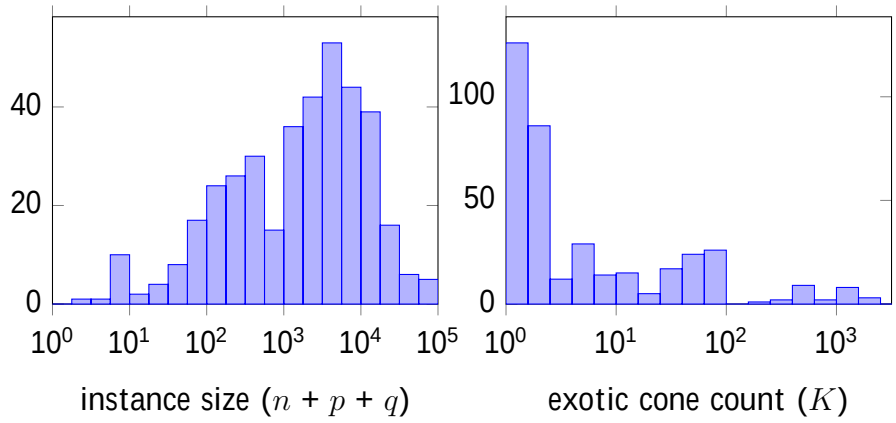

Figure 2-1: Histograms summarizing the benchmark instances in the primal conic form (2.4). Instance size (log scale) is the sum of the primal variable, equality, and conic constraint dimensions. Exotic cone count (log scale) is the number of exotic cones comprising the Cartesian product cone.

2.7.2 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 PDIPM iteration count and solve time. Across the benchmark set, we compare performance between consecutive pairs of the five stepping procedures outlined in Section 2.4.5.

basic. The basic prediction or centering stepping procedure without any enhancements; described in Section 2.4.5, 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], Papp and Yıldız [2017].

prox. The *basic* procedure modified to use a less restrictive central path proximity condition; described in Section 2.4.5.

example	#	cones in at least one instance
CBLIB	10	$K_{\geq}$ K^{*} K_{ℓ_2} K_{sqr} $K_{\log}$ K_{gpow}
central polynomial matrix	24	$K_{\geq}$ K^{*} K_{sqr} K_{gpow} K_{rtdet} $K_{\log}$ K_{sepspec}
classical-quantum capacity	9	$K_{\geq}$ K^{*} $K_{\log}$ K_{sepspec}
condition number	6	$K_{\geq}$ K^{*} K_{LMI}
contraction analysis	8	K K^{*}_{matSOS}
convexity parameter	7	K K^{*}_{matSOS}
covariance estimation	13	$K_{\geq}$ K^{*} K_{sqr} K_{gpow} K_{rtdet} $K_{\log}$ K_{sepspec}
density estimation	16	$K_{\geq}$ K^{*} K_{sqr} K_{geo} $K_{\log}$ K_{SOS}
discrete maximum likelihood	7	$K_{\geq}$ K_{pow} $K_{\log}$ K_{sepspec}
D-optimal design	16	$K_{\geq}$ K^{*} K_{ℓ_1} K_{ℓ_2} K_{sqr} K_{geo} K_{rtdet} $K_{\log}$ K_{logdet}
entanglement-assisted capacity	3	K K_{sepspec} $K_{\text{matrelent}}$
experiment design	13	$K_{\geq}$ K^{*} K_{sqr} K_{gpow} K_{rtdet} $K_{\log}$ K_{sepspec}
linear program	3	$K_{\geq}$
Lotka-Volterra	3	K
Lyapunov stability	10	K K^{*}_{matsqr}
matrix completion	11	$K_{\geq}$ K^{*} K_{sqr} $K_{\ell_{\text{spec}}}$ K_{gpow} K_{geo} $K_{\log}$
matrix quadratic	8	K K^{*}_{matsqr}
matrix regression	11	$K_{\geq}$ K^{*} K_{ℓ_1} K_{ℓ_2} K_{sqr} $K_{\ell_{\text{spec}}}$
maximum volume hypercube	15	$K_{\geq}$ K_{ℓ_1} K_{ℓ_2} K_{sqr} K_{geo}
nearest correlation matrix	3	$K_{\text{matrelent}}$
nearest polynomial matrix	8	K K_{SOS} K_{matSOS}
nearest PSD matrix	28	K K_{sPSD}
nonparametric distribution	10	$K_{\geq}$ K_{sqr} K_{geo} $K_{\log}$ K_{sepspec}
norm cone polynomial	10	$K_{\ell_1\text{SOS}}$ $K_{\ell_2\text{SOS}}$
polynomial envelope	7	K_{SOS}
polynomial minimization	15	K K_{SOS}
polynomial norm	10	K_{SOS} K_{matSOS} $K_{\ell_1\text{SOS}}$ $K_{\ell_2\text{SOS}}$
portfolio	9	$K_{\geq}$ K_{ℓ_1} K_{ℓ_2}
region of attraction	6	K K_{SOS}
relative entropy of entanglement	6	K $K_{\text{matrelent}}$
robust geometric programming	6	$K_{\geq}$ K_{ℓ_1} $K_{\log}$ K_{relent}
semidefinite polynomial matrix	18	K K_{ℓ_2} K_{matSOS}
shape constrained regression	11	$K_{\geq}$ K^{*} K_{ℓ_1} K_{ℓ_2} K_{SOS} K_{matSOS}
signomial minimization	13	$K_{\geq}$ $K_{\log}$ K_{relent}
sparse LMI	15	K K_{sPSD} K_{LMI}
sparse principal components	6	$K_{\geq}$ K^{*} K_{ℓ_1}
stability number	6	$K_{\geq}$ K^{*} K_{DNN}

Table 2.2: For each example, the count of instances and list of exotic cones (defined in Section 2.5) used in at least one instance.

TOA. The *prox* procedure with the TOA enhancement to incorporate third order LHSCB information; described in Section 2.4.5.

curve. The *TOA* procedure adapted for a single backtracking search on a curve instead of two backtracking line searches; described in Section 2.4.5.

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 2.4.5.

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 0.5.1 (with default options), on dedicated hardware with an AMD Ryzen 9 3950X 16-core processor (32 threads) and 128GB of RAM. In Section 2.6, 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 su□cient progress towards meeting the convergence conditions in Section 2.4.3.

Since some instances are more numerically challenging than others, we set the termination tolerances (described in Section 2.4.3) 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:

$$M(v, s) := \prod_{i \in \{1,\ldots,d\}} (v_i + s)^{1=d} - s. \tag{2.41}$$

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 (2.41)) 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 2.7.3.

Table 2.3 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 2.3, as some instances solve very quickly (see Figure 2-2).

Table 2.4 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 2.6).

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 e□ciently compute at least one direction (such as updating and factorizing the positive definite matrix in Section 2.6).

RHS. Performed between one and four times per iteration, depending on the stepping procedure. Includes updating an RHS vector (see (2.25)) for the linear system for search directions. Note that the TOO is only evaluated while computing the centering TOA RHS (2.29b) and the prediction TOA RHS (2.39b).

direc. Performed for each RHS vector. Includes solving the linear system for a search direction (see (2.25)) using the data computed during *LHS* and a single RHS vector computed during *RHS*, and performing iterative refinement on the direction (see Section 2.6).

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.

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 2.4, *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 2-3 and 2-6 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 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 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.

2.7.3 Results

Table 2.3 and Figure 2-6 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 2-2, which shows the distributions of iteration counts and solve times for the *basic* and *comb* stepping procedures. We note that Figure 2-4 (left) supports the intuition that formulation size is strongly positively correlated with solve time for *comb*.

Overall, Table 2.3 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 2-3 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 2-5 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.

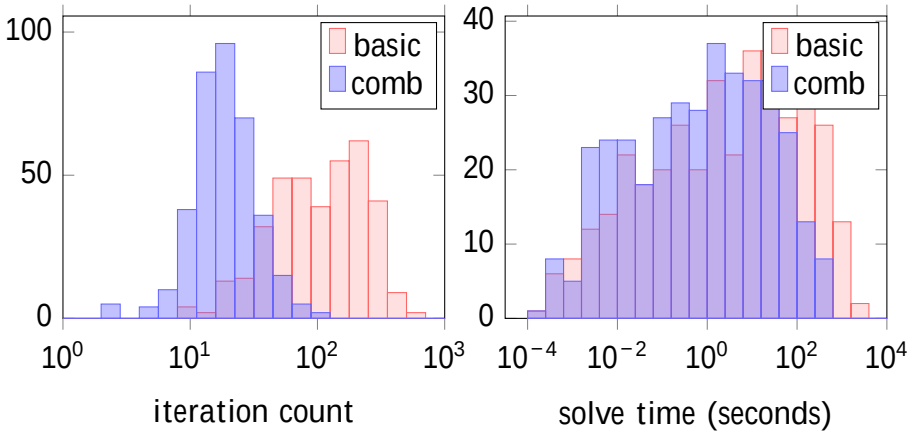

Figure 2-2: Overlayed histograms of iteration count (left, log scale) and solve time (right, log scale, in seconds) for the *basic* and *comb* stepping procedures, excluding instances that fail to converge.

		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

Table 2.3: For each stepping procedure, the number of converged instances and shifted geometric means of iterations and solve times (in milliseconds).

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.

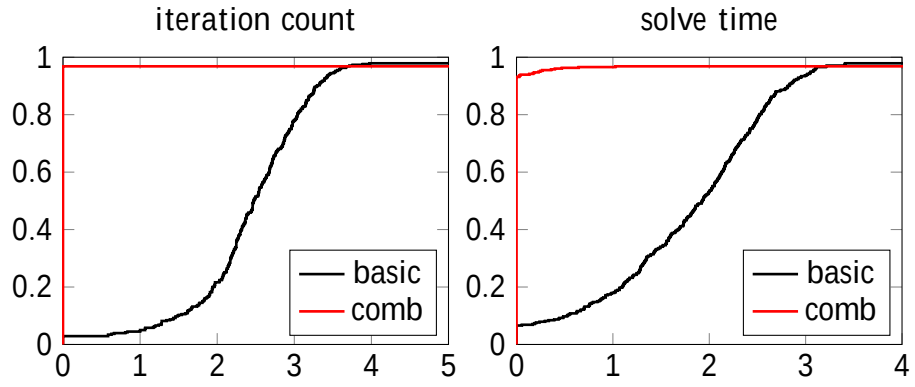

Figure 2-3: Performance profiles (see Section 2.7.2) of iteration count (left) and solve time (right) for the four stepping enhancements overall.

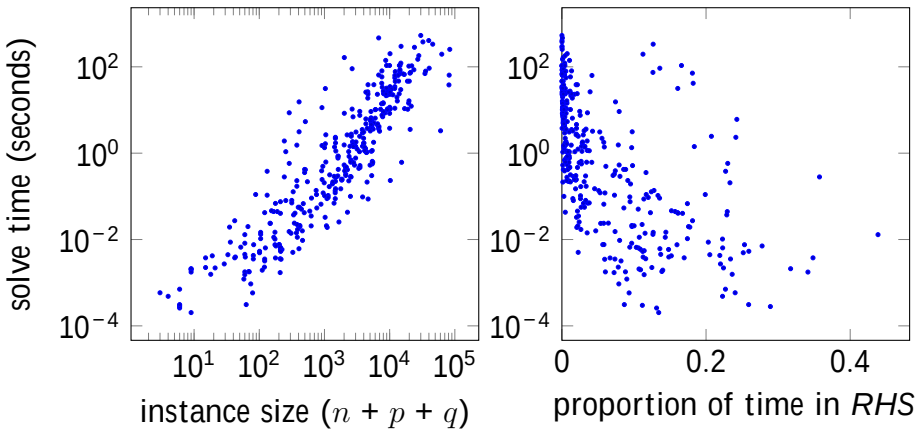

Figure 2-4: Solve time (log scale, in seconds) for the *comb* stepping procedure against (left) instance size (log scale) and (right) the proportion of solve time spent in *RHS*, excluding instances that fail to converge.

Less restrictive proximity

We compare *basic* and *prox* to evaluate the central path proximity enhancement introduced in Section 2.4.5. Figure 2-6 (first row) shows that the iteration count and solve time improve for nearly all instances. From Table 2.3, 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 2-6 and also between the per iteration subtimings in Table 2.4 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

			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

Table 2.4: For each stepping procedure, the shifted geometric means of subtimings (in milliseconds) for the key algorithmic components.

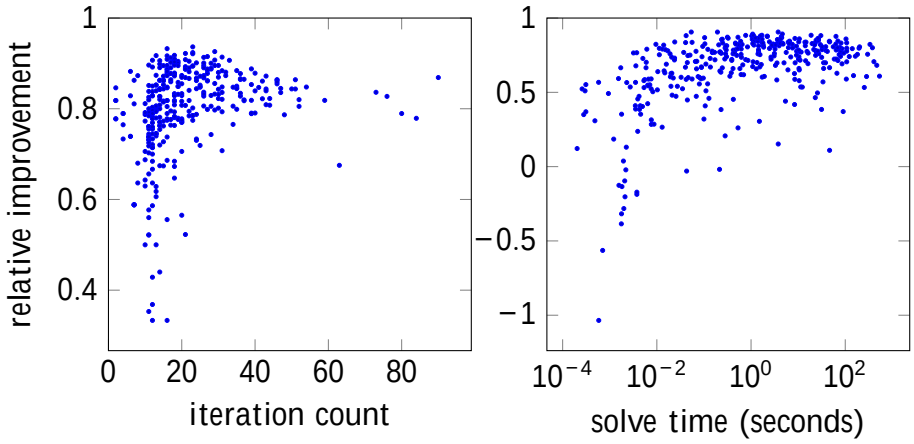

Figure 2-5: Relative improvement, from *basic* to *comb*, in iteration count (left) or solve time (right) against iteration count or solve time (in seconds) respectively for *comb*, over the 356 instances on which both *basic* and *comb* converge.

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], Papp and Yıldız [2021] are too conservative from the perspective of practical

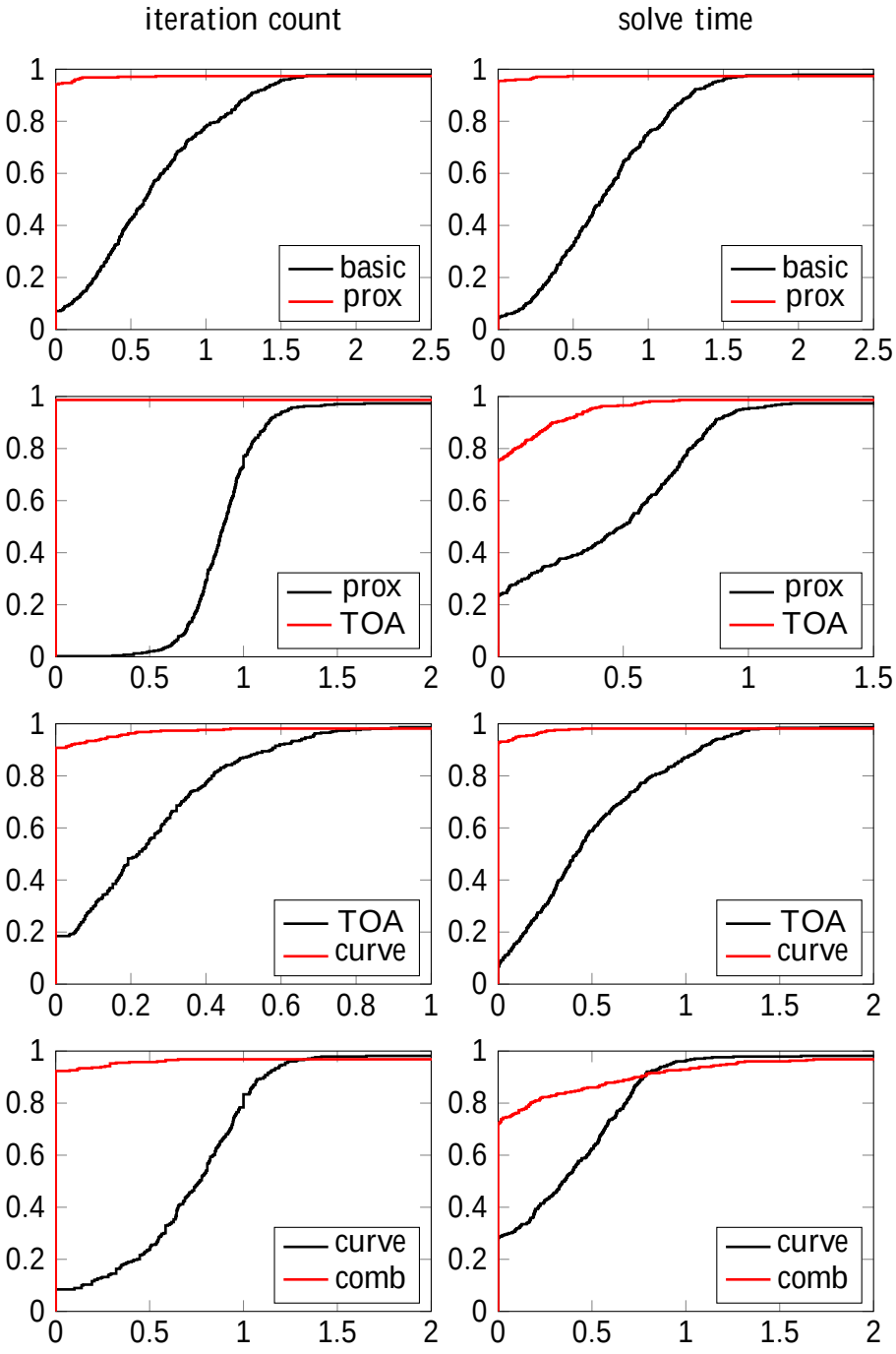

Figure 2-6: Performance profiles (see Section 2.7.2) of iteration count (left column) and solve time (right column) for the four stepping enhancements (rows).

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.

Third order adjustments

We compare *prox* and *TOA* to evaluate the TOA enhancement introduced in Section 2.4.5. Figure 2-6 (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 2.3, 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 2.4 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.

Curve search

We compare *TOA* and *curve* to evaluate the curve search enhancement introduced in Section 2.4.5. Figure 2-6 (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 2.3, 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 2.4 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.

Combined directions

Finally, we compare *curve* and *comb* to evaluate the combined directions enhancement introduced in Section 2.4.5. Figure 2-6 (fourth row) shows that the iteration count and solve time improve on around 90% and 70% of instances respectively. From Table 2.3, 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 2.4. 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 2.6). Furthermore, Figure 2-4 (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 3

Epigraphs of spectral functions

This chapter is based on the submitted paper Coey et al. [2021a]. We chose to rely on Jordan algebraic notation for this work. Here symmetric cones are characterized via their description as cones of squares of a Euclidean Jordan algebra.

3.1 Introduction

A class of functions that commonly arise in convex optimization applications are 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 et al., 2006, Grant and Boyd, 2014]. We define *spectral cones* as proper cones defined from epigraphs of convex homogeneous spectral functions or epigraphs of perspective functions of convex spectral functions. These cones allow simple, natural conic reformulations of a wide range of convex optimization problems. Despite this, to our knowledge there has been little prior work enabling direct support for various spectral cones in primal-dual conic solvers. For many spectral cones (e.g. for the negative entropy function), equivalent EFs using only symmetric cones do not exist, and when they do, they can

be impractically large.

Recall that Hypatia allows specifying a proper cone K by implementing a small list of oracles. Once K is defined, both K and its dual cone K^* may be used in any combination with other recognized cones in Hypatia to construct conic models. The oracles to implement are: an initial interior point $t \in \text{int}(K)$, a feasibility test for the cone interior int(K) (and optionally for the dual cone interior int(K^*)), and several derivative oracles for an LHSCB for the cone. The LHSCB oracles needed for ideal performance are the gradient, the Hessian operator (i.e. the second order directional derivative applied once to a given direction), the inverse of the Hessian operator, and the third order directional derivative (applied twice to a given direction). Fast and numerically stable procedures for evaluating oracles are crucial for practical performance in conic PDIPM solvers such as Hypatia.

Our first main contribution is to define simple logarithmically homogeneous barriers for spectral cones and derive efficient and numerically stable barrier oracle procedures. For example, for the case where the spectral function of the cone 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 cones, we derive highly-efficient specialized oracle procedures.

Our second main contribution is to show that for two important subclasses of spectral 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 separable spectral functions, in particular the trace (or sum) of the negative logarithm (i.e. the negative log-determinant), negative entropy, and certain power functions. Furthermore, the dual cones of the MMD cones allow modeling epigraphs of even more separable spectral functions, such as the trace of the inverse, exponential, and more power functions.

These two contributions enable efficient and numerically stable implementations of the MMD cone and the log-determinant and root-determinant cones in nonsymmetric conic PDIPMs. 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.

We use these new spectral cones in Hypatia to 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 equivalent EFs written in terms of the handful of standard cones recognized by either ECOS or MOSEK 9 (i.e. the common symmetric cones and the three-dimensional exponential and power cones). 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 closed-form formula for the MMD cone inverse Hessian product is faster and more numerically reliable than a naive direct solve using an explicit Hessian factorization.

3.1.1 Chapter 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 PDIPM. 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 e□cient 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 e□cient 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 $\quad : V \times V \to V$. For $w \in V$, $w^2 := w \quad w$. We refer to V as a *Jordan algebra* if for all $w_a, w_b \in V$:

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

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

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

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

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

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 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 [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 (unlike all other chapters, here e is not just a vector of ones). Note that $P(w)e = w^2$. 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 $\langle w_a \quad w_b, w_c \rangle = \langle w_b, w_a \quad w_c \rangle$ for all $w_a, w_b, w_c \in V$.

We call $\mathcal{Q}$ a *cone of squares* on V if $\mathcal{Q} = \{w \quad w : w \in V\}$. 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 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 $a \quad b$ instead of $a - b \in \mathcal{Q}$, or $a \quad b$ instead of $a - b \in \text{int}(\mathcal{Q})$, where $\mathcal{Q}$ is clear from context. If $w \quad 0$, 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 $P(w^{1=2}) = P(w)^{1=2}$ 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 [Vieira, 2007, Page 27]. Thus for $w \in \operatorname{int}(Q)$ 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 any Jordan frame in V, and define for $i, j \in \{1, \ldots, d\}$:

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

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

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

V has the direct sum decomposition $V = \bigoplus_{i;j \in \{1,\ldots,d\}: i \le j} V_{i;j}$ [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 $V_{i;i} = \{tE_{i;i} : t \in \mathbb{R}\}$ 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:

$$r = \sum_{i;j \in \{1,\ldots,d\}: i \le j} r_{i;j} = \sum_{i;j \in \{1,\ldots,d\}: i < j} r_{i;j} + \sum_{i \in \{1,\ldots,d\}} r_i c_i, \tag{3.8}$$

where $r_i = \langle r, c_i \rangle \in \mathbb{R}$ and $r_{i;j} \in V_{i;j}$ for all $i, j \in \{1, \ldots, d\}$. Each $r_{i;j}$ is a projection of r onto $V_{i;j}$, where:

$$r_{i;i} = r_i c_i = P(c_i)r \qquad \forall i \in \{1, \ldots, d\}, \tag{3.9a}$$

$$r_{i;j} = 4L(c_i)L(c_j)r = 4c_i \circ (c_j \circ r) \qquad \forall i, j \in \{1, \ldots, d\} : j \neq i. \tag{3.9b}$$

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 $i, j \in \{1, \ldots, d\}$.

We list some facts relating to compositions of projection operators (see Faraut

and Koranyi [1998, Theorem IV.2.2] and Sun and Sun [2008, page 430]):

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

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

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

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

$$\sum_{i,j \in [d]: i<j} 4L(c_i)L(c_j) + \sum_{i \in [d]} P(c_i) = L(e). \tag{3.10e}$$

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

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

The inverse operator Γ^{-1} is given by:

$$\Gamma^{-1} = \sum_{i,j \in [d]: i<j} 4\lambda_{i,j}^{-1} L(c_i)L(c_j) + \sum_{i \in [d]} \lambda_{i,i}^{-1} P(c_i). \tag{3.12}$$

It can be verified using (3.10) that for any $r \in V$, $\Gamma^{-1} r = \Gamma^{-1} \Gamma r = r$. 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:

$$\Gamma = \sum_{i,j \in [d]: i<j} 4w_i w_j L(c_i)L(c_j) + \sum_{i \in [d]} w_i^2 P(c_i) = P(w), \tag{3.13a}$$

$$\Gamma^{-1} = \sum_{i,j \in [d]: i<j} 4w_i^{-1} w_j^{-1} L(c_i)L(c_j) + \sum_{i \in [d]} w_i^{-2} P(c_i) = P(w^{-1}). \tag{3.13b}$$

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 Sun and Sun [2008, Theorem 4.1], the gradient of φ at w is:

$$\nabla\varphi(w) = \sum_{i \in \mathbb{J}d\mathbb{K}} (\nabla f)_i c_i. \tag{3.14}$$

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

$$\nabla^2\varphi(w)[r] = \sum_{i,j \in \mathbb{J}d\mathbb{K}: i<j} \frac{(\nabla f)_i - (\nabla f)_j}{w_i - w_j} r_{i;j} + \sum_{i,j \in \mathbb{J}d\mathbb{K}} (\nabla^2 f)_{i;j} r_i c_j. \tag{3.15}$$

Sun and 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:

$$w(t) := w + tr = \sum_{i \in \mathbb{J}d\mathbb{K}} w_i(t) c_i(t), \tag{3.16}$$

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 (3.15):

$$\frac{\mathrm{d}}{\mathrm{d}t}\nabla\varphi(w(t)) = \nabla^2\varphi(w(t))[r] \tag{3.17a}$$

$$= \sum_{i,j\in[d]:i<j} \frac{(\nabla f(t))_i - (\nabla f(t))_j}{w_i(t) - w_j(t)} r_{i,j}(t) + \sum_{i,j\in[d]} (\nabla^2 f(t))_{i,j} r_i(t) c_j(t). \tag{3.17b}$$

We differentiate (3.17) once more. From Vieira [2016, Corollary 1 and Theorem 3.3] and Sun and Sun [2008, Equation 37], for $i \in [d]$ we have:

$$\frac{\mathrm{d}}{\mathrm{d}t} w_i(t) = r_i(t), \tag{3.18a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} c_i(t) = s_i(t) := \sum_{j\in[d]:j\neq i} \frac{r_{i,j}(t)}{w_i(t) - w_j(t)}. \tag{3.18b}$$

Using the chain rule, (3.18a) implies:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\nabla f(t))_i = \sum_{k\in[d]} (\nabla^2 f(t))_{i,k} r_k(t) \qquad \forall i \in [d], \tag{3.19a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\nabla^2 f(t))_{i,j} = \sum_{k\in[d]} (\nabla^3 f(t))_{i,j,k} r_k(t) \qquad \forall i,j \in [d]. \tag{3.19b}$$

Applying the chain and product rules, we have:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{1}{w_i(t) - w_j(t)} = \frac{r_j(t) - r_i(t)}{(w_i(t) - w_j(t))^2} \qquad \forall i,j \in [d] : i \neq j, \tag{3.20a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} r_{i,j}(t) = \frac{\mathrm{d}}{\mathrm{d}t}(4c_i(t) \quad (c_j(t) \quad r)) \tag{3.20b}$$

$$= 4c_i(t) \quad (s_j(t) \quad r) + 4s_i(t) \quad (c_j(t) \quad r) \qquad \forall i,j \in [d] : i \neq j, \tag{3.20c}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle c_i(t), r\rangle c_j(t) = \langle s_i(t), r\rangle c_j(t) + r_i(t)s_j(t) \qquad \forall i,j \in [d]. \tag{3.20d}$$

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

$$\nabla^3 \varphi[r, r] = \frac{\mathrm{d}^2}{\mathrm{d}t^2} \nabla \varphi(w(t))\big|_{t=0} \tag{3.21a}$$

$$= \sum_{i,j \in [d]: i<j} \frac{(\nabla f)_i - (\nabla f)_j}{w_i - w_j} \left(4c_i \otimes (s_j \otimes r) + 4s_i \otimes (c_j \otimes r) - \frac{r_i - r_j}{w_i - w_j} r_{i,j} \right) + \tag{3.21b}$$

$$\sum_{i,j,k \in [d]: i<j} \frac{(\nabla^2 f)_{i,k} - (\nabla^2 f)_{j,k}}{w_i - w_j} r_k r_{i,j} +$$

$$\sum_{i,j \in [d]} (\nabla^2 f)_{i,j} (\langle s_i, r \rangle c_j + r_i s_j) + \sum_{i,j,k \in [d]} (\nabla^3 f)_{i,j,k} r_i r_k c_j$$

$$= \sum_{i,j \in [d]: i<j} \frac{(\nabla f)_i - (\nabla f)_j}{w_i - w_j} \left(4c_i \otimes (s_j \otimes r) + 4s_i \otimes (c_j \otimes r) - \frac{r_i - r_j}{w_i - w_j} r_{i,j} \right) + \tag{3.21c}$$

$$\sum_{i,j \in [d]} (\nabla^2 f)_{i,j} (2r_j s_i + \langle s_i, r \rangle c_j) + \sum_{i,j,k \in [d]} (\nabla^3 f)_{i,j,k} r_i r_k c_j .$$

The derivative expressions simplify significantly for $V = \mathbb{R}^d$. For $V = \mathbb{S}^d$, the form of (3.15) is well-known [Faybusovich and Zhou, 2021] and the form of (3.21c) 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 $h(w) := \sum_{i \in [d]} h(w_i) c_i$. This allows us to write $\varphi(w) = \mathrm{tr}(h(w)) = \sum_{i \in [d]} h(\lambda_i)$. Note that φ is convex if and only if h is convex. For example, if $h(w) = -\log(w)$ then $\varphi(w) = \mathrm{tr}(-\log(w)) = -\mathrm{logdet}(w)$; we consider this special case in Section 3.3.3.

We specialize the derivatives from (3.14), (3.15) and (3.21c), maintaining the simplifying assumption of distinct eigenvalues. Since $(\nabla^2 f)_{i,j} = (\nabla^3 f)_{i,j,k} = 0$ unless

$i = j = k$, we have:

$$r\,\varphi(w) = \sum_{i \in [d]} r\,h(w_i)c_i, \tag{3.22a}$$

$$r^2\varphi(w)[r] = \sum_{i,j \in [d]:i<j} \frac{r\,h(w_i) - r\,h(w_j)}{w_i - w_j} 4c_i \quad (c_j \quad r) + \sum_{i \in [d]} r^2 h(w_i) P(c_i) r, \tag{3.22b}$$

$$r^3\varphi(w)[r,r] = \tag{3.22c}$$

$$\sum_{i,j \in [d]:i<j} \frac{r\,h(w_i) - r\,h(w_j)}{w_i - w_j} \left(4c_i \quad (s_j \quad r) + 4s_i \quad (c_j \quad r) - \frac{r_i - r_j}{w_i - w_j} r_{i,j} \right) + \tag{3.22d}$$

$$\sum_{i \in [d]} r^2 h(w_i)(2r_i s_i + \langle s_i, r \rangle c_i) + \sum_{i \in [d]} r^3 h(w_i) r_i^2 c_i. \tag{3.22e}$$

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 $\Lambda := P(w^{-1/2})r \in V$. First, note that (similar to Vieira [2007, Lemma 3.3.4]):

$$\langle w^{-1}, r \rangle = \langle P(w^{-1/2})e, r \rangle \overset{3.6}{=} \langle e, P(w^{-1/2})r \rangle = \operatorname{tr}(\Lambda). \tag{3.23}$$

Due to Faraut and Koranyi [1998, Proposition II.2.3]:

$$r_w(\operatorname{tr}(\Lambda)) = r_w(w^{-1})[r] = -P(w^{-1})r. \tag{3.24}$$

Adapting the result in Faybusovich and Tsuchiya [2017, Lemma 3.4]:

$$r_w(P(w^{-1})r)[r] = -2P(w^{-1/2})\Lambda^2. \tag{3.25}$$

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

$$r\,\varphi(w) = -w^{-1}, \tag{3.26}$$

so the second and third order directional derivatives are:

$$r^2\varphi(w)[r] \overset{3.24}{=} P(w^{-1})r, \tag{3.27a}$$

$$r^3\varphi(w)[r,r] \overset{3.25}{=} -2P(w^{-1=2})(P(w^{-1=2})r)^2. \tag{3.27b}$$

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:

$$K := \mathrm{cl}\{\varpi \in E : \zeta(\varpi) \geq 0\} \quad \forall, \tag{3.28}$$

where $\zeta : E \to \mathrm{R}$ is a concave, (degree one) homogeneous function and E is some convex cone in the space $\forall$. 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 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 $\zeta(u,w) := u - \varphi(w)$ is concave and homogeneous, and we let $E := \mathrm{R} \times \mathrm{int}(Q)$ and $\forall := \mathrm{R} \times V$. This defines a convex cone that is the closure of the epigraph set of φ:

$$K_h := \mathrm{cl}\{(u,w) \in \mathrm{R} \times \mathrm{int}(Q) : u \geq \varphi(w)\}. \tag{3.29}$$

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 int(K_h), we first determine whether w *2* int(Q) (which is equivalent to positivity of the eigenvalues), and if so, whether $\zeta(\text{¤}) > 0$.

3.4.2 The non-homogeneous case

Now we suppose that φ is convex and non-homogeneous. We define the perspective function of φ, per φ : R$_>$ × int(Q) *!* R, as (per φ)(v, w) := $v\varphi(v^{-1}w)$. This is a homogeneous and convex function [Boyd and Vandenberghe, 2004, Section 3.2.6]. We let $\zeta(u, v, w)$:= u − (per φ)(v, w), with E := R × R$_>$ × int(Q) and $\mathbb{V}$:= R × R × V. This defines a convex cone that is the closure of the epigraph set of the perspective function of φ:

$$K_p := \text{cl}\{(u, v, w) \in \mathbb{R} \times \mathbb{R}_> \times \text{int}(Q) : u \geq v\varphi(v^{-1}w)\}. \tag{3.30}$$

Equivalently, we can view K_p as the closed conic hull of the epigraph set of φ [Nesterov and Nemirovskii, 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 int(K_p) is similar to that of int(K_h) except we first check whether $v > 0$.

3.4.3 Dual cones

Recall the definition for the dual cone K^* of a cone K from (1.2). When K is defined through Hypatia's generic cone interface, both K and K^* become available for constructing conic models.

We assume that φ is convex, and we derive the dual cones of the epigraph cones K_h and K_p (these steps can be adapted for analogous hypograph cones if φ is concave). The convex conjugate function φ^* : V *!* R *[1* of φ is defined as in (1.4). 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 K_p in (3.30), Zhang [2004, Theorem 3.2] and Rockafellar [2015, Theorem 14.4] derive the dual cone K_p^*:

$$K_p^* = \mathrm{cl}\{(u, v, w) \in \mathbb{R}_> \times \mathbb{R} \times V : v \geq u\varphi^*(u^{-1}w)\}. \tag{3.31}$$

We can view 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 (3.30)). Depending on the natural domain of φ^*, the w component of K_p^* is not necessarily restricted to lie in 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 $\mathrm{int}(Q)$.

For K_h in (3.29), we derive the dual cone 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 K_p for φ is not a primitive cone, as it can be written as a (permuted) Cartesian product of $\mathbb{R}_\geq$ and K_h:

$$K_p = \mathrm{cl}\{(u, v, w) \in \mathbb{V} : v \in \mathbb{R}_\geq, (u, w) \in K_h\}. \tag{3.32}$$

Since the dual cone of a Cartesian product of cones is the Cartesian product of their dual cones, we have (since $\mathbb{R}_\geq^* = \mathbb{R}_\geq$):

$$K_p^* = \mathrm{cl}\{(u, v, w) \in \mathbb{V} : v \in \mathbb{R}_\geq, (u, w) \in K_h^*\}. \tag{3.33}$$

By [Lasserre, 1998, Theorem 2.1], the homogeneity of φ implies that φ^* can only take the values zero or infinity. Hence by (3.31), we know:

$$K_p^* = \mathrm{cl}\{(u, v, w) \in \mathbb{R}_> \times \mathbb{R} \times V : v \geq 0, u\varphi^*(u^{-1}w) < 1\}. \tag{3.34}$$

Since (3.33) and (3.34) describe the same cone, we can conclude that the dual cone

of K_h is:

$$K_h^* = \mathrm{cl}\{(u, w) \in \mathbb{R}_> \times V : \varphi^*(u^{-1}w) < 1\}. \tag{3.35}$$

3.4.4 Barrier functions and oracles

Recall that a logarithmically homogeneous barrier (LHB) function for a proper cone K is C^2-smooth and satisfies $(x_i) \to 1$ along every sequence $x_i \in \mathrm{int}(K)$ converging to the boundary of K, and:

$$(\theta x) = (x) - \nu \log(\theta) \qquad \forall x \in \mathrm{int}(K), \theta \in \mathbb{R}_+, \tag{3.36}$$

for some $\nu \geq 0$ [Nesterov and Nemirovskii, 1994, Definition 2.3.2]. If is also self-concordant, then it is an LHSCB with parameter $\nu \geq 1$ (or a ν-LHSCB) for K, which we define in (1.3).

The LHB we consider for a cone of the form (3.28) is:

$$(x) := -\log(\zeta(x)) + (x), \tag{3.37}$$

where can be thought of as an LHSCB for the domain of ζ or $\mathrm{cl}(E)$. The negative logarithm function $-\log$ is the standard LHSCB for $\mathbb{R}_\geq$, with parameter $\nu = 1$. Similarly, the spectral function $-\log\det$ (see Section 3.3.3) is the standard LHSCB for a cone of squares Q of V, with $\nu = d$ (the rank of V). For K_h we let $(x) = -\log\det(w)$, hence has parameter $\nu = 1 + d$. Since an LHSCB for a Cartesian product of cones is the sum of LHSCBs for the primitive cones, for K_p we let $(x) = -\log(v) - \log\det(w)$, hence has parameter $\nu = 2 + d$. Note that although is an LHB, it is not necessarily self-concordant; in Sections 3.6.4 and 3.7.3 we prove that is an LHSCB for some useful special cases.

We now define four barrier oracles that Hypatia's PDIPM uses; for ideal performance, these oracle implementations should be e□cient and numerically stable. For an interior point $x \in \mathrm{int}(K)$ and a direction $p \in V$, the gradient g, the Hessian product H, the inverse Hessian product H, and the third order directional derivative T

are:

$$g := r \quad (x), \tag{3.38a}$$

$$H := r^2 \quad (x)[p], \tag{3.38b}$$

$$H := (r^2 \quad (x))^{-1}[p], \tag{3.38c}$$

$$T := r^3 \quad (x)[p, p]. \tag{3.38d}$$

Note g, H, H, T 2 $\mathcal{V}$, and compare T here with T from (2.2). In later sections, we use subscripts to refer to subcomponents of these oracles, for example the w component of the gradient oracle is $g_w := r_w \quad (x)$ 2 V. Ideally, H applies the positive definite linear operator $r^2 \quad (x) : \mathcal{V} \; ! \; \mathcal{V}$ without constructing an explicit Hessian, and similarly, H applies the (unique) inverse operator $(r^2 \quad (x))^{-1} : \mathcal{V} \; ! \; \mathcal{V}$ without constructing or factorizing an explicit Hessian.

We note that for the standard LHSCB for a cone of squares, e□cient and numerically stable procedures for these four oracles are well-known. The same cannot be said for the LHB currently. In Section 3.5, we derive these oracles for K_p (noting that they can be adapted easily for K_h).

3.5 Barrier oracles for epigraph-perspective cones

We consider the epigraph-perspective cone K_p defined in (3.30). Recall that we let $p = (p, q, r)$ 2 R × R × V and $x = (u, v, w)$ 2 int(K_p), and we define ζ and : int(K_p) ! R from (3.37) as:

$$\zeta(x) := u - v\varphi(v^{-1}w), \tag{3.39a}$$

$$(x) := - \log(\zeta(x)) - \log(v) - \operatorname{logdet}(w). \tag{3.39b}$$

In this section, we derive expressions and evaluation procedures for the g, H, T, and H oracles (defined in Section 3.4.4) corresponding to the LHB for K_p. We note that the oracles for K_h in (3.29) 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 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 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 $\mu : \mathbb{R}_{>} \times \operatorname{int}(Q) \to \operatorname{int}(Q)$ and its first directional derivative $\xi \in V$ in the direction (q, r) as:

$$\mu(v, w) := v^{-1}w, \tag{3.40a}$$

$$\xi := \mathrm{D}\mu(v, w)[(q, r)] = \mathrm{D}_v\mu(v, w)q + \mathrm{D}_w\mu(v, w)[r] = v^{-1}(r - q\mu(v, w)). \tag{3.40b}$$

For convenience, we fix the constants $\mu := \mu(v, w)$, $\varphi := \varphi(\mu)$, and $\zeta := \zeta(\varpi)$. Let $\mathrm{D}\varphi$, $\mathrm{D}^2\varphi$, and $\mathrm{D}^3\varphi$ be the derivatives of φ evaluated at μ, and let $\mathrm{D}\zeta$, $\mathrm{D}^2\zeta$, and $\mathrm{D}^3\zeta$ be the derivatives of ζ evaluated at ϖ. Using (3.40), the directional derivatives of ζ

can be written compactly as:

$$r_u\zeta = 1, \tag{3.41a}$$

$$r_v\zeta = -\varphi + r\,\varphi[\mu], \tag{3.41b}$$

$$r_w\zeta = -r\,\varphi, \tag{3.41c}$$

$$r\,\zeta[p] = p - q\varphi - v\,r\,\varphi[\xi], \tag{3.41d}$$

$$(r^2\zeta[p])_v = r^2\varphi[\xi,\mu], \tag{3.41e}$$

$$(r^2\zeta[p])_w = -r^2\varphi[\xi], \tag{3.41f}$$

$$r^2\zeta[p,p] = -v\,r^2\varphi[\xi,\xi], \tag{3.41g}$$

$$(r^3\zeta[p,p])_v = r^3\varphi[\xi,\xi,\mu] + r^2\varphi[\xi,\xi] - 2v^{-1}q\,r^2\varphi[\xi,\mu], \tag{3.41h}$$

$$(r^3\zeta[p,p])_w = 2v^{-1}q\,r^2\varphi[\xi] - r^3\varphi[\xi,\xi], \tag{3.41i}$$

$$r^3\zeta[p,p,p] = -v\,r^3\varphi[\xi,\xi,\xi] + 3q\,r^2\varphi[\xi,\xi]. \tag{3.41j}$$

Using (3.41), we now derive the directional derivatives of . For convenience, we let r , r^2 , r^3 be the derivatives of evaluated at π. We define:

$$\sigma := -r_v\zeta = \varphi - r\,\varphi[\mu] \; 2 \; \mathrm{R}. \tag{3.42}$$

The components of the gradient g of are:

$$g_u = -\zeta^{-1}, \tag{3.43a}$$

$$g_v = \zeta^{-1}\sigma - v^{-1}, \tag{3.43b}$$

$$g_w \overset{3.26}{=} \zeta^{-1}\,r\,\varphi - w^{-1}. \tag{3.43c}$$

Differentiating (3.43), the Hessian components are:

$$r^2_{u;u} = \zeta^{-2} > 0, \tag{3.44a}$$

$$r^2_{v;u} = -\zeta^{-2}\sigma \; 2 \; \mathsf{R}, \tag{3.44b}$$

$$r^2_{w;u} = -\zeta^{-2} \, r\varphi \; 2 \; V, \tag{3.44c}$$

$$r^2_{v;v} = v^{-2} + \zeta^{-2}\sigma^2 + v^{-1}\zeta^{-1} \, r^2\varphi[\mu,\mu] > 0, \tag{3.44d}$$

$$r^2_{w;v} = \zeta^{-2}\sigma \, r\varphi - v^{-1}\zeta^{-1} \, r^2\varphi[\mu] \; 2 \; V. \tag{3.44e}$$

Differentiating (3.43c) in the direction r:

$$r^2_{w;w}\;[r] \overset{3.24}{=} \zeta^{-2} \, r\varphi[r] \, r\varphi + v^{-1}\zeta^{-1} \, r^2\varphi[r] + P(w^{-1})r \; 2 \; V. \tag{3.45}$$

Let:

$$\chi := \zeta^{-1}(p - q\sigma - \; r\varphi[r]) \; 2 \; \mathsf{R}. \tag{3.46}$$

The components of the Hessian product H are:

$$H_u = \zeta^{-1}\chi, \tag{3.47a}$$

$$H_v = -\zeta^{-1}\sigma\chi - \zeta^{-1} \, r^2\varphi[\xi,\mu] + v^{-2}q, \tag{3.47b}$$

$$H_w = -\zeta^{-1}\chi \, r\varphi + \zeta^{-1} \, r^2\varphi[\xi] + P(w^{-1})r. \tag{3.47c}$$

Let:

$$\kappa := 2\zeta^{-1}(\chi + v^{-1}q) \, r^2\varphi[\xi] - \zeta^{-1} \, r^3\varphi[\xi,\xi] \; 2 \; V. \tag{3.48}$$

The components of the third order directional derivative T are:

$$T_u = -2\zeta^{-1}\chi^2 - v\zeta^{-2} \, r^2\varphi[\xi,\xi], \tag{3.49a}$$

$$T_v = -T_u\sigma + \hbar\kappa,\mu i - \zeta^{-1} \, r^2\varphi[\xi,\xi] - 2q^2v^{-3}, \tag{3.49b}$$

$$T_w \overset{3.25}{=} -T_u \, r\varphi - \kappa - 2P(w^{-1=2})(P(w^{-1=2})r)^2. \tag{3.49c}$$

3.5.2 Inverse Hessian operator

The Hessian of at any point $\pi \in \text{int}(K_p)$ is a positive definite linear operator and hence invertible. By treating the components of the Hessian in (3.44) and (3.45) analogously to blocks of a positive definite matrix, we derive the inverse operator. For convenience, we let:

$$Y_u := (r^2_{w;w})^{-1} r^2_{w;u}, \tag{3.50a}$$

$$Y_v := (r^2_{w;w})^{-1} r^2_{w;v}, \tag{3.50b}$$

$$Z_{u;u} := r^2_{u;u} - h r^2_{w;u}, Y_u i, \tag{3.50c}$$

$$Z_{v;u} := r^2_{v;u} - h r^2_{w;u}, Y_v i, \tag{3.50d}$$

$$Z_{v;v} := r^2_{v;v} - h r^2_{w;v}, Y_v i. \tag{3.50e}$$

Note $Y_u, Y_v \in V$. We let Z be:

$$Z := \begin{bmatrix} Z_{u;u} & Z_{v;u} \\ Z_{v;u} & Z_{v;v} \end{bmatrix} \in \mathbb{S}^2, \tag{3.51}$$

and its inverse is:

$$Z := Z^{-1} = \frac{1}{Z_{u;u}Z_{v;v} - Z^2_{v;u}} \begin{bmatrix} Z_{v;v} & -Z_{v;u} \\ -Z_{v;u} & Z_{u;u} \end{bmatrix} \in \mathbb{S}^2. \tag{3.52}$$

It can be verified (for example, by analogy to the block symmetric matrix inverse formula) that the inverse Hessian product oracle H in (3.38c) is:

$$H_u = Z_{u;u}(p - hY_u, r i) + Z_{v;u}(q - hY_v, r i), \tag{3.53a}$$

$$H_v = Z_{v;u}(p - hY_u, r i) + Z_{v;v}(q - hY_v, r i), \tag{3.53b}$$

$$H_w = -H_u Y_u - H_v Y_v + (r^2_{w;w})^{-1} r. \tag{3.53c}$$

Hence computing H is essentially only as difficult as applying the positive definite linear operator $(r^2_{w;w})^{-1}$. We are not aware of a simple expression for $(r^2_{w;w})^{-1}$ in gen-

eral, 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 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 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:

$$m_{i;j} := \begin{cases} \zeta^{-1} \frac{(\nabla h)_i - (\nabla h)_j}{w_i - w_j} + w_i^{-1} w_j^{-1} & i \neq j, \\ \zeta^{-1} \nu^{-1} (\nabla^2 h)_i + w_i^{-2} & i = j. \end{cases} \tag{3.54}$$

Since h is convex, $m_{i;j} > 0, \forall i, j \in [d]$. Let $M : V \to V$ be the self-adjoint linear operator:

$$M := \nu^{-1} \zeta^{-1} \nabla^2 \varphi + P(w^{-1}) = \sum_{i;j \in [d] : i < j} 4 m_{i;j} L(c_i) L(c_j) + \sum_{i \in [d]} m_{i;i} P(c_i). \tag{3.55}$$

Using (3.12), we have the self-adjoint inverse operator of M:

$$M^{-1} = \sum_{i;j \in [d] : i < j} 4 m_{i;j}^{-1} L(c_i) L(c_j) + \sum_{i \in [d]} m_{i;i}^{-1} P(c_i). \tag{3.56}$$

Substituting (3.55) into (3.45), we have:

$$\nabla^2_{w;w} [r] = \zeta^{-2} \nabla \varphi[r] \nabla \varphi + Mr. \tag{3.57}$$

Note that the first term in (3.57) is analogous to the application (to r) of a low-rank update to M, and that M^{-1} in (3.56) 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 $(\nabla^2_{w;w})^{-1}r$.

We let:

$$\alpha := M^{-1}\nabla\varphi = \sum_{i\in[\![d]\!]} m_{i;i}^{-1}(\nabla h)_i c_i \in V, \tag{3.58a}$$

$$\gamma := v^{-2}\zeta^{-1}M^{-1}\nabla^2\varphi[w] = v^{-2}\zeta^{-1}\sum_{i\in[\![d]\!]} m_{i;i}^{-1}(\nabla^2 h)_i w_i c_i \neq 0. \tag{3.58b}$$

Noting that $\gamma, w^{-1} \neq 0$ implies $\langle\gamma, w^{-1}\rangle > 0$, we define the scalar constants:

$$k_1 := \zeta^2 + \langle\nabla\varphi, \alpha\rangle > 0, \tag{3.59a}$$

$$k_2 := \sigma + \langle\nabla\varphi, \gamma\rangle = \sigma + v^{-2}\zeta^{-1}\langle\nabla^2\varphi[w], \alpha\rangle, \tag{3.59b}$$

$$k_3 := v^{-2} + v^{-2}\zeta^{-1}\langle\nabla^2\varphi[w], \mu - \gamma\rangle = v^{-2} + v^{-1}\langle\gamma, w^{-1}\rangle > 0. \tag{3.59c}$$

Now using the Sherman-Morrison-Woodbury formula:

$$(\nabla^2_{w;w})^{-1}r = M^{-1}r - \frac{\zeta^{-2}\langle M^{-1}\nabla\varphi, r\rangle}{1 + \zeta^{-2}\langle M^{-1}\nabla\varphi,\ \nabla\varphi\rangle}M^{-1}\nabla\varphi \tag{3.60a}$$

$$= M^{-1}r - k_1^{-1}\langle\alpha, r\rangle\alpha. \tag{3.60b}$$

Substituting (3.44) and (3.60) into (3.50):

$$Y_u = (r^2_{w;w})^{-1}(-\zeta^{-2} r\varphi) \tag{3.61a}$$
$$= -\zeta^{-2}\alpha + \zeta^{-2}k_1^{-1}h\alpha,\ r\varphi i\alpha \tag{3.61b}$$
$$= -k_1^{-1}\alpha, \tag{3.61c}$$
$$Y_v = (r^2_{w;w})^{-1}(\zeta^{-2}\sigma r\varphi - v^{-1}\zeta^{-1} r^2\varphi[\mu]) \tag{3.61d}$$
$$= -\sigma Y_u - v^{-2}\zeta^{-1}(r^2_{w;w})^{-1} r^2\varphi[w] \tag{3.61e}$$
$$= \sigma k_1^{-1}\alpha - \gamma + v^{-2}\zeta^{-1}k_1^{-1}h\alpha,\ r^2\varphi[w]i\alpha \tag{3.61f}$$
$$= k_1^{-1}k_2\alpha - \gamma, \tag{3.61g}$$
$$Z_{u;u} = \zeta^{-2} - h\, r^2_{w;u}\ , Y_u i \tag{3.61h}$$
$$= \zeta^{-2} - \zeta^{-2}k_1^{-1}h\, r\varphi, \alpha i \tag{3.61i}$$
$$= k_1^{-1}, \tag{3.61j}$$
$$Z_{v;u} = -\zeta^{-2}\sigma - h\, r^2_{w;u}\ , Y_v i \tag{3.61k}$$
$$= -\zeta^{-2}(\sigma - h\, r\varphi, k_1^{-1}k_2\alpha - \gamma i) \tag{3.61l}$$
$$= -\zeta^{-2}(\sigma - k_1^{-1}k_2(k_1 - \zeta^2) + k_2 - \sigma) \tag{3.61m}$$
$$= -k_1^{-1}k_2, \tag{3.61n}$$
$$Z_{v;v} = r^2_{v;v} - h\, r^2_{w;v}\ , Y_v i \tag{3.61o}$$
$$= r^2_{v;v} + \sigma h\, r^2_{w;u}\ , Y_v i + v^{-2}\zeta^{-1}h\, r^2\varphi[w], Y_v i \tag{3.61p}$$
$$\overset{3.61n}{=} r^2_{v;v} + \sigma(k_1^{-1}k_2 - \zeta^{-2}\sigma) + v^{-2}\zeta^{-1}h\, r^2\varphi[w], Y_v i \tag{3.61q}$$
$$= v^{-2} + v^{-3}\zeta^{-1} r^2\varphi[w, w] + \sigma k_1^{-1}k_2 + v^{-2}\zeta^{-1}h\, r^2\varphi[w], Y_v i \tag{3.61r}$$
$$= v^{-2} + v^{-3}\zeta^{-1} r^2\varphi[w, w] + \sigma k_1^{-1}k_2 + k_1^{-1}k_2(k_2 - \sigma) - v^{-2}\zeta^{-1}h\, r^2\varphi[w], \gamma i \tag{3.61s}$$
$$= v^{-2} + k_1^{-1}k_2^2 + v^{-2}\zeta^{-1}h\, r^2\varphi[w], \mu - \gamma i \tag{3.61t}$$
$$= k_3 + k_1^{-1}k_2^2. \tag{3.61u}$$

For Z in (3.51), we have $\det(Z) = k_1^{-1}k_3$, so its inverse Z in (3.52) is:

$$Z_{u;u} = k_1(k_3 + k_1^{-1}k_2^2)k_3^{-1} = k_1 + k_2^2k_3^{-1}, \tag{3.62a}$$

$$Z_{u;v} = k_2k_3^{-1}, \tag{3.62b}$$

$$Z_{v;v} = k_3^{-1}. \tag{3.62c}$$

Finally, we substitute (3.61) and (3.62) into (3.53) to derive the inverse Hessian product H. We let:

$$c_1 := p - hY_u, ri \overset{3.61}{=} p + k_1^{-1}h\alpha, ri, \tag{3.63a}$$

$$c_2 := q - hY_v, ri \overset{3.61}{=} q - k_1^{-1}k_2h\alpha, ri + h\gamma, ri. \tag{3.63b}$$

For convenience, we derive H_v before H_u and H_w:

$$H_v = Z_{u;v}c_1 + Z_{v;v}c_2 \tag{3.64a}$$

$$= k_3^{-1}(k_2c_1 + c_2) \tag{3.64b}$$

$$= k_3^{-1}(k_2p + q + h\gamma, ri), \tag{3.64c}$$

$$H_u = Z_{u;u}c_1 + Z_{u;v}c_2 \tag{3.64d}$$

$$\overset{3.64b}{=} (Z_{u;u} - Z_{u;v}k_2)c_1 + Z_{u;v}k_3H_v \tag{3.64e}$$

$$= k_1p + k_2H_v + h\alpha, ri, \tag{3.64f}$$

$$H_w = -H_uY_u - H_vY_v + (r^2_{w;w})^{-1}r \tag{3.64g}$$

$$\overset{3.60}{=} H_uk_1^{-1}\alpha - H_v(k_1^{-1}k_2\alpha - \gamma) + M^{-1}r - k_1^{-1}h\alpha, ri\alpha \tag{3.64h}$$

$$= p\alpha + H_v\gamma + M^{-1}r. \tag{3.64i}$$

In Section 3.8.5, we compare the efficiency and numerical performance of the closed-form formula for H in (3.64) against a naive approach to computing 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-

e□cient, 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 $\varphi(w) = -\operatorname{logdet}(w) = -\sum_{i=1}^{n} \log(w_i)$. In Section 3.6, we show that is an LHSCB in this case. We let:

$$\hat{\xi} := P(w^{-1/2})\xi = v^{-1}P(w^{-1/2})(-v^{-1}qw + r) = v^{-1}(-v^{-1}qe + \hat{r}) \in V. \tag{3.65}$$

We have:

$$r\varphi \overset{3.26}{=} -\mu^{-1} = -vP(w^{-1/2})e, \tag{3.66a}$$

$$r^2\varphi[\xi] \overset{3.27a}{=} v^2P(w^{-1})\xi = v^2P(w^{-1/2})\hat{\xi}, \tag{3.66b}$$

$$r^3\varphi[\xi,\xi] \overset{3.27b}{=} -2v^3P(w^{-1/2})\hat{\xi}^2. \tag{3.66c}$$

The constants from Section 3.5.1 have the form:

$$\sigma \overset{3.42}{=} \varphi + d, \tag{3.67a}$$

$$\chi \overset{3.46}{=} \zeta^{-1}(p - q\sigma + v\operatorname{tr}(\hat{r})), \tag{3.67b}$$

From (3.43), the w component of the gradient is:

$$g_w = -(1 + v\zeta^{-1})w^{-1}. \tag{3.68a}$$

From (3.47), the v and w components of the Hessian product are:

$$H_v = -\zeta^{-1}\sigma\chi - v\zeta^{-1}\operatorname{tr}(\hat{\xi}) + v^{-2}q, \tag{3.69a}$$

$$H_w = P(w^{-1/2})(v\zeta^{-1}\chi e + v^2\zeta^{-1}\hat{\xi} + \hat{r}). \tag{3.69b}$$

From (3.49), the third order directional derivative is:

$$T_u = -2\zeta^{-1}\chi^2 - v^3\zeta^{-2}\,\mathrm{tr}(\hat{\xi}^2), \tag{3.70a}$$

$$T_v = -T_u\sigma + v\zeta^{-1}(2(\chi + v^{-1}q)\,\mathrm{tr}(\hat{\xi}) + v\,\mathrm{tr}(\hat{\xi}^2)) - 2v^{-3}q^2, \tag{3.70b}$$

$$T_w = P(w^{-1=2})(T_u ve - 2\zeta^{-1}v^2((\chi + v^{-1}q)\hat{\xi} + v\hat{\xi}^2) - 2\hat{r}^2). \tag{3.70c}$$

We derive the inverse Hessian product H by specializing the separable case in (3.64). We let:

$$r := P(w^{1=2})r \; 2 \; V, \tag{3.71a}$$

$$\theta := v^2(\zeta + (1 + d)v)^{-1}. \tag{3.71b}$$

From (3.58) and (3.59), we have:

$$M^{-1} = \zeta(\zeta + v)^{-1}P(w), \tag{3.72a}$$

$$\alpha = -v\zeta(\zeta + v)^{-1}w, \tag{3.72b}$$

$$\gamma = (\zeta + v)^{-1}w, \tag{3.72c}$$

$$k_1 = \zeta^2 + dv^2\zeta(\zeta + v)^{-1}, \tag{3.72d}$$

$$k_2 = \varphi + d\zeta(\zeta + v)^{-1}, \tag{3.72e}$$

$$k_3^{-1} = (\zeta + v)\theta. \tag{3.72f}$$

For convenience, we derive H_v before H_u and H_w as in (3.64):

$$H_v = k_3^{-1}(k_2 p + q + h\gamma, r i) \tag{3.73a}$$
$$= (\zeta + v)\theta((\varphi + d\zeta(\zeta + v)^{-1})p + q + hr, w i(\zeta + v)^{-1}) \tag{3.73b}$$
$$= \theta((\zeta + v)(\varphi p + q) + d\zeta p + \mathsf{tr}(r)), \tag{3.73c}$$
$$H_u = k_1 p + (\varphi + d\zeta(\zeta + v)^{-1})H_v + h\alpha, r i \tag{3.73d}$$
$$= (\zeta^2 + dv^2\zeta(\zeta + v)^{-1})p + (\varphi + d\zeta(\zeta + v)^{-1})H_v + h - v\zeta(\zeta + v)^{-1}w, r i \tag{3.73e}$$
$$= \zeta(\zeta + v)^{-1}(dv^2 p + dH_v - vhw, r i) + \zeta^2 p + \varphi H_v, \tag{3.73f}$$
$$H_w = p\alpha + H_v\gamma + M^{-1}r \tag{3.73g}$$
$$= -pv\zeta(\zeta + v)^{-1}w + H_v(\zeta + v)^{-1}w + \zeta(\zeta + v)^{-1}P(w)r \tag{3.73h}$$
$$= (\zeta + v)^{-1}P(w^{1=2})((-\zeta pv + H_v)e + \zeta r). \tag{3.73i}$$

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 $V = \mathsf{S}^d$ and $V = \mathsf{H}^d$, 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 K_{MMD} in Section 3.6.2. K_{MMD} is a special case of the epigraph-perspective cone K_p with a separable spectral function φ. In Section 3.6.4, we prove that our barrier function for K_{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:

$$f(x) = \alpha + \beta x + \int_0^1 \frac{x}{x+t} \, \mathrm{d}\rho(t) = \alpha + \beta x + \int_0^1 \left(1 - \frac{t}{x+t}\right) \mathrm{d}\rho(t), \tag{3.74}$$

where $\alpha \in \mathbb{R}, \beta \in \mathbb{R}_{\geq}$ and ρ is a positive measure on $\mathbb{R}_{>}$ such that $\int_0^1 (1+t)^{-1} \, \mathrm{d}\rho(t) < \infty$.

This result implies that, for a cone of squares Q of a Jordan algebra, and for $w \in \mathrm{int}(Q)$ with the spectral decomposition $w = \sum_{i \in \mathbb{J}d\mathbb{K}} w_i c_i$, we have:

$$\begin{aligned} f(w) &= \sum_{i \in \mathbb{J}d\mathbb{K}} f(w_i) c_i && \text{(3.75a)} \\ &= \alpha e + \beta w + \int_0^1 \sum_{i \in \mathbb{J}d\mathbb{K}} \left(1 - \frac{t}{w_i + t}\right) c_i \, \mathrm{d}\rho(t) && \text{(3.75b)} \\ &= \alpha e + \beta w + \int_0^1 (e - t(w + te)^{-1}) \, \mathrm{d}\rho(t). && \text{(3.75c)} \end{aligned}$$

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 (1.4)), 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 $x \mapsto x^p$ for $p \in [0,1]$ are matrix monotone. This implies that in Table 3.1, each function in the $r\,h$ 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], Furuta [2008].

MMD function	h	$r\,h$	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}	$\mathbb{R}$	$(p-1)(x_-/p)^q$

Table 3.1: Examples of MMD functions. We let $q := p/(p-1)$, which gives $q \in (-1,0)$ for $p \in (0,1)$ in the NegPower case, or $q \in [2,-1)$ for $p \in (1,2]$ in the Power case. We also let $x_- := \max(-x, 0)$ in the Power case.

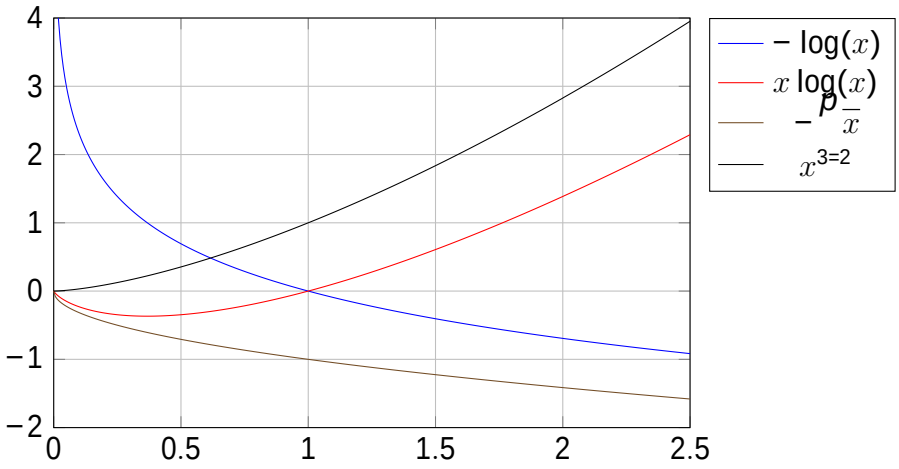

Figure 3-1: Plots of example MMD functions.

Suppose Q is the cone of squares of a Jordan algebra V with rank d. Let $\varphi : \text{int}(Q) \to \mathbb{R}$ be the C^3-smooth function $\varphi(w) = \text{tr}(h(w)) = \sum_{i \in [d]} h(w_i)$, which is a convex separable spectral function (see Section 3.3.2). As in Section 3.4.2, we let $\mathfrak{u} = (u, v, w) \in E = \mathbb{R} \times \mathbb{R}_> \times \text{int}(Q)$. The function $\zeta : E \to \mathbb{R}$ has the form:

$$\zeta(\mathfrak{u}) := u - v\,\text{tr}(h(v^{-1}w)). \tag{3.76}$$

We define the matrix monotone derivative cone K_{MMD}, a special case of the epigraph-perspective cone K_p in (3.30), as:

$$K_{\mathrm{MMD}} := \mathrm{cl}\{(u, v, w) \in E : u \geq v\,\mathrm{tr}(h(v^{-1}w))\}, \tag{3.77}$$

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 K_{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 (1.4)) of φ is $\varphi^*(r) = \mathrm{tr}(h^*(r))$. So from (3.31), the dual cone is:

$$K^*_{\mathrm{MMD}} := \mathrm{cl}\{(u, v, w) \in \mathbb{R}_{>} \times \mathbb{R} \times V : v \geq u\,\mathrm{tr}(h^*(u^{-1}w))\}. \tag{3.78}$$

3.6.3 Derivatives of the MMD trace

Suppose $w \succ 0$. Since $\varphi(w) = \mathrm{tr}(h(w))$ and $-h$ is matrix monotone, from (3.75) we can write the gradient:

$$-\nabla\varphi(w) = -\nabla h(w) = \alpha e + \beta w + \int_0^1 (e - t(w + te)^{-1})\,\mathrm{d}\rho(t). \tag{3.79}$$

Note that $w + te \succ 0$ for $t \geq 0$, so $(w + te)^{-1}$ is well-defined. Differentiating (3.79) in the direction $r \in V$, we have the second order directional derivative:

$$-\nabla^2\varphi(w)[r] \overset{(3.24)}{=} \beta r + \int_0^1 tP((w + te)^{-1})r\,\mathrm{d}\rho(t), \tag{3.80}$$

and the third order directional derivative:

$$-\nabla^3\varphi(w)[r, r] \overset{(3.25)}{=} -2\int_0^1 tP((w + te)^{-1/2})(P((w + te)^{-1/2})r)^2\,\mathrm{d}\rho(t). \tag{3.81}$$

3.6.4 Self-concordant barrier

For K_{MMD}, the LHB $\psi : \mathrm{int}(K_{\mathrm{MMD}}) \to \mathbb{R}$ from (3.37) has the form:

$$\psi(u, v, w) := -\log(\zeta(u, v, w)) - \log(v) - \mathrm{logdet}(w). \tag{3.82}$$

We describe easily-computable oracles for this in Section 3.5, including an inverse Hessian product 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 S:

$$S := \mathrm{cl}\{(u, w) \in \mathbb{R} \times \mathrm{int}(Q) : u - \varphi(w) \geq 0\}. \tag{3.83}$$

K_{MMD} is the conic hull of S. In Theorem 3.6.1, we prove that our barrier in (3.82) is self-concordant, hence it is an LHSCB for K_{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 Nemirovskii [1994, Proposition 5.1.4] yields the parameter $800(1 + d)$). Since the optimal barrier parameter for $\mathrm{cl}(E)$ is $1 + d$, our parameter cannot be reduced by more than one.

Theorem 3.6.1. *in* (3.82) *is a* $(2 + d)$*-LHSCB for* K_{MMD} *in* (3.77).

Proof. We show that ζ in (3.76) is $(\mathbb{R}_{\geq}, 1)$-compatible with the domain E, in the sense of Nesterov and Nemirovskii [1994, Definition 5.1.1]. This follows if (i) ζ is C^3-smooth on E, (ii) ζ is concave with respect to $\mathbb{R}_{\geq}$, (iii) for any point $\mathbb{u} \in \mathrm{int}(K_{\mathrm{MMD}})$ and direction $\mathbb{p} = (p, q, r) \in \mathbb{R} \times \mathbb{R} \times V$ satisfying $v \pm q \geq 0$ and $w \pm r \neq 0$ it holds that:

$$\mathrm{D}^3\zeta(\mathbb{u})[\mathbb{p}, \mathbb{p}, \mathbb{p}] \leq -3\,\mathrm{D}^2\zeta(\mathbb{u})[\mathbb{p}, \mathbb{p}]. \tag{3.84}$$

Suppose $v \pm q \geq 0$ and $w \pm r \neq 0$. As in (3.40), we let $\mu := \mu(v, w) = v^{-1}w \neq 0$ and $\xi := v^{-1}(r - q\mu)$. From (3.41), the second and third order directional derivatives of ζ at $\mathbb{u}$ in direction $\mathbb{p}$ are:

$$\mathrm{D}^2\zeta(\mathbb{u})[\mathbb{p}, \mathbb{p}] = -v\,\mathrm{D}^2\varphi(\mu)[\xi, \xi], \tag{3.85a}$$

$$\mathrm{D}^3\zeta(\mathbb{u})[\mathbb{p}, \mathbb{p}, \mathbb{p}] = -v\,\mathrm{D}^3\varphi(\mu)[\xi, \xi, \xi] + 3q\,\mathrm{D}^2\varphi(\mu)[\xi, \xi]. \tag{3.85b}$$

Since φ is convex and C^3-smooth on $\mathrm{int}(Q)$ by assumption, (3.76) and (3.85) imply

that ζ is concave and C^3-smooth on E. It remains to show that (3.84) holds.

For $t \geq 0$, let:

$$a(t) := \mu + te \quad 0, \tag{3.86a}$$

$$a(t) := a(t)^{-1=2} \quad 0, \tag{3.86b}$$

$$\hat{\xi}(t) := P(a(t))\xi. \tag{3.86c}$$

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:

$$r^2\varphi(\mu)[\xi,\xi] \overset{3.80}{=} \beta \operatorname{tr}(\xi^2) + \int_0^1 t \operatorname{tr}(\hat{\xi}(t)^2)\ \mathrm{d}\rho(t), \tag{3.87a}$$

$$r^3\varphi(\mu)[\xi,\xi,\xi] \overset{3.81}{=} -2\int_0^1 t \operatorname{tr}(\hat{\xi}(t)^3)\ \mathrm{d}\rho(t). \tag{3.87b}$$

From (3.85) and (3.87), the compatibility condition (3.84) is equivalent to nonnegativity of:

$$-3r^2\zeta(\mathfrak{u})[\mathfrak{p},\mathfrak{p}] - r^3\zeta(\mathfrak{u})[\mathfrak{p},\mathfrak{p},\mathfrak{p}] \tag{3.88a}$$

$$= 3(v-q)\beta \operatorname{tr}(\xi^2) + \int_0^1 t(3(v-q)\operatorname{tr}(\hat{\xi}(t)^2) - 2v\operatorname{tr}(\hat{\xi}(t)^3))\ \mathrm{d}\rho(t). \tag{3.88b}$$

Since $v \geq q$, the first term in (3.88b) is nonnegative. The second term (the integral) is nonnegative if for all $t \geq 0$, the following inner term is nonnegative:

$$3(v-q)\operatorname{tr}(\hat{\xi}(t)^2) - 2v\operatorname{tr}(\hat{\xi}(t)^3) = \langle\hat{\xi}(t)^2, 3(v-q)e - 2v\hat{\xi}(t)\rangle. \tag{3.89}$$

By self-duality of Q, (3.89) is nonnegative if $3(v-q)e - 2v\hat{\xi}(t) \quad 0$, which we now prove. For $t \geq 0$, let:

$$b(t) := (1 - v^{-1}q)a(t) - \xi = v^{-1}(w-r) + t(1-v^{-1}q)e. \tag{3.90}$$

Since $w \succeq r$ and $1 - v^{-1}q \geq 0$, we have $b(t) \succeq 0$. Hence we have:

$$(1 - v^{-1}q)e - \hat{\xi}(t) \overset{3.90}{=} P(a(t))b(t) \succeq 0, \tag{3.91}$$

since $a(t) \succ 0$ implies $P(a(t))$ is an automorphism on Q (see Faraut and Koranyi [1998, Page 48]). Therefore:

$$3(v - q)e - 2v\hat{\xi}(t) \overset{3.91}{\succeq} 3(v - q)e - 2v(1 - v^{-1}q)e = (v - q)e \succeq 0. \tag{3.92}$$

So (3.89) is nonnegative, which implies the integral term in (3.88b) is nonnegative.

Thus (3.88b) is nonnegative, so (3.84) holds and compatibility is proved. Now by Nesterov and Nemirovskii [1994, Proposition 5.1.7], $\barf$ is an LHSCB for K_{MMD} with parameter $2 + d$. □

3.7 Root-determinant cones

In Section 3.7.1, we define the root-determinant cone K_{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 for K_{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 Q is a cone of squares of a Jordan algebra V with rank d. Let $\varphi : Q \to \mathbb{R}_+$ denote the root-determinant function (or the geometric mean of the eigenvalues):

$$\varphi(w) := \det(w)^{1/d} = \prod_{i \in J d K} w_i^{1/d}, \tag{3.93}$$

which is a concave, homogeneous nonseparable spectral function (see Section 3.3.1). We let $\mathbb{u} := (u, w) \in E = \mathbb{R} \times Q$. The function $\zeta : E \to \mathbb{R}$ has the form:

$$\zeta(\mathbb{u}) := \varphi(w) - u. \tag{3.94}$$

We define the root-determinant cone K_{rtdet} and its dual cone as:

$$K_{\text{rtdet}} := \{(u, w) \in \mathbb{R} \times Q : u \leq \det(w)^{1/d}\}, \tag{3.95a}$$

$$K^*_{\text{rtdet}} := \{(u, w) \in \mathbb{R}_{\leq} \times Q : -d^{-1}u \leq \det(w)^{1/d}\}. \tag{3.95b}$$

We note that K_{rtdet} is a hypograph modification of the epigraph cone K_h in (3.29), and it is a primitive proper cone. K^*_{rtdet} can be derived by modifying the steps we use to derive K^*_h in (3.35) 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 (3.26) gives us the gradient:

$$\nabla\varphi(w) = d^{-1}\varphi(w)w^{-1}. \tag{3.96}$$

Let $r \in V$ and $\mathbb{r} := P(w^{-1/2})r \in V$. Using the product rule on (3.96), we have the second order directional derivative:

$$\nabla^2\varphi(w)[r] = d^{-2}\langle\mathbb{r}w^{-1}, r\rangle\nabla\varphi(w) + d^{-1}\varphi(w)\nabla_w(w^{-1})[r] \tag{3.97a}$$

$$\overset{3.24}{=} d^{-1}\varphi(w)\operatorname{tr}(\mathbb{r})w^{-1} - d^{-1}\varphi(w)P(w^{-1})r \tag{3.97b}$$

$$= d^{-1}\varphi(w)(d^{-1}\operatorname{tr}(\mathbb{r})w^{-1} - P(w^{-1})r) \tag{3.97c}$$

$$= d^{-1}\varphi(w)P(w^{-1/2})(d^{-1}\operatorname{tr}(\mathbb{r})e - \mathbb{r}). \tag{3.97d}$$

Finally, using the product rule on (3.97c), we have the third order directional derivative:

$$\begin{aligned} r^3\varphi(w)[r,r] &= d^{-1}\hbar d^{-1}\operatorname{tr}(\hat{r})w^{-1} - P(w^{-1})r, r\,i\,r\,\varphi(w) + d^{-1}\varphi(w)(\\ &\quad d^{-1}(\hbar w^{-1}, r\,i\,r_w(\hbar w^{-1}, r\,i) + \operatorname{tr}(\hat{r})\,r_w(w^{-1})[r]) - r_w(P(w^{-1})r)[r]) \end{aligned} \tag{3.98a}$$

$$\begin{aligned} &\overset{3.25}{=} d^{-2}\varphi(w)(d^{-1}\operatorname{tr}(\hat{r})^2 - \operatorname{tr}(\hat{r}^2))w^{-1} + d^{-1}\varphi(w)(\\ &\quad - 2d^{-1}\operatorname{tr}(\hat{r})P(w^{-1})r + 2P(w^{-1=2})(P(w^{-1=2})r)^2) \end{aligned} \tag{3.98b}$$

$$= d^{-1}\varphi(w)P(w^{-1=2})(d^{-1}(d^{-1}\operatorname{tr}(\hat{r})^2 - \operatorname{tr}(\hat{r}^2))e - 2d^{-1}\operatorname{tr}(\hat{r})\hat{r} + 2\hat{r}^2). \tag{3.98c}$$

3.7.3 Self-concordant barrier

For K_{rtdet}, the LHB $\quad : \operatorname{int}(K_{\text{rtdet}}) \; ! \; \mathbb{R}$ from (3.37) has the form:

$$(\varpi) := -\log(\zeta(\varpi)) - \operatorname{logdet}(w). \tag{3.99}$$

In Theorem 3.7.1 we show that $\quad$ is self-concordant with parameter $1 + d$. Since the optimal barrier parameter for E is d, our parameter cannot be reduced by more than one.

Theorem 3.7.1. *in* (3.99) *is a* $(1 + d)$*-LHSCB for* K_{rtdet} *in* (3.95a)*.*

Proof. Note $\quad(\varpi) := -\operatorname{logdet}(w)$ is a d-LHSCB for E. We show that ζ in (3.94) is $(\mathbb{R}_{\geq}, 1)$-compatible with the barrier $\quad$ in the sense of Nesterov and Nemirovskii [1994, Definition 5.1.2]. Compatibility follows if (i) ζ is C^3-smooth on $\operatorname{int}(E)$, (ii) concave with respect to $\mathbb{R}_{\geq}$, (iii) for any point $\varpi \in \operatorname{int}(K_{\text{rtdet}})$ and direction $\mathfrak{p} = (p, r) \in \mathbb{R} \times V$ it holds that:

$$r^3\zeta(\varpi)[\mathfrak{p},\mathfrak{p},\mathfrak{p}] \leq -3(r^2 \quad (\varpi)[\mathfrak{p},\mathfrak{p}])^{1=2}\, r^2\zeta(\varpi)[\mathfrak{p},\mathfrak{p}]. \tag{3.100}$$

Suppose $\varpi \in \text{int}(K_{\text{rtdet}})$. From (3.94), we have:

$$r^2\zeta(\varpi)[\bar{p},\bar{p}] = r^2\varphi(w)[r,r], \tag{3.101a}$$

$$r^3\zeta(\varpi)[\bar{p},\bar{p},\bar{p}] = r^3\varphi(w)[r,r,r]. \tag{3.101b}$$

Since φ is concave and C^3-smooth on int(Q), (3.101) implies ζ is concave and C^3-smooth on int(E). It remains to show that (3.100) holds.

Let $\sigma \in \mathbb{R}^d$ be the eigenvalues of $\hat{r} := P(w^{-1/2})r$. Then:

$$(r^2\zeta(\varpi)[\bar{p},\bar{p}])^{1/2} \overset{3.27a}{=} \text{tr}(\hat{r}^2)^{1/2} = k\sigma k. \tag{3.102}$$

Let $m_k := d^{-1}\,\text{tr}(\hat{r}^k), \forall k \in [3]$, and let $\delta_i := \sigma_i - m_1, \forall i \in [d]$. By the formulae for variance and skewness, we have:

$$m_2 - m_1^2 = d^{-1}\sum_{i\in[d]}\delta_i^2, \tag{3.103a}$$

$$m_3 - 3m_1m_2 + 2m_1^3 = d^{-1}\sum_{i\in[d]}\delta_i^3. \tag{3.103b}$$

For convenience, let $\varphi := \varphi(w) > 0$ be a constant. We have:

$$r^2\varphi(w)[r,r] \overset{3.97}{=} d^{-1}\varphi\langle P(w^{-1/2})(d^{-1}\,\text{tr}(\hat{r})e - \hat{r}), r\rangle \tag{3.104a}$$

$$= -\varphi(d^{-1}\,\text{tr}(\hat{r}^2) - d^{-2}\,\text{tr}(\hat{r})^2) \tag{3.104b}$$

$$= -\varphi(m_2 - m_1^2) \tag{3.104c}$$

$$\overset{3.103a}{=} -d^{-1}\varphi\sum_{i\in[d]}\delta_i^2 \leq 0. \tag{3.104d}$$

Similarly:

$$r^3\varphi(w)[r,r,r] \overset{3.98}{=} d^{-1}\varphi\hbar d^{-1}(d^{-1}\operatorname{tr}(\hat{r})^2 - \operatorname{tr}(\hat{r}^2))e - 2d^{-1}\operatorname{tr}(\hat{r})\hat{r} + 2\hat{r}^2, \hat{r}i \tag{3.105a}$$

$$= d^{-1}\varphi(d^{-1}(d^{-1}\operatorname{tr}(\hat{r})^2 - \operatorname{tr}(\hat{r}^2))\operatorname{tr}(\hat{r}) - 2d^{-1}\operatorname{tr}(\hat{r})\operatorname{tr}(\hat{r}^2) + 2\operatorname{tr}(\hat{r}^3)) \tag{3.105b}$$

$$= \varphi(m_1^3 - 3m_1m_2 + 2m_3) \tag{3.105c}$$

$$= \varphi(3m_1(m_2 - m_1^2) + 2(m_3 - 3m_1m_2 + 2m_1^3)) \tag{3.105d}$$

$$\overset{3.103}{=} d^{-1}\varphi\sum_{i2\mathrm{J}d\mathsf{K}}(3m_1\delta_i^2 + 2\delta_i^3) \tag{3.105e}$$

$$= d^{-1}\varphi\sum_{i2\mathrm{J}d\mathsf{K}}\delta_i^2(m_1 + 2\sigma_i). \tag{3.105f}$$

Finally, using (3.102), (3.104) and (3.105) the compatibility condition (3.100) is equivalent to nonnegativity of:

$$- r^3\zeta(\mathfrak{u})[\mathfrak{p},\mathfrak{p},\mathfrak{p}] - 3(r^2\ (\mathfrak{u})[\mathfrak{p},\mathfrak{p}])^{1=2}r^2\zeta(\mathfrak{u})[\mathfrak{p},\mathfrak{p}] \tag{3.106a}$$

$$= -d^{-1}\varphi\sum_{i2\mathrm{J}d\mathsf{K}}\delta_i^2(m_1 + 2\sigma_i) + 3k\sigma kd^{-1}\varphi\sum_{i2\mathrm{J}d\mathsf{K}}\delta_i^2 \tag{3.106b}$$

$$= d^{-1}\varphi\sum_{i2\mathrm{J}d\mathsf{K}}\delta_i^2(k\sigma k - m_1 + 2(k\sigma k - \sigma_i)). \tag{3.106c}$$

Clearly, $d^{-1}\varphi\delta_i^2 \geq 0$ and $\sigma_i \leq k\sigma k$ for all i 2 JdK. We have $m_1 \leq d^{-1}k\sigma k_1 \leq d^{-1=2}k\sigma k \leq k\sigma k$. Hence (3.106c) is nonnegative.

Thus (3.100) holds and compatibility is proved. Now by Nesterov and Nemirovskii [1994, Proposition 5.1.7], is a $(1 + d)$-LHSCB for K_{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 (3.99). Let $\mathfrak{u}$ 2 int(K_{rtdet}) and $\mathfrak{p} = (p, r)$ 2 R × V. For convenience, let

$\varphi := \varphi(w) > 0$ be a constant. We define the scalar constants:

$$\eta := d^{-1}\varphi\zeta^{-1}, \tag{3.107a}$$

$$\theta := 1 + \eta, \tag{3.107b}$$

$$\chi := -\zeta^{-1}p + \eta\,\mathrm{tr}(\hat{f}), \tag{3.107c}$$

$$\tau := \chi - d^{-1}\,\mathrm{tr}(\hat{f}), \tag{3.107d}$$

$$\upsilon := \mathrm{tr}(\hat{f}^2) - d^{-1}\,\mathrm{tr}(\hat{f})^2. \tag{3.107e}$$

Note that:

$$r_u(\zeta^{-1}) = \zeta^{-2}, \tag{3.108a}$$

$$r_u\eta = \zeta^{-1}\eta, \tag{3.108b}$$

$$r_u\chi = \zeta^{-1}\chi, \tag{3.108c}$$

$$r_w(\zeta^{-1}) = -\zeta^{-2}\,r\,\varphi(w) \overset{3.96}{=} -\zeta^{-1}\eta w^{-1}, \tag{3.108d}$$

$$r_w\eta = \eta(d^{-1} - \eta)w^{-1}, \tag{3.108e}$$

$$r_w\chi \overset{3.24}{=} \zeta^{-1}\eta p w^{-1} + \eta(d^{-1} - \eta)\,\mathrm{tr}(\hat{f})w^{-1} - \eta P(w^{-1})r \tag{3.108f}$$

$$= -\eta(\tau w^{-1} + P(w^{-1})r). \tag{3.108g}$$

The gradient of in (3.99) is:

$$g_u = \zeta^{-1}, \tag{3.109a}$$

$$g_w = -\zeta^{-1}\,r\,\varphi(w) - w^{-1} \tag{3.109b}$$

$$= -\theta w^{-1}. \tag{3.109c}$$

Differentiating (3.109), the Hessian product is:

$$H_u = r_u g_u p + r_u g_w[r] \tag{3.110a}$$
$$= \zeta^{-2} p - \zeta^{-1} \eta \operatorname{tr}(\hat{\Lambda}) \tag{3.110b}$$
$$= -\zeta^{-1} \chi, \tag{3.110c}$$
$$H_w = r_w g_u p + r_w g_w[r] \tag{3.110d}$$
$$\overset{3.24}{=} -\zeta^{-1} \eta p w^{-1} - \operatorname{tr}(\hat{\Lambda}) \eta (d^{-1} - \eta) w^{-1} + \theta P(w^{-1}) r \tag{3.110e}$$
$$= P(w^{-1/2})(\eta \tau e + \theta \hat{\Lambda}). \tag{3.110f}$$

Differentiating (3.110), the the third order directional derivative is:

$$T_u = r_u H_u p + r_u H_w[r] \tag{3.111a}$$
$$= -2\zeta^{-2} p \chi + \zeta^{-1} \eta (\tau \operatorname{tr}(\hat{\Lambda}) + \operatorname{tr}(\hat{\Lambda}^2)) + \zeta^{-1} \eta \operatorname{tr}(\hat{\Lambda}) \chi \tag{3.111b}$$
$$= \zeta^{-1} (2\chi^2 + \eta \upsilon), \tag{3.111c}$$
$$T_w = r_w H_u p + r_w H_w[r] \tag{3.111d}$$
$$\overset{3.25}{=} \zeta^{-1} \eta p \chi w^{-1} + \zeta^{-1} \eta p (\tau w^{-1} + P(w^{-1}) r) - \eta \tau P(w^{-1}) r +$$
$$\operatorname{tr}(\hat{\Lambda}) \tau \eta (d^{-1} - \eta) w^{-1} + \eta (-\eta (\tau \operatorname{tr}(\hat{\Lambda}) + \operatorname{tr}(\hat{\Lambda}^2)) + d^{-1} \operatorname{tr}(\hat{\Lambda}^2)) w^{-1} + \tag{3.111e}$$
$$\eta (d^{-1} - \eta) \operatorname{tr}(\hat{\Lambda}) P(w^{-1}) r - 2\theta P(w^{-1/2}) \hat{\Lambda}^2$$
$$= \eta (-\chi \tau + \zeta^{-1} p \chi + (d^{-1} - \eta)(\operatorname{tr}(\hat{\Lambda}) \tau + \operatorname{tr}(\hat{\Lambda}^2))) w^{-1} +$$
$$\eta (-\tau + \zeta^{-1} p + (d^{-1} - \eta) \operatorname{tr}(\hat{\Lambda})) P(w^{-1}) r - 2\theta P(w^{-1/2}) \hat{\Lambda}^2 \tag{3.111f}$$
$$= P(w^{-1/2})(\eta (-2\chi \tau + (d^{-1} - \eta) \upsilon) e - 2\eta \tau \hat{\Lambda} - 2\theta \hat{\Lambda}^2). \tag{3.111g}$$

In Lemma 3.7.2 below, we give a closed-form inverse Hessian product operator. This operator (3.112) a similar structure to the Hessian product operator (3.110), except that it applies $P(w^{1/2})$ instead of $P(w^{-1/2})$.

Lemma 3.7.2. *Letting* $r := P(w^{1/2})r \in V$*, the inverse Hessian product is:*

$$H_u = (\zeta^2 + d^{-1}\varphi^2)p + d^{-1}\varphi \operatorname{tr}(r), \tag{3.112a}$$

$$H_w = P(w^{1/2})(d^{-1}(\varphi p + \eta\theta^{-1}\operatorname{tr}(r))e + \theta^{-1}r). \tag{3.112b}$$

Proof. Note that the Hessian operator (3.110) is a positive definite linear operator, so it has a unique inverse linear operator. We show that $(\nabla^2 f(w))^{-1}(\nabla^2 f(w)[\hat{p}]) = \hat{p}$. Into (3.112), we substitute the values from (3.110) 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:

$$r = P(w^{1/2})H_w = \eta\tau e + \theta\hat{r}, \tag{3.113a}$$

$$\operatorname{tr}(r) = d\eta\tau + \theta \operatorname{tr}(\hat{r}). \tag{3.113b}$$

We have:

$$H_u = (\zeta^2 + d^{-1}\varphi^2)(-\zeta^{-1}\chi) + d^{-1}\varphi(d\eta\tau + \theta \operatorname{tr}(\hat{r})) \tag{3.114a}$$

$$= -\zeta\chi + \varphi\eta(\tau - \chi) + d^{-1}\varphi\theta \operatorname{tr}(\hat{r}) \tag{3.114b}$$

$$= -\zeta(\chi - \eta \operatorname{tr}(\hat{r})) \tag{3.114c}$$

$$= p, \tag{3.114d}$$

and:

$$H_w = P(w^{1/2})(d^{-1}(-\varphi\zeta^{-1}\chi + \eta\theta^{-1}(d\eta\tau + \theta \operatorname{tr}(\hat{r})))e + \theta^{-1}(\eta\tau e + \theta\hat{r})) \tag{3.115a}$$

$$= P(w^{1/2})(\eta(-\tau + \eta\theta^{-1}\tau + \theta^{-1}\tau)e + \hat{r}) \tag{3.115b}$$

$$= P(w^{1/2})(\hat{r}) \tag{3.115c}$$

$$= r. \tag{3.115d}$$

Hence (3.112) is the unique inverse operator of (3.110). □

We note the polynomial-like structure of the oracles. In particular, the w compo-

nents 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 H is computed by applying $P(w^{1=2})$ to a polynomial of degree one in r. This structure leads to simple, e□cient, 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 $V = S^d$ and $V = H^d$, 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 to 3.8.4, 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 we describe in Section 3.8.3 and show that Hypatia often solves the NFs much more e□ciently than Hypatia, MOSEK, or ECOS solve the EFs. Finally, in Section 3.8.5, we exemplify the computational impact of e□cient oracle procedures by comparing the performance of our closed-form inverse Hessian product formula in (3.64) with that of a naive direct solve using the explicit Hessian matrix.

3.8.1 Spectral function cones in Hypatia

Recall Hypatia's generic cone interface allows specifying a *vectorized* proper cone $K \subseteq R^q$ for some dimension q. Recall that for the real symmetric PSD cone S^d, we use the standard *svec* transformation, which rescales and stores only the elements of the matrix triangle in a vector of dimension $d(d + 1)/2$. For the complex Hermitian PSD cone H^d, we perform a modified *svec* transformation to a d^2-dimensional vector,

storing each real diagonal element as a single element and each complex off-diagonal element in the triangle as two (rescaled) consecutive real elements (the real part followed by the imaginary part). These transformations preserve inner products and the self-duality of cones of squares.

We adapt these transformations to enable vectorization of spectral function cones. For example, for the epigraph-perspective cone K_p in (3.30), 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 K_p^* in (3.31).

For the domains $\mathbb{R}^d$, $\mathbb{S}^d$, and $\mathbb{H}^d$, we implement vectorizations of the MMD cone K_{MMD}, the log-determinant cone K_{logdet}, and the root-determinant cone (K_{rtdet}).[1] This allows the user to model with these cones or their dual cones in Hypatia. As we discuss at the end of Sections 3.5.4 and 3.7.4, for K_{logdet} and K_{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 K_{MMD}, we predefine the MMD functions in Table 3.1 (e.g. *NegEntropy*). Recall that K_{MMD}^* in (3.78) 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 K_{MMD}^*. We write *NegLogdet* or *NegRtdet* for the negative log-determinant or negative root-determinant function, the epigraph of which we represent using K_{logdet} or K_{rtdet}. In our examples, we choose not to use K_{rtdet}^* or K_{logdet}^* (or equivalently, K_{MMD}^* with *NegLogConj*), because these particular dual cones provide little additional modeling power over their primal cones.

[1] Our K_{logdet} implementation is for the hypograph of the log-determinant, rather than the epigraph of negative log-determinant considered 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.

3.8.2 Building natural and extended formulations

To assess the computational value of our new cones and e□cient oracles, we compare the performance of Hypatia on NFs over K_{MMD}, K_{logdet}, and K_{rtdet} against that of other conic IPM solvers on equivalent EFs. ECOS [Domahidi et al., 2013] is another open-source conic IPM solver, but it only supports nonnegative, second-order, and three-dimensional exponential cones. Recall MOSEK version 9 [MOSEK ApS, 2020] supports the same cones as ECOS as well as three-dimensional power cones and real symmetric PSD cones (the standard cones). To build the standard cone EFs, we use a variety of formulation techniques, some of which we discuss and analyze in Coey et al. [2021d].[2]

For $V = \mathsf{R}^d$, our EFs are constructed as follows. 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 uses $V = \mathsf{R}^d$.

For $V = \mathsf{S}^d$, our EFs are constructed as follows. For most spectral functions, we adapt the EF from Ben-Tal and Nemirovski [2001, Proposition 4.2.1], which requires using an EF from the $V = \mathsf{R}^d$ case for the corresponding spectral function, and adding constraints on the sum of the i largest eigenvalues of a matrix for each $i \in \mathbb{J}d\mathsf{K}$. This is a large formulation with many additional variables and PSD constraints. For *NegLog* and *NegRtdet*, we use a much simpler EF from Ben-Tal and Nemirovski [2001, Example 18.d]. 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 $V = \mathsf{H}^d$, we reformulate any complex PSD cone constraint to a real PSD cone constraint with twice the side dimension [MOSEK ApS, 2020, Section 6.2.7]. The examples in Section 3.8.4 use $V = \mathsf{S}^d$ and the example in Section 3.8.4 uses $V = \mathsf{H}^d$.

[2] The EFs build automatically via the functions in https://github.com/chriscoey/Hypatia.jl/blob/master/examples/spectral_functions_JuMP.jl.

3.8.3 Computational methodology

We perform all instance generation, computational experiments, and results analysis with Ubuntu 21.10, Julia 1.8.0-DEV.862, and Hypatia 0.5.3.[3] 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 to 3.8.4, 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 PDIPM solvers in MOSEK version 9 and ECOS version 2.0.5 (with no features disabled). Hypatia uses the default algorithmic implementation that we describe in Chapter 2 (the combined directions method with the QR-Cholesky linear system procedure). We limit each 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 Coey et al. [2021d, Equation 23]. In Figures 3-2 to 3-5, we plot solve times in seconds against an instance size parameter, excluding solves for which $\epsilon > 10^{-5}$. Hypatia-NF (i.e. Hypatia solving the NF) is faster than any EF solver (Hypatia-EF, MOSEK-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

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,

[3] 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.

given some prior information expressed with $d/2$ linear equality constraints. Adapting Boyd and Vandenberghe [2004, Section 7.2], the problem is:

$$\min_{\rho \in \mathbb{R}^d} \quad \varphi(\rho) : \tag{3.116a}$$

$$\operatorname{tr}(\rho) = d, \tag{3.116b}$$

$$A\rho = b. \tag{3.116c}$$

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 (3.116). The solver timings are summarized in Figure 3-2. Note that for *NegRtdet*, no solve times are plotted for MOSEK-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 MOSEK-EF is significantly slower than Hypatia-NF. We do not plot results for *NegLogdet* (the K_{logdet} formulation using the specialized oracles from Section 3.5.4) as they are nearly identical to the results for K_{MMD}/*NegLog*; however, the efficiency benefits of *NegLogdet* are realized for the matrix domain in Section 3.8.4.

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:

$$\min_{\rho \in \mathbb{R}^{2d}} \quad \varphi(V \operatorname{Diag}(\rho) V^\top) : \tag{3.117a}$$

$$\operatorname{tr}(\rho) = 2d, \tag{3.117b}$$

$$\rho \geq 0, \tag{3.117c}$$

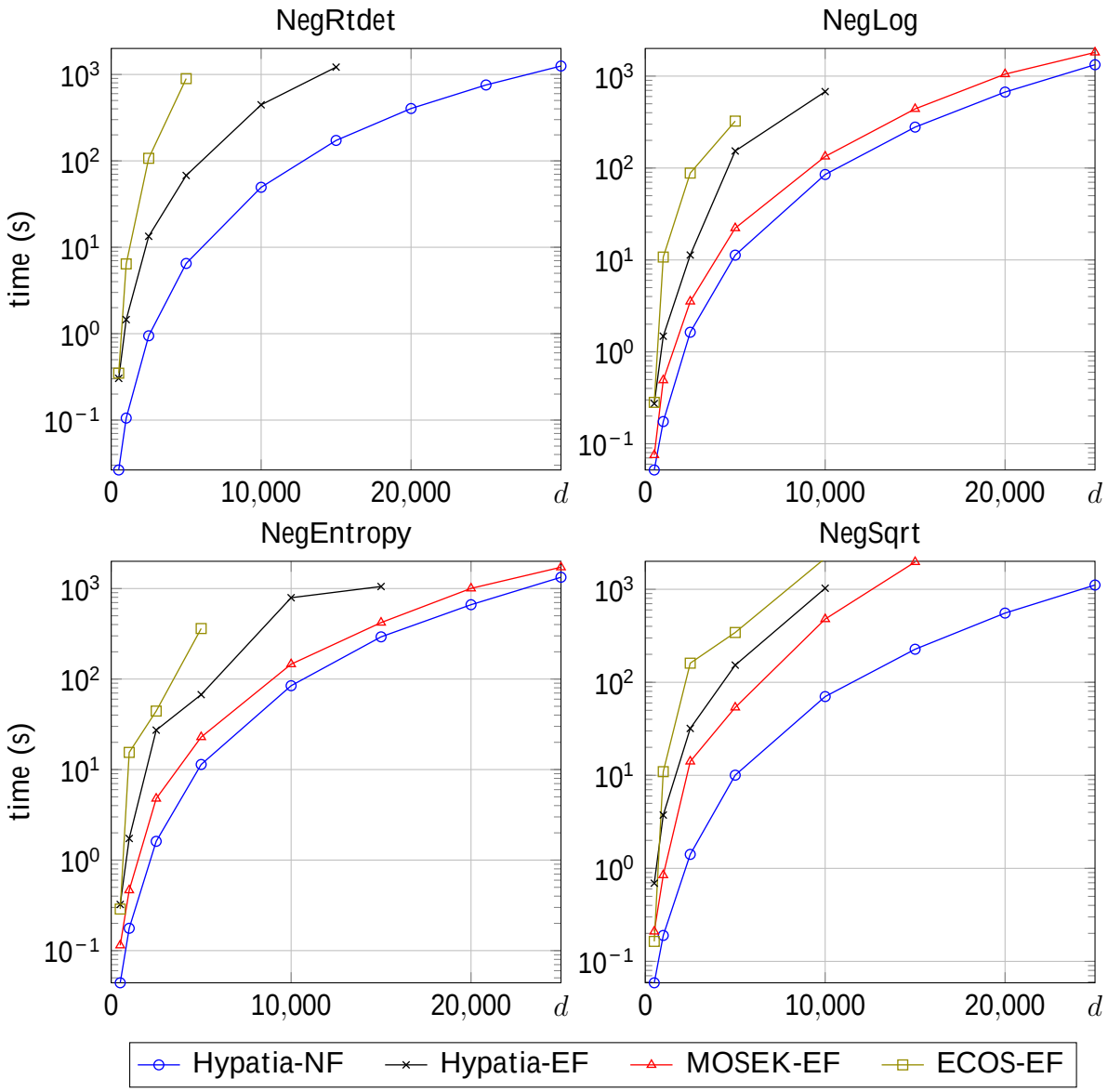

Figure 3-2: Nonparametric distribution estimation solver performance.

where $V^\top$ is the transpose of V and Diag(ρ) is the diagonal matrix of ρ. For four different φ on $\mathbb{S}^d$ and various d, we build random instances of (3.117). The solver timings are summarized in Figure 3-3. Since ECOS does not support $\mathbb{S}^d$, we only compare with MOSEK. The *Hypatia-NegLogdet* curve indicates that Hypatia with K_{logdet} is somewhat more efficient than Hypatia with the equivalent K_{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.

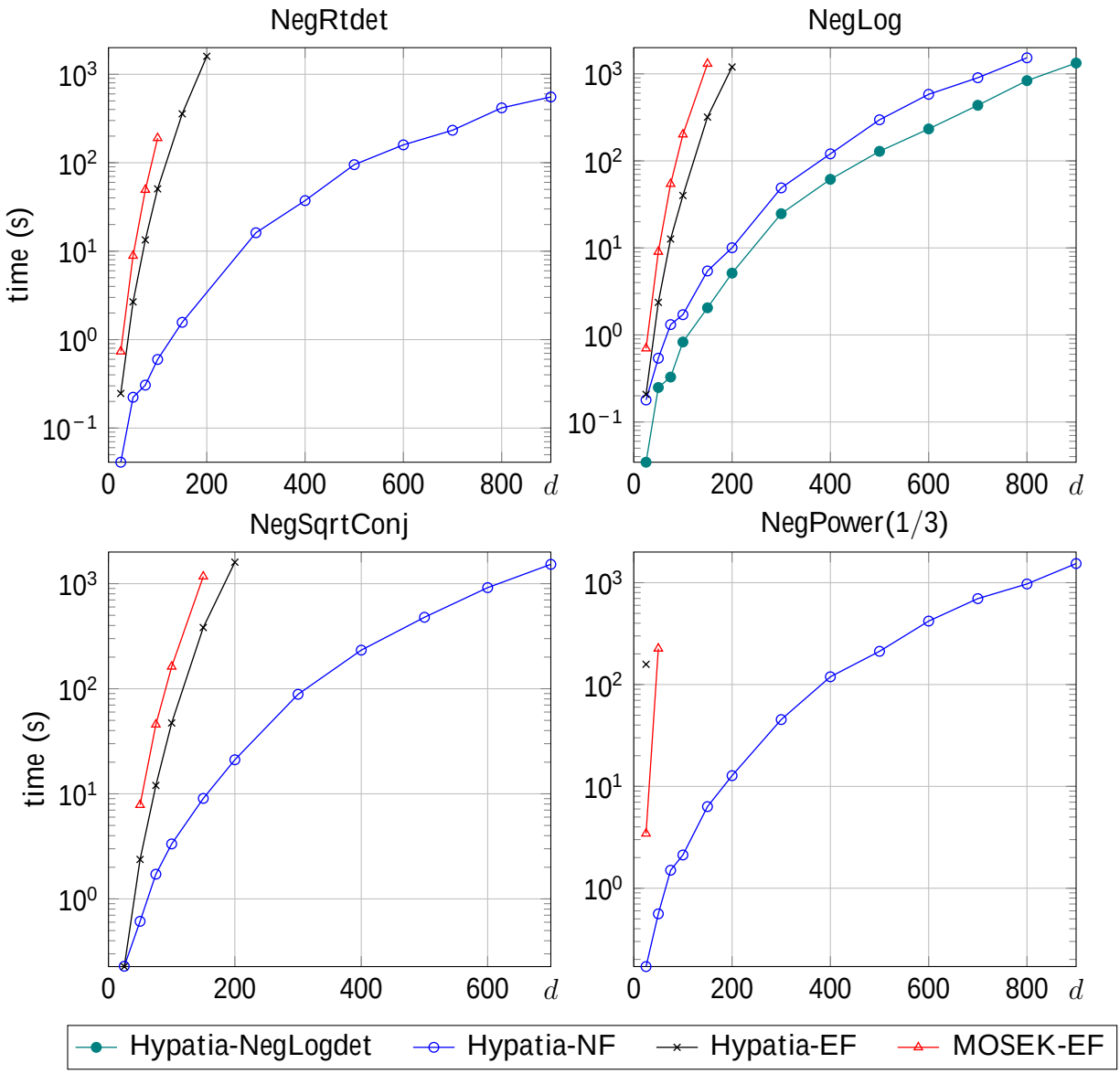

Figure 3-3: Experiment design solver performance.

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 2 R^U be the monomial coe□cients of the polynomial. We seek a Gram matrix ρ 2 S^L corresponding to b [Parrilo, 2012, Lemma 3.33] that minimizes

a convex spectral function φ:

$$\min_{\rho \in \mathbb{S}^L} \quad \varphi(\rho) : \tag{3.118a}$$

$$C\,\mathrm{vec}(\rho) = b, \tag{3.118b}$$

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 (3.118), 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 solver timings are summarized in Figure 3-4 (a log-log plot).

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 Fawzi and Fawzi [2018, Section 3.1]. The variable $\rho \in \mathbb{R}^d$ is a probability distribution on the d-dimensional input alphabet. For $i \in \mathbb{J}d\mathbb{K}$, let $P_i \in \mathbb{H}^d$ be fixed density matrices satisfying $\mathrm{tr}(P_i) = 1$. Letting φ represent the trace of *NegEntropy* on $\mathbb{H}^d$, the formulation is:

$$\min_{\rho \in \mathbb{R}^d} \quad \varphi\big(\textstyle\sum_{i \in \mathbb{J}d\mathbb{K}} \rho_i P_i\big) - \sum_{i \in \mathbb{J}d\mathbb{K}} \rho_i \varphi(P_i) : \tag{3.119a}$$

$$\mathrm{tr}(\rho) = 1, \tag{3.119b}$$

$$\rho \geq 0. \tag{3.119c}$$

We generate random instances of (3.119), varying d. The solver timings are summarized in Figure 3-5.

3.8.5 Inverse Hessian product oracle

To illustrate the importance of our efficient and numerically stable oracle procedures, we compare the performance of two different approaches to computing the inverse Hessian product oracle H in (3.38c) for $\mathcal{K}_{\mathrm{MMD}}$ cones. The naive approach is to

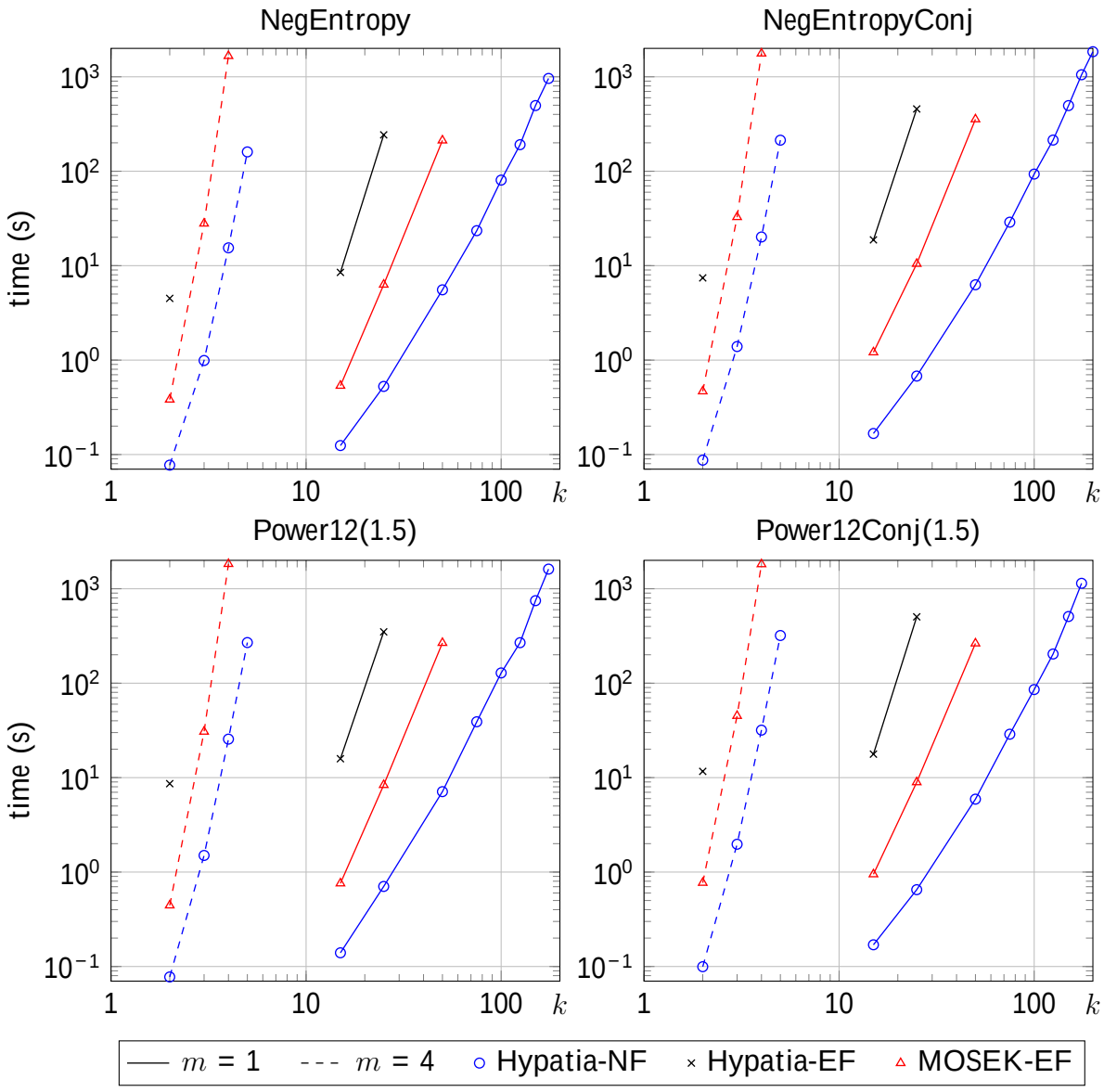

Figure 3-4: Central polynomial Gram matrix solver performance.

compute the explicit Hessian matrix, perform a Cholesky factorization, and use a direct linear solve. Alternatively, we derive a closed-form formula for H in (3.64), since K_{MMD} is a special case of K_p with a separable spectral function. This formula is essentially as easy to compute as the Hessian product oracle H in (3.47) (which does not use an explicit Hessian matrix). In Table 3.2, we compare the worst-case memory and time complexities for these procedures.

To compare the practical performance of these procedures, we perform computa-

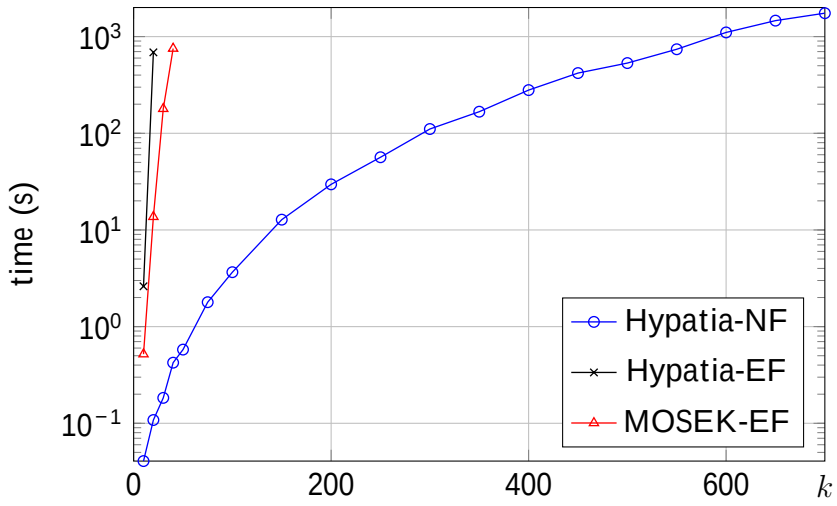

Figure 3-5: Classical-quantum channel capacity solver performance.

		closed-form formula		factorize and solve	
V	dim(K_{MMD})	memory	time	memory	time
R^d	$O(d)$	$O(d)$	$O(d)$	$O(d^2)$	$O(d^3)$
S^d or H^d	$O(d^2)$	$O(d^2)$	$O(d^3)$	$O(d^4)$	$O(d^6)$

Table 3.2: Cone dimension and worst-case complexities for the two inverse Hessian product procedures.

tional experiments using Hypatia. We first solve NF instances of a range of sizes for the examples from Section 3.8.4 (with $V = \mathsf{R}^d$) and Section 3.8.4 (with $V = \mathsf{S}^d$), using K_{MMD} with the *NegEntropy* function. For each instance, at Hypatia's final PDIPM iterate, we take the direction $r = g$ (i.e. the gradient oracle in (3.38a) at the iterate) and compute H for this direction using both procedures. To measure the numerical accuracy of each procedure, we compute $\epsilon := |1 - \nu^{-1}\langle H, g\rangle|$, which measures the violation of a particular identity [Nesterov and Todd, 1997, Equation 2.5] satisfied by a logarithmically homogeneous function such as the LHSCB 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 R^d example and at $d = 20, 50, 200$ for the S^d example; when this occurs, Hypatia uses a Bunch-Kaufman factorization as a fallback (note Julia

calls performant OpenBLAS routines for the Cholesky and Bunch-Kaufman factorizations). 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 H faster and with greater numerical accuracy. Also, the closed-form procedure is much more memory e□cient than the factorization-based procedure, as it never forms an explicit Hessian matrix.

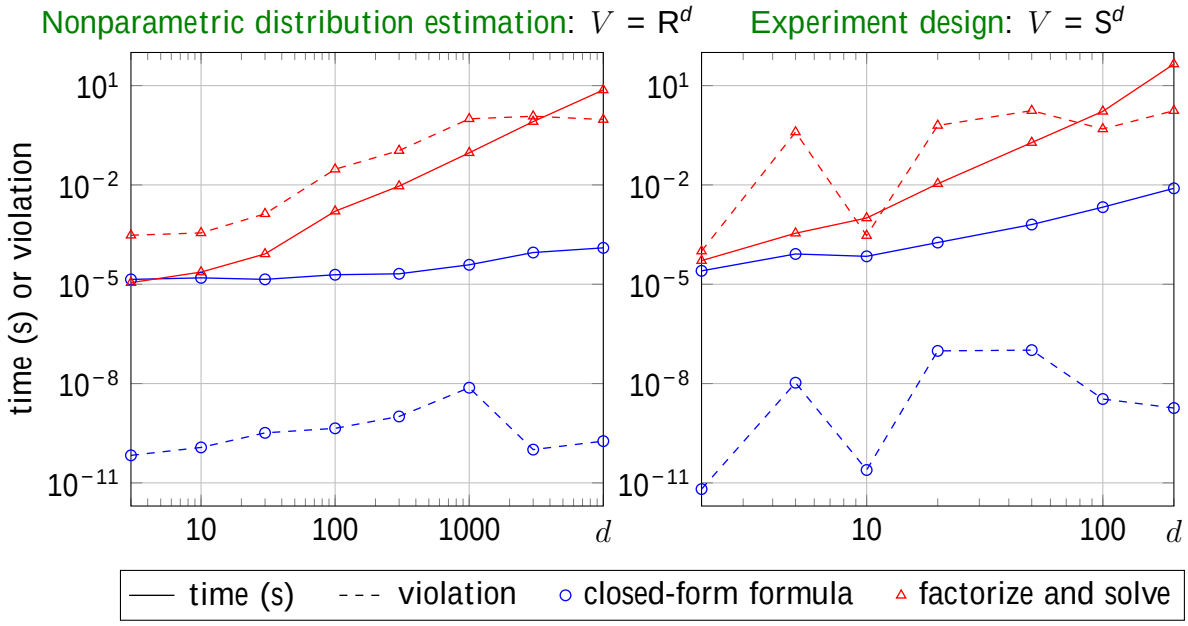

Figure 3-6: For instances of two examples using K_{MMD} with *NegEntropy*, the speed and logarithmic homogeneity condition violation (at the final iterate) for the two inverse Hessian product procedures.

Chapter 4

Oracles for slices of the PSD cone

This chapter is based on an appendix from the submitted paper Coey et al. [2021c].

4.1 Intersections of linear slices of the PSD cone

Many of the cones in Hypatia, including some polynomial cones that we focus on in Chapter 5, can be described as a linear slice of the PSD cone. The aim of this chapter is to describe e□cient techniques for evaluating the oracles of cones in this class for the algorithm from Chapter 2. These techniques are also implemented in practice for the cones in Chapter 5.

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

$$K := \{s \in \mathbb{R}^q : \mathcal{A}(s) \succeq 0\}, \tag{4.1}$$

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

$$K^* := \{s \in \mathbb{R}^q : \exists S \succeq 0, s = \mathcal{A}^*(S)\}. \tag{4.2}$$

We note that for $K = \mathbb{S}^l_+$ (the self-dual vectorized PSD cone), we can let $q = \mathrm{sd}(j)$, $\mathcal{A}(s) = \mathrm{mat}(s)$, and $\mathcal{A}^*(S) = \mathrm{vec}(S)$. Given a point $s \in \mathbb{R}^q$, strict feasibility for K can

be checked, for example, by attempting a Cholesky factorization $(s) = LL^\top$, where L is lower triangular.

For K we have the LHSCB $f(s) = -\log\det((s))$ with parameter $\nu = $. Given a point $s \in \text{int}(K)$, we have $(s) \in \mathbb{S}^{|}$ and its inverse $^{-1}(s) \in \mathbb{S}^{|}$. For a direction $\delta \in \mathbb{R}^q$, for f at s we can write the gradient, and the Hessian and TOO applied to δ, as (compare to Papp and Yildiz [2019, Section 3]):

$$g(s) = - \ ^*(\ ^{-1}(s)), \tag{4.3a}$$

$$H(s)\delta = \ ^*(\ ^{-1}(s) \ (\delta) \ ^{-1}(s)), \tag{4.3b}$$

$$\mathrm{T}(s,\delta) = \ ^*(\ ^{-1}(s) \ (\delta) \ ^{-1}(s) \ (\delta) \ ^{-1}(s)). \tag{4.3c}$$

If we have, for example, a Cholesky factorization $(s) = LL^\top$ (computed during the feasibility check), then the oracles in (4.3) are easy to compute if and * are easy to apply. We can compute the TOO (4.3c) using the following steps:

$$Y := L^{-1} \ (\delta) \ ^{-1}(s), \tag{4.4a}$$

$$Z := Y^\top Y = \ ^{-1}(s) \ (\delta) \ ^{-1}(s) \ (\delta) \ ^{-1}(s), \tag{4.4b}$$

$$\mathrm{T}(s,\delta) = \ ^*(Z). \tag{4.4c}$$

We note (4.4a) can be computed using back-substitutions with L, and (4.4b) is a simple symmetric outer product. We use this approach to derive simple TOO procedures for K_{LMI} in Section 4.2 and for K^*_{SOS} and K^*_{matSOS} in Section 4.3.

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

$$K := \{s \in \mathbb{R}^q : \ _l(s) \ \ 0, \forall l \in \llbracket r \rrbracket\}, \tag{4.5}$$

where $_l : \mathbb{R}^q \to \mathbb{S}^{|_l}$, for $l \in \llbracket r \rrbracket$. Then the dual cone can be characterized as:

$$K^* := \left\{s \in \mathbb{R}^q : \exists S_1, \ldots, S_r \ \ 0, s = \textstyle\sum_{l \in \llbracket r \rrbracket} \ ^*_l(S_l)\right\}. \tag{4.6}$$

Feasibility for K can be checked by performing r Cholesky factorizations. If we let $f_l(s) = -\log\det(\ _l(s)), \forall l \in [r]$, then $f(s) = \sum_{l\in[r]} f_l(s)$ is an LHSCB for K with parameter $\nu = \sum_{l\in[r]} \nu_l$. The oracles $g(s)$, $H(s)\delta$ (and the explicit Hessian matrix), and $T(s,\delta)$ can all be computed as sums over $l \in [r]$ of the terms in (4.3c).

4.2 LMI cone

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

$$(w) := \sum_{i\in[d]} w_i P_i \in S^s, \tag{4.7a}$$

$$^*(W) := (\langle P_i, W\rangle)_{i\in[d]} \in R^d. \tag{4.7b}$$

Our implementation uses specializations of (4.3) and (4.4) for K_{LMI}. For $w \in \mathrm{int}(K_{\mathrm{LMI}})$ and direction $\delta \in R^d$, using the Cholesky factorization $(w) = LL^\top$, we compute:

$$Q_i := L^{-1}P_iL^{-\top} \in S^s \quad \forall i \in [d], \tag{4.8a}$$

$$g(w) = (-\mathrm{tr}(Q_i))_{i\in[d]}, \tag{4.8b}$$

$$R := \sum_{j\in[d]} \delta_j Q_j \in S^s, \tag{4.8c}$$

$$H(w)\delta = (\langle Q_i, R\rangle)_{i\in[d]}, \tag{4.8d}$$

$$T(w,\delta) = (\langle Q_i, R^\top R\rangle)_{i\in[d]}, \tag{4.8e}$$

and we compute the explicit Hessian oracle as:

$$(H(w))_{i,j} = \langle Q_i, Q_j\rangle \quad \forall i, j \in [d]. \tag{4.9}$$

The symmetric form of Q_i and the use of a symmetric outer product $R^\top R$ in (4.8e) are beneficial for efficiency and numerical performance.

4.3 Matrix and scalar WSOS dual cones

Recall from Chapter 2 that Hypatia uses LHSCBs for $K^*_{\text{SOS}}, K^*_{\text{matSOS}}$, because LHSCBs for $K_{\text{SOS}}, K_{\text{matSOS}}$ with tractable oracles are not known. Since the scalar WSOS dual cone K^*_{SOS} is a special case of the matrix WSOS dual cone K^*_{matSOS} with $t = 1$, we only consider K^*_{matSOS} here. In general, K^*_{matSOS} is an intersection of r slices of K (see (4.5)), 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:

$$\rho_{i;j} := \begin{cases} 1 & \text{if } i = j, \\ \sqrt{2} & \text{otherwise.} \end{cases} \tag{4.10}$$

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

$$(w) := \left[P^\top \operatorname{Diag}\left(\rho_{i;j}^{-1} w_{\max(i;j);\min(i;j);:}\right) P\right]_{i;j \in \mathbb{J}t\mathbb{K}} \in \mathbb{S}^{st}, \tag{4.11a}$$

$$^*(W) := (\rho_{i;j} \operatorname{diag}(P(W)_{i;j} P^\top))_{i \in \mathbb{J}t\mathbb{K}; j \in \mathbb{J}i\mathbb{K}} \in \mathbb{R}^{\text{sd}(t)d}, \tag{4.11b}$$

where $w = (w_{i;j;:})_{i \in \mathbb{J}t\mathbb{K}; j \in \mathbb{J}i\mathbb{K}}$ 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 $[S_{i;j}]_{i;j \in \mathbb{J}t\mathbb{K}}$ 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 (4.3) and (4.4). Suppose we have $w \in \operatorname{int}(K^*_{\text{matSOS}})$ and direction $\delta \in \mathbb{R}^{\text{sd}(t)d}$, and a Cholesky factorization $(w) = LL^\top$. For each $i, j \in \mathbb{J}t\mathbb{K} : i \geq j$ and $p \in \mathbb{J}d\mathbb{K}$, we

implicitly compute oracles according to:

$$(Q)_{i;j;p} := ((L^{-1})_{i;j} P^{>})e_p \ 2 \ \mathsf{R}^s, \tag{4.12a}$$

$$(g(w))_{i;j;p} = -\rho_{i;j} Q^{>}_{i;:;p} Q_{:;j;p}, \tag{4.12b}$$

$$(R)_{i;j;p} := (L^{-1} \quad (\delta)(L^{-1})^{>} Q)_{i;j} e_p \ 2 \ \mathsf{R}^s, \tag{4.12c}$$

$$(H(w)\delta)_{i;j;p} = \rho_{i;j} Q^{>}_{i;:;p} R_{:;j;p}, \tag{4.12d}$$

$$(\mathsf{T}(w,\delta))_{i;j;p} = \rho_{i;j} R^{>}_{i;:;p} R_{:;j;p}. \tag{4.12e}$$

Letting $Q^2_{i;j} := (Q^{>}Q)_{i;j} \ 2 \ \mathsf{S}^d$, we compute the Hessian oracle according to:

$$(H(w))_{(i;j;:);(k;l;:)} = \tfrac{1}{2}\rho_{i;j}\rho_{k;l}\big(Q^2_{i;k} \quad Q^2_{j;l} + Q^2_{i;l} \quad Q^2_{j;k}\big) \ 2 \ \mathsf{S}^d \quad 8i,j,k,l \ 2 \ \mathtt{J}t\mathsf{K}, \tag{4.13}$$

where $X \quad Y \ 2 \ \mathsf{S}^d$ denotes the Hadamard (elementwise) product of $X, Y \ 2 \ \mathsf{S}^d$.

4.4 Sparse PSD cone

Let $S = ((i_l, j_l))_{l2\mathtt{J}d\mathsf{K}}$ 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 S to be a chordal sparsity pattern (unlike Andersen et al. [2013], Burer [2003]), as this restriction is not necessary for the oracles Hypatia uses. Note $s \le d \le \mathrm{sd}(s)$. For K_{sPSD} parametrized by S, we define $\quad : \mathsf{R}^d \ ! \ \mathsf{S}^s$ as the linear operator satisfying, for all $i,j \ 2 \ \mathtt{J}s\mathsf{K} : i \ge j$:

$$(\quad (w))_{i;j} := \begin{cases} \rho^{-1}_{i;j} w_l & \text{if } i = i_l = j = j_l, \\ 0 & \text{otherwise,} \end{cases} \tag{4.14}$$

where $\rho_{i;j}$ is given by (4.10). Then $\quad^*$ is the vectorized projection onto S, i.e. for $W \ 2 \ \mathsf{S}^s$:

$$^*(W) := (\rho_{i;j} W_{i;j})_{(i;j)2S} \ 2 \ \mathsf{R}^d. \tag{4.15}$$

Consider $w \in \text{int}(K_{\text{sPSD}})$ and direction $\delta \in \mathbb{R}^d$. The gradient (4.3a) and Hessian product (4.3b) for K_{sPSD} can be computed using Andersen et al. [2013, Algorithms 4.1 and 5.1]. To derive the TOO, we use the fact that:

$$-2\mathrm{T}(w,\delta) = \nabla^3 f(w)[\delta,\delta] = \tfrac{d^2}{dt^2} \nabla f(w + t\delta)\big|_{t=0}. \tag{4.16}$$

In order to succinctly describe our TOO approach as an extension of the procedures in Andersen et al. [2013], we describe an approach based on a sparse LDL factorization of $\nabla^2 f(w)$. However, our current implementation in Hypatia uses a sparse Cholesky ($LL^\top$) factorization, which is very similar to the LDL-based approach here. We compute the sparse Cholesky factors using Julia's SuiteSparse wrapper of CHOLMOD [Chen et al., 2008]. We note that Hypatia implements a *supernodal* generalization (see Andersen et al. [2013, Section 7]) of the TOO procedure we describe below. Before we describe the TOO procedure, we repeat useful definitions from Andersen et al. [2013], define higher order derivative terms, and differentiate several equations that are used for the gradient and Hessian oracles. As discussed in Section 2.5, Hypatia computes the feasibility check and gradient oracles before the TOO, and our TOO procedure reuses cached values computed for these oracles.

We define:

$$R := \nabla f(w + t\delta)). \tag{4.17}$$

Let $LDL^\top = \nabla^2 f(w)$ 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* [Andersen et al., 2013, Section 2], and each column of L corresponds to a node of this tree. Let I_k be the ordered row indices of nonzeros below the diagonal in column k of L, and let $J_k = I_k \cup \{k\}$. Let $\text{ch}(i)$ denote the children of node i in the tree. For an index set I let $I(i)$ denote the ith element. For index sets $J \subseteq I$, we define $E_{I;J} \in \mathbb{R}^{|I| \times |J|}$ satisfying, $i \in [|I|]$, $j \in [|J|]$:

$$(E_{I;J})_{i,j} := \begin{cases} 1 & \text{if } I(i) = J(j), \\ 0 & \text{otherwise.} \end{cases} \tag{4.18}$$

Let U_i be the update matrix for node i (see Andersen et al. [2013, Equation 14]):

$$U_i := -\sum_{k \in \mathrm{ch}(i)[fig} D_{k;k} L_{I_i;k} L_{I_i;k}^{\top}. \tag{4.19}$$

Let 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 (4.16). For convenience, we let:

$$L_j := \begin{bmatrix} 1 & 0 \\ -L_{I_j;j} & I \end{bmatrix}. \tag{4.20}$$

Suppose we have computed D, L, U according to Andersen et al. [2013, Equation 30]. Differentiating Andersen et al. [2013, Equation 30] once with respect to t gives:

$$\begin{bmatrix} \dot{D}_{j;j} & P_j^{\top} \\ P_j & 2D_{j;j}\dot{L}_{I_j;j}\dot{L}_{I_j;j}^{\top} + \dot{U}_j \end{bmatrix} = L_j \left(\sum_{i \in \mathrm{ch}(j)} E_{J_j;I_i} \dot{U}_i E_{J_j;I_i}^{\top} \right) L_j^{\top}, \tag{4.21}$$

where $P_j := 2D_{j;j}\dot{L}_{I_j;j} + \dot{D}_{j;j}L_{I_j;j}$ for convenience. This allows us to compute $\dot{D}$, $\dot{L}$, $\dot{U}$. Andersen et al. [2013, Equations 21 and 22] show that:

$$R_{I_j;j} = -R_{I_j;I_j} L_{I_j;j}, \tag{4.22a}$$

$$\begin{bmatrix} R_{j;j} & R_{I_j;j}^{\top} \\ R_{I_j;j} & R_{I_j;I_j} \end{bmatrix} \begin{bmatrix} 1 \\ L_{I_j;j} \end{bmatrix} = \begin{bmatrix} D_{j;j}^{-1} \\ 0 \end{bmatrix}, \tag{4.22b}$$

for each node j. Differentiating (4.22a) once with respect to t gives:

$$\dot{R}_{I_j;j} = -R_{I_j;I_i}\dot{L}_{I_j;j} - \dot{R}_{I_j;I_j} L_{I_j;j}. \tag{4.23}$$

Differentiating (4.22b) twice and substituting (4.22a) and (4.23), we have:

$$\begin{bmatrix} \ddot{R}_{j;j} & \ddot{R}_{I_j;j}^{\top} \\ \ddot{R}_{I_j;j} & \ddot{R}_{I_j;I_j} \end{bmatrix} = L_j^{\top} \begin{bmatrix} 2D_{j;j}^2 D_{j;j}^{-3} - \ddot{D}_{j;j} D_{j;j}^{-2} + 2\dot{L}_{I_j;j}^{\top} \dot{R}_{I_j;I_j} \dot{L}_{I_j;j} & Q_j^{\top} \\ Q_j & \ddot{R}_{I_j;I_j} \end{bmatrix} L_j, \tag{4.24}$$

where $Q_j := -R_{I_j;I_j}L_{I_j;j} - 2R_{I_j;I_j}L_{I_j;j}$ for convenience. This allows us to compute R. Finally, by (4.16) and (4.17), we can compute the TOO as:

$$-2\mathsf{T}(w,\delta) = {}^{*}(R). \tag{4.25}$$

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

$$= (\delta)\ 2\ \mathsf{S}^{s}. \tag{4.26}$$

Following Andersen et al. [2003], we define K and M as sparse matrices with the same structure as L, satisfying for all j 2 $\mathtt{J}s\mathtt{K}$:

$$K_{j;j} = D_{j;j}, \tag{4.27a}$$

$$K_{I_j;j} = D_{j;j}L_{I_j;j}, \tag{4.27b}$$

$$M_{j;j} = D_{j;j}^{-2}K_{j;j}, \tag{4.27c}$$

$$M_{I_j;j} = D_{j;j}^{-1}R_{I_j;I_j}K_{I_j;j}. \tag{4.27d}$$

The first three steps in the TOO procedure below compute D, L, U, and R and are identical to steps in Andersen et al. [2013, Algorithm 5.1].

1. Iterate over j 2 $\mathtt{J}s\mathtt{K}$ in topological order, computing $K_{J_j;j}$ and U_j according to:

$$\begin{bmatrix} K_{j;j} & K_{I_j;j}^{>} \\ K_{I_j;j} & U_j^{>} \end{bmatrix} = L_j \left(\begin{bmatrix} j;j & {}^{>}_{I_j;j} \\ I_j;j & 0 \end{bmatrix} + \sum_{i2\mathsf{ch}(j)} E_{J_j;I_i}U_i^{>}E_{J_j;I_i}^{>} \right) L_j^{>}. \tag{4.28}$$

2. For j 2 $\mathtt{J}s\mathtt{K}$, store $D_{j;j}$ and $L_{I_j;j}$ from (4.27a) and (4.27b), and compute $M_{J_j;j}$ from (4.27c) and (4.27d).

3. Iterate over j 2 $\mathtt{J}s\mathtt{K}$ in reverse topological order, computing $R_{J_j;j}$ according to:

$$\begin{bmatrix} R_{j;j} & R_{I_j;j}^{>} \\ R_{I_j;j} & R_{I_j;I_j} \end{bmatrix} = L_j^{>} \begin{bmatrix} M_{j;j} & M_{I_j;j}^{>} \\ M_{I_j;j} & R_{I_j;I_j} \end{bmatrix} L_j, \tag{4.29}$$

and updating matrices $R_{I_j;I_j}$ for each child $i \in \mathrm{ch}(j)$ of vertex j according to:

$$R_{I_i;I_i} = E_{J_j;I_i}^{\top} \begin{bmatrix} R_{j;j} & R_{I_j;j}^{\top} \\ R_{I_j;j} & R_{I_j;I_j} \end{bmatrix} E_{J_j;I_i}. \tag{4.30}$$

4. Iterate over $j \in \mathrm{J}s\mathrm{K}$ in topological order, computing $D_{j;j}, L_{I_j;j}, U_j$ from (4.21).

5. Iterate over $j \in \mathrm{J}s\mathrm{K}$ in reverse topological order, computing $R_{j;j}$, $R_{I_j;j}, R_{I_j;I_j}$ from (4.24).

6. Compute $\mathrm{T}(w, \delta)$ using R and (4.25).

4.5 Euclidean norm cone and Euclidean norm square cone

Although $K_{\ell_2}, K_{\mathrm{sqr}} \subseteq \mathbb{R}^q$ are inverse linear images of $\mathbb{S}^q$ and hence admit LHSCBs with parameter $\nu = q$, we use the standard LHSCBs with parameter $\nu = 2$, which have a different form (see Vandenberghe [2010, Section 2.2]). For $K_{\ell_2}, K_{\mathrm{sqr}}$, the LHSCB is $f(s) = -\log(s^\top J s)$, where $J \in \mathbb{S}^q$ is defined according to, for $i, j \in \mathrm{J}q\mathrm{K} : i \geq j$:

$$J_{i;j} := \begin{cases} 1 & \text{if } j = 1 \text{ and } \big(i = 1 \text{ for } K_{\ell_2} \text{ or } i = 2 \text{ for } K_{\mathrm{sqr}}\big), \\ -1 & \text{if } i = j \text{ and } \big(i > 1 \text{ for } K_{\ell_2} \text{ or } i > 2 \text{ for } K_{\mathrm{sqr}}\big), \\ 0 & \text{otherwise.} \end{cases} \tag{4.31}$$

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

$$g(s) = -2Js, \tag{4.32a}$$

$$H(s)\delta = 2J(2Jss^\top J\delta - J\delta), \tag{4.32b}$$

$$\mathrm{T}(s, \delta) = J(Js\delta^\top H\delta + H\delta s^\top J\delta - s^\top H\delta J\delta). \tag{4.32c}$$

These oracles are computed in order of d time. The Hessian oracle is computed in order of d^2 time as:

$$H(s) = 2J(2JJss^{>}J - J). \tag{4.33}$$

Chapter 5

Sum of squares generalizations for conic sets

This chapter is based on the submitted paper Kapelevich et al. [2021].

5.1 Introduction

The *sum of squares* (*SOS*) condition is commonly used as a tractable restriction of polynomial nonnegativity. While SOS programs have traditionally been formulated and solved using semidefinite programming (SDP), Papp and Yildiz [2019] recently demonstrated the effectiveness of a nonsymmetric interior point algorithm in solving SOS programs without SDP formulations. In this chapter, we focus on structured SOS constraints that can be modeled using more specialized cones. We describe and give barrier functions for three related cones for modeling functions of dense polynomials, which we hope will become useful modeling primitives.

The first is the cone of *SOS matrices*, which is described by Coey et al. [2021d, Section 3.13] without derivation. We show that this cone can be computationally favorable to equally low-dimensional SOS formulations. Characterizations of univariate SOS matrix cones in the context of optimization algorithms have previously been given by Genin et al. [2003, Section 6]. However, their use of monomial or Chebyshev bases complicates computations of oracles in an interior point algorithm [Papp and

Yildiz, 2019, Section 3.1] and prevents e□ective generalizations to the multivariate case.

The second is an *SOS ℓ_2-norm* (*SOS-L2*) cone, which can be used to certify pointwise membership in the second order cone for a vector with polynomial components. The third is an *SOS ℓ_1-norm* (*SOS-L1*) cone, which can be used to certify pointwise membership in the epigraph set of the ℓ_1-norm function. Although it is straightforward to use SOS representations to approximate these sets, such formulations introduce cones of higher dimension than the constrained polynomial vector. We believe we are first to describe how to handle these sets in an interior point algorithm without introducing auxiliary conic variables or constraints. We suggest new barriers, with lower barrier parameters than SOS formulations allow.

In the remainder of this section we provide background on SOS polynomials and implementation details of interior point algorithms that are required for later sections. In Section 5.2 we describe the constraints we wish to model using each new cone, and suggest alternative SOS formulations for comparison. In Section 5.3 we outline how ideas introduced by Papp and Alizadeh [2013] can be used to characterize the cone of SOS matrices and the SOS-L2 cone. Section 5.4 is focused on improving the parameter of the barriers for the SOS-L2 and SOS-L1 cones. In Section 5.5 we outline implementation advantages of the new cones. In Section 5.6 we compare various formulations using a numerical example and conclude in Section 5.7.

In this chapter, J$a..b$K are the integers in the interval $[a, b]$. *jAj* denotes the dimension of a set A. $\mathbf{I}_m$ is the identity in $\mathbb{R}^{m\times m}$. $\otimes_K : \mathbb{R}^{a_1\times a_2} \times \mathbb{R}^{b_1\times b_2} \to \mathbb{R}^{a_1b_1\times a_2b_2}$ is the usual Kronecker product. All vectors, matrices, and higher order tensors are written in bold font. s_i is the ith element of a vector $\mathbf{s}$ and recall that $\mathbf{s}_{i2\mathrm{J}1::N\mathrm{K}}$ is the product $(\mathbf{s}_1, \ldots, \mathbf{s}_N)$. If A is a vector space then A^n is the Cartesian product of n spaces A.

$\mathbb{R}[\mathbf{x}]_{n;d}$ is the ring of polynomials in the variables $\mathbf{x} = (x_1, \ldots, x_n)$ with maximum degree d. Following the notation of Papp and Yildiz [2019], we use $L = \binom{n+d}{n}$ and $U = \binom{n+2d}{n}$ to denote the dimensions of $\mathbb{R}[\mathbf{x}]_{n;d}$ and $\mathbb{R}[\mathbf{x}]_{n;2d}$ respectively, when n and d are given in the surrounding context.

5.1.1 The SOS polynomials cone and generic interior point algorithms

A polynomial $p(\mathbf{x})$ 2 $\mathrm{R}[\mathbf{x}]_{n;2d}$ is SOS if it can be expressed in the form $p(\mathbf{x}) = \sum_{i2\mathrm{J1::}N\mathrm{K}} q_i(\mathbf{x})^2$ for some N 2 N and $q_{i2\mathrm{J1::}N\mathrm{K}}(\mathbf{x})$ 2 $\mathrm{R}[\mathbf{x}]_{n;d}$. We denote the set of SOS polynomials in $\mathrm{R}[\mathbf{x}]_{n;2d}$ by K_{SOS}, which is a proper cone in $\mathrm{R}[\mathbf{x}]_{n;2d}$ [Nesterov, 2000].

We also say that $\mathbf{s}$ 2 K_{SOS} for $\mathbf{s}$ 2 R^U if $\mathbf{s}$ represents a vector of coe□cients of an SOS polynomial under a given basis. We use such *vectorized* definitions interchangeably with functional definitions of polynomial cones. To construct a vectorized definition for K_{SOS}, suppose we have a fixed basis for $\mathrm{R}[\mathbf{x}]_{n;2d}$, and let $p_{i2\mathrm{J1::}L\mathrm{K}}(\mathbf{x})$ be basis polynomials for $\mathrm{R}[\mathbf{x}]_{n;d}$. Let λ : J1..LK^2 ! R^U be a function such that $\lambda(i,j)$ returns the vector of coe□cients of the polynomial $p_i(\mathbf{x})p_j(\mathbf{x})$ using the fixed basis for $\mathrm{R}[\mathbf{x}]_{n;2d}$. Define the *lifting operator* : R^U ! S^L, introduced by Nesterov [2000], as:

$$(\mathbf{s})_{i;j} = h\lambda(i,j), \mathbf{s}i_{\mathrm{R}^U} \quad 8i,j \text{ 2 J1..}L\mathrm{K}, \tag{5.1}$$

where $(\mathbf{s})_{ij}$ is a component in row i and column j. Now the cones K_{SOS} and K^*_{SOS} admit the characterization [Nesterov, 2000, Theorem 7.1]:

$$K_{\mathrm{SOS}} = f\mathbf{s} \text{ 2 } \mathrm{R}^U : 9\mathbf{S} \text{ 2 } \mathrm{S}^L_+, \mathbf{s} = \ ^*(\mathbf{S})g, \tag{5.2a}$$

$$K^*_{\mathrm{SOS}} = f\mathbf{s} \text{ 2 } \mathrm{R}^U : \ (\mathbf{s}) \text{ 2 } \mathrm{S}^L_+g. \tag{5.2b}$$

(5.2) shows that the dual cone K^*_{SOS} is an inverse linear image of the positive semidefinite (PSD) cone, and therefore has an e□ciently computable *logarithmically homogeneous self-concordant barrier* (LHSCB) (see [Nesterov and Nemirovskii, 1994, Definitions 2.3.1, 2.3.2]). In particular, by linearity of , the function $\mathbf{s}$ 7! − logdet((s)) is an LHSCB for K^*_{SOS} [Nesterov and Nemirovskii, 1994, Proposition 5.1.1] with parameter L (an L-LHSCB for short). This makes it possible to solve optimization problems over K_{SOS} or K^*_{SOS} with a generic primal-dual interior point algorithm in polynomial time, for example, the algorithm from Chapter 2.[1]

[1] We direct the interested reader to Faybusovich [2002], who obtained non-linear barriers for the

Recall the algorithm from Chapter 2 only requires a membership check, an initial interior point, and evaluations of derivatives of an LHSCB for each cone *or* its dual. Optimizing over K_{SOS} (or K^*_{SOS}) directly instead of building SDP formulations is appealing because the dimension of K_{SOS} is generally much smaller than the cone dimension in SDP formulations that are amenable to more specialized algorithms [Papp and Yildiz, 2019, Coey et al., 2021d]. In later sections we describe e□cient LHSCBs and membership checks for each cone we introduce.

The output of the lifting operator depends on the polynomial basis chosen for $\mathrm{R}[\mathbf{x}]_{n;d}$ as well as the basis for $\mathrm{R}[\mathbf{x}]_{n;2d}$. Following Papp and Yildiz [2019], we use a set of Lagrange polynomials that are interpolant on some points $\mathbf{t}_{i\,2\,\mathrm{J1}::U\mathrm{K}}$ as the basis for $\mathrm{R}_{n;2d}[\mathbf{x}]$ and the multivariate Chebyshev polynomials [Ho□man and Withers, 1988] as the basis in $\mathrm{R}_{n;d}[\mathbf{x}]$. These choices give the particular lifting operator we implement, $_{\text{SOS}}(\mathbf{s})$:

$$_{\text{SOS}}(\mathbf{s})_{i;j} = \sum_{u\,2\,\mathrm{J1}::U\mathrm{K}} p_i(\mathbf{t}_u)p_j(\mathbf{t}_u)s_u \quad 8i, j \;2\; \mathrm{J1}..L\mathrm{K}. \tag{5.3}$$

Equivalently, $_{\text{SOS}}(\mathbf{s}) = \mathbf{P}^{>}\,\text{Diag}(\mathbf{s})\mathbf{P}$, where $P_{u;\cdot} = p_{\cdot}(\mathbf{t}_u)$ for all $u \;2\; \mathrm{J1}..U\mathrm{K}, \ell \;2\; \mathrm{J1}..L\mathrm{K}$. The adjoint $^{*}_{\text{SOS}} : \mathrm{S}^{L} \;!\; \mathrm{R}^{U}$ is given by $^{*}_{\text{SOS}}(\mathbf{S}) = \text{diag}(\mathbf{PSP}^{>})$. Papp and Yildiz [2019] show that the Lagrange basis gives rise to expressions for the gradient and Hessian of the barrier for K^*_{SOS} that are computable in $O(LU^2)$ time for any $d, n \geq 1$. Although we assume for simplicity that p is a dense basis for $\mathrm{R}[\mathbf{x}]_{n;d}$, this is without loss of generality. A modeler with access to a suitable sparse basis of $L < L$ polynomials in $\mathrm{R}[\mathbf{x}]_{n;d}$ and $U < U$ interpolation points, could use (5.3) and obtain a barrier with parameter L.

cone of univariate polynomials generated by Chebyshev systems by computing the universal volume barrier of Nesterov and Nemirovskii [1994], which is unrelated to the SDP representations of these polynomials.

5.2 Polynomial generalizations for three conic sets

The first set we consider are the polynomial matrices $\mathbf{Q}(\mathbf{x}) \in \mathrm{R}[\mathbf{x}]^{m\times m}_{n;2d}$ (i.e. $m \times m$ matrices with components that are polynomials in n variables of maximum degree $2d$)[2] satisfying the constraint:

$$\mathbf{Q}(\mathbf{x}) \succeq 0 \quad \forall \mathbf{x}. \tag{5.4}$$

One of the first applications of matrix SOS constraints was by Henrion and Lasserre [2006]. The moment-SOS hierarchy was extended from the scalar case to the matrix case, using a suitable extension of Putinar's Positivstellesatz studied by Hol and Scherer [2004] and Kojima [2003]. This constraint has various applications in statistics, control, and engineering [Aylward et al., 2007, 2008, Doherty et al., 2004, Hall, 2019]. A tractable restriction for (5.4) is given by the SOS formulation:

$$\mathbf{y}^\top \mathbf{Q}(\mathbf{x})\mathbf{y} \in K_{\mathrm{SOS}} \quad \forall \mathbf{y} \in \mathrm{R}^m. \tag{5.5}$$

This formulation is sometimes implemented in practice (e.g. [Legat et al., 2017]) and requires an SOS cone of dimension $U\,\mathrm{sd}(m)$ (by exploiting the fact that all terms are bilinear in the $\mathbf{y}$ variables). It is well known that (5.5) is equivalent to restricting $\mathbf{Q}(\mathbf{x})$ to be an *SOS matrix* of the form $\mathbf{Q}(\mathbf{x}) = \mathbf{M}(\mathbf{x})^\top\mathbf{M}(\mathbf{x})$ for some $N \in \mathrm{N}$ and $\mathbf{M}(\mathbf{x}) \in \mathrm{R}[\mathbf{x}]^{N\times m}_{n;d}$ [Blekherman et al., 2012, Definition 3.76]. To be consistent in terminology with the other cones we introduce, we refer to SOS matrices as *SOS-PSD* matrices, or belonging to K_{SOSPSD}. We show how to characterize K_{SOSPSD} and use it directly in an interior point algorithm in Section 5.3.

The second set we consider are the polynomial vectors $\mathbf{q}(\mathbf{x}) \in \mathrm{R}[\mathbf{x}]^m_{n;2d}$ satisfying:

$$q_1(\mathbf{x}) \geq \sqrt{\textstyle\sum_{i=2,3,\ldots,m} (q_i(\mathbf{x}))^2} \quad \forall \mathbf{x}, \tag{5.6}$$

[2] We assume that polynomial components in vectors and matrices involve the same variables and have the same maximum degree, to avoid detracting from the key ideas in this paper. This assumption could be removed at the expense of more cumbersome notation.

and hence requiring $\mathbf{q}(\mathbf{x})$ to be in the epigraph set of the ℓ_2-norm function (second order cone) pointwise (cf. (5.4) requiring the polynomial matrix to be in the PSD cone). A tractable restriction for this constraint is given by the SOS formulation:

$$\mathbf{y}^\top \mathrm{Arw}(\mathbf{q}(\mathbf{x}))\mathbf{y} \in K_{\mathrm{SOS}} \quad \forall \mathbf{y} \in \mathbb{R}^m, \tag{5.7}$$

where $\mathrm{Arw} : \mathbb{R}[\mathbf{x}]^m_{n;2d} \to \mathbb{R}[\mathbf{x}]^{m\times m}_{n;2d}$ is defined by:

$$\mathrm{Arw}(\mathbf{p}(\mathbf{x})) = \begin{bmatrix} p_1(\mathbf{x}) & \mathbf{p}(\mathbf{x})^\top \\ \mathbf{p}(\mathbf{x}) & p_1(\mathbf{x})\mathbf{I}_{m-1} \end{bmatrix}, \tag{5.8}$$
$$\mathbf{p}(\mathbf{x}) = (p_1(\mathbf{x}), \mathbf{p}(\mathbf{x})) \in \mathbb{R}[\mathbf{x}]_{n;2d} \times \mathbb{R}[\mathbf{x}]^{m-1}_{n;2d}.$$

Due to the equivalence between (5.5) and membership in K_{SOSPSD}, (5.7) is equivalent to requiring that $\mathbf{q}(\mathbf{x})$ belongs to the cone we denote $K_{\mathrm{ArwSOSPSD}}$ defined by:

$$K_{\mathrm{ArwSOSPSD}} = \{\mathbf{q}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]^m_{n;2d} : \mathrm{Arw}(\mathbf{q}(\mathbf{x})) \in K_{\mathrm{SOSPSD}}\}. \tag{5.9}$$

Membership in $K_{\mathrm{ArwSOSPSD}}$ ensures (5.6) holds due to the SDP representation of the second order cone [Alizadeh and Goldfarb, 2003], and the fact that the SOS-PSD condition certifies pointwise positive semidefiniteness. An alternative restriction of (5.6) is described by the set we denote $K_{\mathrm{SOS}\text{`}_2}$, which is not representable by the usual scalar polynomial SOS cone in general:

$$K_{\mathrm{SOS}\text{`}_2} = \left\{ \begin{array}{r} \mathbf{q}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]^m_{n;2d} : \exists N \in \mathbb{N}, \mathbf{p}_{i\in\{1,\ldots,N\}}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]^m_{n;d}, \\ \mathbf{q}(\mathbf{x}) = \sum_{i\in\{1,\ldots,N\}} \mathbf{p}_i(\mathbf{x}) \quad \mathbf{p}_i(\mathbf{x}) \end{array} \right\}, \tag{5.10}$$

where $\quad : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ is defined by:

$$\mathbf{x} \quad \mathbf{y} = \begin{bmatrix} \mathbf{x}^\top \mathbf{y} \\ x_1\mathbf{y} + y_1\mathbf{x} \end{bmatrix}, \quad \mathbf{x} = (x_1, \mathbf{x}), \quad \mathbf{y} = (y_1, \mathbf{y}) \in \mathbb{R} \times \mathbb{R}^{m-1}, \tag{5.11}$$

and $\quad : \mathbb{R}[\mathbf{x}]^m_{n;d} \times \mathbb{R}[\mathbf{x}]^m_{n;d} \to \mathbb{R}[\mathbf{x}]^m_{n;2d}$ on polynomial vectors is defined analogously.

This set was also studied by Kojima and Muramatsu with a focus on extending Positivstellensatz results [Kojima and Muramatsu, 2007]. The validity of $K_{\text{SOS}^{\backprime}_2}$ as a restriction of (5.6) follows from the the characterization of the second order cone as a *cone of squares* [Alizadeh and Goldfarb, 2003, Section 4]. For this reason we will refer to the elements of $K_{\text{SOS}^{\backprime}_2}$ as the *SOS-L*2 polynomials. For a polynomial vector in $\mathsf{R}[\mathbf{x}]^m_{n;2d}$, the dimension of $K_{\text{SOS}^{\backprime}_2}$ is Um, which is favorable to the dimension $U\,\text{sd}(m)$ of K_{SOS} required for (5.7) or K_{SOSPSD} in (5.9). In addition, we show in Section 5.4.1 that $K_{\text{SOS}^{\backprime}_2}$ admits an LHSCB with smaller parameter than $K_{\text{Arw SOSPSD}}$. However, we conjecture that for general n and d, $K_{\text{SOS}^{\backprime}_2}$ ($K_{\text{Arw SOSPSD}}$ (for example, consider the vector $[1 + x^2, 1 - x^2, 2x]$, which belongs to $K_{\text{Arw SOSPSD}}$ but not $K_{\text{SOS}^{\backprime}_2}$). Our experiments in Section 5.6 also include instances where using $K_{\text{SOS}^{\backprime}_2}$ and $K_{\text{Arw SOSPSD}}$ gives di□erent objective values. A third formulation can be obtained by modifying the SDP formulation for $K_{\text{Arw SOSPSD}}$ to account for all sparsity in the $\mathbf{y}$ monomials (by introducing a specialized cone for the Gram matrix of $\mathbf{y}^{>}\,\text{Arw}(\mathbf{q}(\mathbf{x}))\mathbf{y}$). However, this approach su□ers from requiring $O(L^2)$ conic variables for each polynomial in $\mathbf{q}(\mathbf{x})$, so we choose to focus on K_{SOS} and K_{SOSPSD} formulations for $K_{\text{Arw SOSPSD}}$ instead.

The third and final set we consider is also described through a constraint on a polynomial vector $\mathbf{q}(\mathbf{x})$ *2* $\mathsf{R}[\mathbf{x}]^m_{n;2d}$. This constraint is given by:

$$q_1(\mathbf{x}) \geq \sum_{i2\mathsf{J}2::m\mathsf{K}} jq_i(\mathbf{x})j \quad 8\mathbf{x}, \tag{5.12}$$

and hence requires the polynomial vector to be in the epigraph set of the ℓ_1-norm function (ℓ_1-*norm cone*) pointwise. A tractable restriction for this constraint is given by the SOS formulation:

$$q_1(\mathbf{x}) - \sum_{i2\mathsf{J}2::m\mathsf{K}}(p_i(\mathbf{x})^+ + p_i(\mathbf{x})^-) \; 2 \; K_{\text{SOS}}, \tag{5.13a}$$

$$q_i(\mathbf{x}) = p_i(\mathbf{x})^+ - p_i(\mathbf{x})^- \qquad 8i \; 2 \; \mathsf{J}2..m\mathsf{K}, \tag{5.13b}$$

$$p_i(\mathbf{x})^+, p_i(\mathbf{x})^- \; 2 \; K_{\text{SOS}} \qquad 8i \; 2 \; \mathsf{J}2..m\mathsf{K}, \tag{5.13c}$$

which uses auxiliary polynomial variables $p^+_{i2\mathsf{J}2::m\mathsf{K}}(\mathbf{x})$ *2* $\mathsf{R}[\mathbf{x}]_{n;2d}$ and $p^-_{i2\mathsf{J}2::m\mathsf{K}}(\mathbf{x})$ *2*

$\mathrm{R}[\mathbf{x}]_{n;2d}$. We refer to the projection of (5.13) onto $\mathbf{q}(\mathbf{x})$ *2* $\mathrm{R}[\mathbf{x}]^m_{n;2d}$ as $K_{\mathrm{SOS}\text{-}1}$ and to its elements as the *SOS-L*1 polynomials. Note that the dimension of $K_{\mathrm{SOS}\text{-}1}$ is Um, while (5.13) requires $2m - 1$ SOS cones of dimension U and $U(m - 1)$ additional equality constraints. In Section 5.4.2 we derive an Lm-LHSCB that allows us to optimize over $K_{\mathrm{SOS}\text{-}1}$ directly, while (5.13) would require an LHSCB with parameter $L(2m - 1)$.

We summarize some key properties of the new cones and SOS formulations in Table 5.1: the total dimension of cones involved, the parameter of an LHSCB for the conic sets, the time complexity to calculate the Hessian of the LHSCB (discussed in Section 5.5), the level of conservatism of each new conic set compared to its alternative SOS formulation, and the number of auxiliary equality constraints and variables that need to be added in an optimization problem.

	SOS-PSD		SOS-L2		SOS-L1	
	K_{SOSPSD}	(5.5)	$K_{\mathrm{SOS}\text{-}2}$	(5.7)	$K_{\mathrm{SOS}\text{-}1}$	(5.13)
cone dim.	$U\,\mathrm{sd}(m)$	$U\,\mathrm{sd}(m)$	Um	$U\,\mathrm{sd}(m)$	Um	$U(2m - 1)$
parameter	Lm	Lm	$2L$	Lm	Lm	$L(2m - 1)$
Hessian	LU^2m^3	LU^2m^5	LU^2m^2	LU^2m^5	LU^2m	LU^2m
conservatism	equal	-	greater	-	equal	-
equalities	0	0	0	0	0	$U(m - 1)$
variables	0	0	0	0	0	$2U(m - 1)$

Table 5.1: Properties of new cones compared to SOS formulations.

5.3 SOS-PSD and SOS-L2 cones from general algebras

The ideas introduced by Papp and Alizadeh [2013] relating to SOS cones in *general algebras* allow us to characterize K_{SOSPSD} and $K_{\mathrm{SOS}\text{-}2}$ without auxiliary SOS polynomial constraints. As in Papp and Alizadeh [2013], let us define $(A, B, \ )$ as a general algebra if A, B are vector spaces and $\ : A \times A \ ! \ B$ is a bilinear product that satisfies the distributive property. For a general algebra $(A, B, \ )$, Papp and Alizadeh [2013]

define the SOS cone K :

$$K = \{b \in B : \exists N \in \mathbb{N}, a_{i \in \{1,\ldots,N\}} \in A, b = \sum_{i \in \{1,\ldots,N\}} a_i \otimes a_i\}. \tag{5.14}$$

For instance, $\mathbb{S}_+$ is equal to the SOS cone of $(\mathbb{R}^m, \mathbb{S}^m, \bar{\otimes})$ for $\bar{\otimes}$ given by $\mathbf{x} \bar{\otimes} \mathbf{y} = \frac{1}{2}(\mathbf{x}\mathbf{y}^\top + \mathbf{y}\mathbf{x}^\top)$. The second order cone is equal to the SOS cone of $(\mathbb{R}^m, \mathbb{R}^m, \otimes)$. K_{SOS} is equal to the SOS cone of $(\mathbb{R}[\mathbf{x}]_{n;d}, \mathbb{R}[\mathbf{x}]_{n;2d}, \cdot)$ where $\cdot$ is the product of polynomials. To obtain our vectorized representation of K_{SOS} we can redefine the function $\lambda : \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}^U$ so that for $\mathbf{p}_i, \mathbf{p}_j \in \mathbb{R}^L$ representing coefficients of any polynomials in $\mathbb{R}[\mathbf{x}]_{n;d}$, $\lambda(\mathbf{p}_i, \mathbf{p}_j)$ returns the vector of coefficients of the product of the polynomials. Then K_{SOS} is equal to the SOS cone of $(\mathbb{R}^L, \mathbb{R}^U, \lambda)$.

As we describe in Section 5.3.1, Papp and Alizadeh [2013] also show how to build lifting operators for general algebras. This allows us to construct membership checks and easily computable LHSCBs for K^*_{SOSPSD} and $K^*_{\text{SOS}\grave{}_2}$ once we represent them as SOS cones of *tensor products* of algebras.

The tensor product of two algebras $(A_1, B_1, \otimes_1)$ and $(A_2, B_2, \otimes_2)$ is a new algebra $(A_1 \otimes A_2, B_1 \otimes B_2, \otimes_1 \otimes \otimes_2)$, where $\otimes_1 \otimes \otimes_2$ is defined via its action on *elementary tensors*. For $\mathbf{u}_1, \mathbf{v}_1 \in A_1$ and $\mathbf{u}_2, \mathbf{v}_2 \in A_2$:

$$(\mathbf{u}_1 \otimes \mathbf{u}_2) \otimes_1 \otimes \otimes_2 (\mathbf{v}_1 \otimes \mathbf{v}_2) = (\mathbf{u}_1 \otimes_1 \mathbf{v}_1) \otimes (\mathbf{u}_2 \otimes_2 \mathbf{v}_2). \tag{5.15}$$

The algebra we are interested in for a functional representation of K_{SOSPSD} is the tensor product of $(\mathbb{R}[\mathbf{x}]_{n;d}, \mathbb{R}[\mathbf{x}]_{n;2d}, \cdot)$ with $(\mathbb{R}^m, \mathbb{S}^m, \otimes)$. We can think of elements in $\mathbb{R}[\mathbf{x}]_{n;d} \otimes \mathbb{R}^m$ as polynomial vectors in $\mathbb{R}[\mathbf{x}]^m_{n;d}$, and $\mathbb{R}[\mathbf{x}]_{n;2d} \otimes \mathbb{S}^m$ as the symmetric polynomial matrices in $\mathbb{R}[\mathbf{x}]^{m \times m}_{n;2d}$. The SOS cone of $(\mathbb{R}[\mathbf{x}]_{n;d} \otimes \mathbb{R}^m, \mathbb{R}[\mathbf{x}]_{n;2d} \otimes \mathbb{S}^m, \cdot \otimes)$ corresponds to the polynomial matrices that can be written as $\sum_{i \in \{1,\ldots,N\}} \mathbf{m}_i(\mathbf{x})\mathbf{m}_i(\mathbf{x})^\top$ with $\mathbf{m}_i(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]^m$ for all $i \in \{1..N\}$ [Papp and Alizadeh, 2013, Section 4.3], which is exactly K_{SOSPSD}. Equivalently, a vectorized representation of K_{SOSPSD} can be characterized as the SOS cone of $(\mathbb{R}^L \otimes \mathbb{R}^m, \mathbb{R}^U \otimes \mathbb{S}^m, \lambda \otimes)$. We can think of $\mathbb{R}^L \otimes \mathbb{R}^m$ as $\mathbb{R}^{L \times m}$ and we can think of $\mathbb{R}^U \otimes \mathbb{S}^m$ as a subspace of $\mathbb{R}^{U \times m \times m}$ that represents the coefficients of symmetric polynomial matrices.

Likewise, the algebra we are interested in for a functional representation of $K_{\text{SOS}`_2}$ is the tensor product of $(\mathrm{R}[\mathbf{x}]_{n;d}, \mathrm{R}[\mathbf{x}]_{n;2d}, \cdot)$ with $(\mathrm{R}^m, \mathrm{R}^m, \ )$. We can think of $\mathrm{R}[\mathbf{x}]_{n;d}$ R^m and $\mathrm{R}[\mathbf{x}]_{n;2d}$ R^m as $\mathrm{R}[\mathbf{x}]^m_{n;d}$ and $\mathrm{R}[\mathbf{x}]^m_{n;2d}$ respectively. The SOS cone of the tensor product of these algebras then corresponds to $K_{\text{SOS}`_2}$ due to (5.10). A vectorized representation of $K_{\text{SOS}`_2}$ may be characterized as the SOS cone of $(\mathrm{R}^L \ \mathrm{R}^m, \mathrm{R}^U \ \mathrm{R}^m, \lambda \ )$. We can think of R^U R^m as the coe□cients of polynomial vectors, represented in $\mathrm{R}^{U\times m}$.

5.3.1 Lifting operators for SOS-PSD and SOS-L2

The lifting operator of $(A, B, \ )$, when A and B are finite dimensional, is defined by Papp and Alizadeh [2013] as the function $\ : B \ ! \ \mathrm{S}^{jAj}$ satisfying $hb, a_1 \ (b)a_2 i_A = hb, a_1 \ a_2 i_B$ for all $a_1, a_2 \ 2 \ A$, $b \ 2 \ B$. This leads to the following descriptions of K and K^* [Papp and Alizadeh, 2013, Theorem 3.2]:

$$K \ = \ f\mathbf{s} \ 2 \ B : 9\mathbf{S} \ \ 0, \mathbf{s} = \ ^*(\mathbf{S})g, \tag{5.16a}$$

$$K^* = f\mathbf{s} \ 2 \ B : \ \ (\mathbf{s}) \ \ 0g. \tag{5.16b}$$

Recall that in order to use either K or K^* in a generic interior point algorithm, we require e□cient oracles for a membership check and derivatives of an LHSCB of K or K^*. If $(\mathbf{s})$ is e□ciently computable, (5.16b) provides a membership check for K^*. Furthermore, an LHSCB for K^* is given by $\mathbf{s} \ 7! \ -\log\det(\ (\mathbf{s}))$ with barrier parameter jAj due to the linearity of [Nesterov and Nemirovskii, 1994, Proposition 5.1.1]. The following lemma describes how to compute $(\mathbf{s})$ for a tensor product algebra.

Lemma 5.3.1. *[Papp and Alizadeh, 2013, Lemma 4.1]: If* $\mathbf{w}_1 \ 2 \ B_1$ *and* $\mathbf{w}_2 \ 2 \ B_2$, *then:*

$$_1 \ _2(\mathbf{w}_1 \ \mathbf{w}_2) = \ _1(\mathbf{w}_1) \ _K \ _2(\mathbf{w}_2). \tag{5.17}$$

Let us define $\quad : \mathbb{R}^U \times \mathbb{S}^m \to \mathbb{R}^{U\times m\times m}$ such that $(\mathbf{u} \quad \mathbf{V})_{i;j;k} = u_i V_{j;k}$ and let us represent the coefficients of a polynomial matrix by a tensor $\mathbf{S} \in \mathbb{R}^{U\times m\times m}$. Then we may write $\mathbf{S} = \sum_{i\in\{1::m\},j\in\{1::i\}} \mathbf{S}_{i;j} \quad \mathbf{E}_{i;j}$, where $\mathbf{E}_{i;j} \in \mathbb{R}^{m\times m}$ is a matrix of zeros and ones with $E_{i;j} = E_{j;i} = 1$ and $\mathbf{S}_{i;j} \in \mathbb{R}^U$ are the coefficients of the polynomial in row i and column j. Applying Lemma 5.3.1, the lifting operator for K_{SOSPSD}, $_{\text{SOSPSD}} : \mathbb{R}^{U\times m\times m} \to \mathbb{S}^{Lm}$ is:

$$_{\text{SOSPSD}}(\mathbf{S}) = \quad ^{-}(\mathbf{S}) = \quad ^{-}(\textstyle\sum_{i\in\{1::m\},j\in\{1::i\}} \mathbf{S}_{i;j} \quad \mathbf{E}_{i;j}) \tag{5.18a}$$

$$= \textstyle\sum_{i\in\{1::m\},j\in\{1::i\}} \quad _{\text{SOS}}(\mathbf{S}_{i;j}) \quad _K \mathbf{E}_{i;j}. \tag{5.18b}$$

The output is a block matrix, where each $L \times L$ submatrix in the ith group of rows and jth group of columns is $\quad _{\text{SOSPSD}}(\mathbf{S})_{i;j} = \quad _{\text{SOS}}(\mathbf{S}_{i;j})$ for all $i, j \in \{1..m\}$. The adjoint operator $\quad ^*_{\text{SOSPSD}} : \mathbb{S}^{Lm} \to \mathbb{R}^{U\times m\times m}$ may also be defined blockwise, $\quad ^*_{\text{SOSPSD}}(\mathbf{S})_{i;j} = \quad ^*_{\text{SOS}}(\mathbf{S}_{i;j})$ for all $i, j \in \{1..m\}$ where $\mathbf{S}_{i;j} \in \mathbb{R}^{L\times L}$ is the (i, j)th submatrix in $\mathbf{S}$.

Likewise, we use a tensor $\mathbf{s} \in \mathbb{R}^{U\times m}$ to describe the coefficients of a polynomial vector, and write $\mathbf{s}_i \in \mathbb{R}^U$ to denote the vector of coefficients of the polynomial in component i. Applying Lemma 5.3.1 again, we obtain the (blockwise) definition of the lifting operator for $K_{\text{SOS}`_2}$, $\quad _{\text{SOS}`_2} : \mathbb{R}^{U\times m} \to \mathbb{S}^{Lm}$:

$$_{\text{SOS}`_2}(\mathbf{s})_{i;j} = \begin{cases} _{\text{SOS}}(\mathbf{s}_1) & i = j \\ _{\text{SOS}}(\mathbf{s}_j) & i = 1, j \neq 1 \\ _{\text{SOS}}(\mathbf{s}_i) & i \neq 1, j = 1 \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j \in \{1..m\}, \tag{5.19}$$

where $\quad _{\text{SOS}`_2}(\mathbf{s})_{i;j} \in \mathbb{S}^L$ is the $(i, j)$th submatrix of $\quad _{\text{SOS}`_2}(\mathbf{s})$. Thus $\quad _{\text{SOS}`_2}(\mathbf{s})$ has a block arrowhead structure. The output of the adjoint operator $\quad ^*_{\text{SOS}`_2} : \mathbb{S}^{Lm} \to \mathbb{R}^{U\times m}$

may be defined as:

$$\overset{*}{\mathrm{SOS}}{}_{\backslash 2}(\mathbf{S})_i = \begin{cases} \sum_{j \in \{1:m\}} \overset{*}{\mathrm{SOS}}(\mathbf{S}_{j,j}) & i = 1 \\ \overset{*}{\mathrm{SOS}}(\mathbf{S}_{1,i}) + \overset{*}{\mathrm{SOS}}(\mathbf{S}_{i,1}) & i \neq 1 \end{cases} \quad \forall i \in \{1..m\}, \tag{5.20}$$

where $\overset{*}{\mathrm{SOS}}{}_{\backslash 2}(\mathbf{S})_i \in \mathbb{R}^U$ is the ith slice of $\overset{*}{\mathrm{SOS}}{}_{\backslash 2}(\mathbf{S})$ and $\mathbf{S}_{i,j} \in \mathbb{R}^{L \times L}$ is the (i,j)th block in $\mathbf{S}$ for all $i, j \in \{1..m\}$.

5.4 Efficient barriers for SOS-L2 and SOS-L1

As for K^*_{SOSPSD} and $K^*_{\mathrm{SOS}\backslash_2}$, we show that a barrier for $K^*_{\mathrm{SOS}\backslash_1}$ can be obtained by composing a linear lifting operator with the logdet barrier. This is sufficient to optimize over K_{SOSPSD}, $K_{\mathrm{SOS}\backslash_2}$ and $K_{\mathrm{SOS}\backslash_1}$ without high dimensional SDP formulations. However, for $K^*_{\mathrm{SOS}\backslash_2}$ and $K^*_{\mathrm{SOS}\backslash_1}$ we can derive improved barriers by composing nonlinear functions with the logdet barrier instead. We show that these compositions are indeed LHSCBs.

5.4.1 SOS-L2

Recall (5.16b) suggests that checking membership in $K^*_{\mathrm{SOS}\backslash_2}$ amounts to checking positive definiteness of $\mathrm{SOS}\backslash_2(\mathbf{s})$ with side dimension Lm. This membership check corresponds to a *straightforward* LHSCB with parameter Lm given by $\mathbf{s} \mapsto -\log\det(\mathrm{SOS}\backslash_2(\mathbf{s}))$. We now show that by working with a Schur complement of $\mathrm{SOS}\backslash_2(\mathbf{s})$, we obtain a membership check for $K^*_{\mathrm{SOS}\backslash_2}$ that requires factorizations of only two matrices with side dimension L and implies an LHSCB with parameter $2L$.

Let $\mathbb{R}^{U \times m} \to \mathbb{S}^L$ return the Schur complement:

$$(\mathbf{s}) = \mathrm{SOS}(\mathbf{s}_1) - \sum_{i \in \{2:m\}} \mathrm{SOS}(\mathbf{s}_i)\ \mathrm{SOS}(\mathbf{s}_1)^{-1}\ \mathrm{SOS}(\mathbf{s}_i). \tag{5.21}$$

By (5.16b) and (5.21):

$$K^*_{\mathrm{SOS}`_2} = \{\mathbf{s} \in \mathbb{R}^{U\times m} : \ _{\mathrm{SOS}`_2}(\mathbf{s}) \succeq 0\} \tag{5.22a}$$

$$= \mathrm{cl}\{\mathbf{s} \in \mathbb{R}^{U\times m} : \ _{\mathrm{SOS}`_2}(\mathbf{s}) \succ 0\} \tag{5.22b}$$

$$= \mathrm{cl}\{\mathbf{s} \in \mathbb{R}^{U\times m} : \ _{\mathrm{SOS}}(\mathbf{s}_1) \succeq 0, \ (\mathbf{s}) \succeq 0\}. \tag{5.22c}$$

(5.22c) describes a simple membership check. Furthermore, the function $F : \mathbb{R}^{U\times m} \to \mathbb{R}$ defined by:

$$F(\mathbf{s}) = -\log\det(\ (\mathbf{s})) - \log\det(\ _{\mathrm{SOS}}(\mathbf{s}_1)) \tag{5.23a}$$

$$= -\log\det(\ _{\mathrm{SOS}`_2}(\mathbf{s})) + (m-2)\log\det(\ _{\mathrm{SOS}}(\mathbf{s}_1)), \tag{5.23b}$$

is a $2L$-LHSCB barrier for $K_{\mathrm{SOS}`_2}$.

Theorem 5.4.1. *The function F defined by* (5.23) *is a $2L$-LHSCB for $K^*_{\mathrm{SOS}`_2}$.*

Proof. It is easy to verify that F is a logarithmically homogeneous barrier, so we show it is a $2L$-self-concordant barrier for $K^*_{\mathrm{SOS}`_2}$. We first show that $\hat{F} : \mathbb{S}^L_{++} \times (\mathbb{R}^{L\times L})^{m-1} \to \mathbb{R}$ defined as $\hat{F}(\mathbf{X}_1, \ldots, \mathbf{X}_m) = -\log\det(\mathbf{X}_1 - \sum_{i\in\{2,\ldots,m\}} \mathbf{X}_i\mathbf{X}_1^{-1}\mathbf{X}_i^\top) - \log\det(\mathbf{X}_1)$, is a $2L$-self-concordant barrier for the cone:

$$K^m_{`_2} = \mathrm{cl}\big\{(\mathbf{X}_1, \ldots, \mathbf{X}_m) \in \mathbb{S}^L_{++} \times (\mathbb{R}^{L\times L})^{m-1} : \mathbf{X}_1 - \textstyle\sum_{i\in\{2,\ldots,m\}} \mathbf{X}_i\mathbf{X}_1^{-1}\mathbf{X}_i^\top \succeq 0\big\}. \tag{5.24}$$

We then argue that F is a composition of $\hat{F}$ with the linear map $(\mathbf{s}_1, \ldots, \mathbf{s}_m) \mapsto (\ _{\mathrm{SOS}}(\mathbf{s}_1), \ldots, \ _{\mathrm{SOS}}(\mathbf{s}_m))$ and $K^*_{\mathrm{SOS}`_2}$ is an inverse image of $K^m_{`_2}$ under the same map. Then by Nesterov and Nemirovskii [1994, Proposition 5.1.1] F is self-concordant.

Let $\mathcal{L} = \mathbb{S}^L_+ \times (\mathbb{R}^{L\times L})^{m-1}$ and $\mathbf{G} : \mathrm{int}(\mathcal{L}) \to \mathbb{S}^L$ be defined as:

$$\mathbf{G}(\mathbf{X}_1, \ldots, \mathbf{X}_m) = \mathbf{X}_1 - \textstyle\sum_{i\in\{2,\ldots,m\}} \mathbf{X}_i\mathbf{X}_1^{-1}\mathbf{X}_i^\top. \tag{5.25}$$

Let us check that $\mathbf{G}$ is $(\mathbb{S}^L_+, 1)$-compatible with the domain $\mathcal{L}$ in the sense of [Nesterov and Nemirovskii, 1994, Definition 5.1.1]. This requires that $\mathbf{G}$ is C^3-smooth on $\mathrm{int}(\mathcal{L})$,

$\mathbf{G}$ is concave with respect to $\mathbb{S}^L_+$, and at each point $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_m)$ *2* int( ) and any direction $\mathbf{V} = (\mathbf{V}_1, \ldots, \mathbf{V}_m)$ *2* $\mathbb{S}^L \times (\mathbb{R}^{L \times L})^{m-1}$ such that $-\mathbf{X}_1 \quad \mathbf{V}_1 \quad \mathbf{X}_1$, the directional derivatives of $\mathbf{G}$ satisfy:

$$\frac{d^3\mathbf{G}}{d\mathbf{X}^3}[\mathbf{V}, \mathbf{V}, \mathbf{V}] \quad -3\frac{d^2\mathbf{G}}{d\mathbf{X}^2}[\mathbf{V}, \mathbf{V}]. \tag{5.26}$$

Let $\mathbf{V}$ *2* $\mathbb{S}^L \times (\mathbb{R}^{L \times L})^{m-1}$. It can be checked that $\frac{d^3\mathbf{G}}{d\mathbf{X}^3}$ is continuous on the domain of $\mathbf{G}$ and we have the directional derivatives:

$$\frac{d^2\mathbf{G}}{d\mathbf{X}^2}[\mathbf{V}, \mathbf{V}] = -2\sum_{i2\mathrm{J}2::m\mathrm{K}}(\mathbf{X}_i\mathbf{X}_1^{-1}\mathbf{V}_1 - \mathbf{V}_i)\mathbf{X}_1^{-1}(\mathbf{X}_i\mathbf{X}_1^{-1}\mathbf{V}_1 - \mathbf{V}_i)^{>}, \tag{5.27}$$

$$\frac{d^3\mathbf{G}}{d\mathbf{X}^3}[\mathbf{V}, \mathbf{V}, \mathbf{V}] = 6\sum_{i2\mathrm{J}2::m\mathrm{K}}(\mathbf{X}_i\mathbf{X}_1^{-1}\mathbf{V}_1 - \mathbf{V}_i)\mathbf{X}_1^{-1}\mathbf{V}_1\mathbf{X}_1^{-1}(\mathbf{X}_i\mathbf{X}_1^{-1}\mathbf{V}_1 - \mathbf{V}_i)^{>}. \tag{5.28}$$

Since $\mathbf{X}_1 \quad 0$ in int( ), $-\frac{d^2\mathbf{G}}{d\mathbf{X}^2}[\mathbf{V}, \mathbf{V}] \quad 0$ and so by Nesterov and Nemirovskii [1994, Lemma 5.1.2], $\mathbf{G}$ is concave with respect to $\mathbb{S}^L_+$. It remains to show that (5.26) is satisfied. Since the directional derivatives decouple by each index i in the sum, it is su□cient to show that the inequality is satisfied for each i *2* J2..mK. For this, it is su□cient that:

$$6\mathbf{X}_1^{-1}\mathbf{V}_1\mathbf{X}_1^{-1} \quad -3 \times -2\mathbf{X}_1^{-1}, \tag{5.29}$$

for all $-\mathbf{X}_1 \quad \mathbf{V}_1 \quad \mathbf{X}_1$, which follows since $\mathbf{X}_1$ is positive definite on int( ). Now by [Nesterov and Nemirovskii, 1994, proposition 5.1.7], $\hat{F}$ is a $2L$-LHSCB. The same is true for F by composing $\hat{F}$ with a linear map. □ □

5.4.2 SOS-L1

By combining (5.2) and (5.13), the $K_{\text{SOS}`_1}$ cone admits the semidefinite representation:

$$K_{\text{SOS}`_1} = \left\{ \begin{array}{c} \mathbf{s} \in \mathbb{R}^{U \times m} : \exists \mathbf{S}_1, \mathbf{S}_{2;+}, \mathbf{S}_{2;-} \ldots, \mathbf{S}_{m;+}, \mathbf{S}_{m;-} \in \mathbb{S}^L_+, \\ \mathbf{s}_1 = {}^{*}_{\text{SOS}}(\mathbf{S}_1) + \sum_{i=2:m} {}^{*}_{\text{SOS}}(\mathbf{S}_{i;+} + \mathbf{S}_{i;-}), \\ \mathbf{s}_i = {}^{*}_{\text{SOS}}(\mathbf{S}_{i;+}) - {}^{*}_{\text{SOS}}(\mathbf{S}_{i;-}) \; \forall i \in 2:m \end{array} \right\}. \tag{5.30}$$

Its dual cone is:

$$K^*_{\text{SOS}`_1} = \left\{ \mathbf{s} \in \mathbb{R}^{U \times m} : \; {}_{\text{SOS}}(\mathbf{s}_1 + \mathbf{s}_i) \succeq 0, \; {}_{\text{SOS}}(\mathbf{s}_1 - \mathbf{s}_i) \succeq 0 \; \forall i \in 2:m \right\}. \tag{5.31}$$

(5.31) suggests that checking membership in $K^*_{\text{SOS}`_1}$ amounts to checking positive definiteness of $2(m-1)$ matrices of side dimension L. This membership check corresponds to a *straightforward* LHSCB with parameter $2L(m-1)$ that is given by $\mathbf{s} \mapsto -\sum_{i=2:m} \log\det({}_{\text{SOS}}(\mathbf{s}_1 + \mathbf{s}_i) \; {}_{\text{SOS}}(\mathbf{s}_1 - \mathbf{s}_i))$. We now describe a membership check for $K^*_{\text{SOS}`_1}$ that requires factorizations of only m matrices, and corresponds to an LHSCB with parameter Lm.

Lemma 5.4.2. *The set* $\{\mathbf{X} \in \mathbb{S}^L_+, \mathbf{Y} \in \mathbb{S}^L : -\mathbf{X} \preceq \mathbf{Y} \preceq \mathbf{X}\}$ *is equal to* $K^2_{\leq} = \text{cl}\{\mathbf{X} \in \mathbb{S}^L_{++}, \mathbf{Y} \in \mathbb{S}^L : \mathbf{X} - \mathbf{Y}\mathbf{X}^{-1}\mathbf{Y} \succeq 0\}$.

Proof. For inclusion in one direction:

$$\text{cl}\{\mathbf{X} \in \mathbb{S}^L_{++}, \mathbf{Y} \in \mathbb{S}^L : \mathbf{X} - \mathbf{Y}\mathbf{X}^{-1}\mathbf{Y} \succeq 0\} \tag{5.32a}$$

$$= \left\{\mathbf{X} \in \mathbb{S}^L_+, \mathbf{Y} \in \mathbb{S}^L : \begin{pmatrix} \mathbf{X} & \mathbf{Y} \\ \mathbf{Y} & \mathbf{X} \end{pmatrix} \succeq 0, \; \begin{pmatrix} \mathbf{X} & -\mathbf{Y} \\ -\mathbf{Y} & \mathbf{X} \end{pmatrix} \succeq 0\right\} \tag{5.32b}$$

$$\left\{\mathbf{X} \in \mathbb{S}^L_+, \mathbf{Y} \in \mathbb{S}^L : 2\mathbf{v}^\top \mathbf{X}\mathbf{v} \pm 2\mathbf{v}^\top \mathbf{Y}\mathbf{v} \geq 0, \; \forall \mathbf{v} \in \mathbb{R}^L\right\} \tag{5.32c}$$

$$= \{\mathbf{X} \in \mathbb{S}^L_+, \mathbf{Y} \in \mathbb{S}^L : \mathbf{X} + \mathbf{Y} \succeq 0, \mathbf{X} - \mathbf{Y} \succeq 0\}. \tag{5.32d}$$

For the other direction, suppose $-\mathbf{X} \preceq \mathbf{Y} \preceq \mathbf{X}$. Then $\mathbf{X} \succeq 0$, $\mathbf{Y} + \mathbf{X} \succeq 0$, $\mathbf{X} - \mathbf{Y} \succeq 0$. Note that $(\mathbf{Y} + \mathbf{X})\mathbf{X}^{-1}(\mathbf{X} - \mathbf{Y}) = \mathbf{X} - \mathbf{Y}\mathbf{X}^{-1}\mathbf{Y}$ is symmetric. Due to Subramanian and Bhagwat [1979, Corollary 1], this product of three matrices also

has nonnegative eigenvalues. We conclude that $-\mathbf{X} \quad \mathbf{Y} \quad \mathbf{X}$ implies $\mathbf{X} \quad 0$ and $\mathbf{X} - \mathbf{Y}\mathbf{X}^{-1}\mathbf{Y} \quad 0$. Since $-\mathbf{X} \quad \mathbf{Y} \quad \mathbf{X} = \mathrm{cl}\{-\mathbf{X} \quad \mathbf{Y} \quad \mathbf{X}\}$, taking closures gives the result. □ □

By Lemma 5.4.2 we can write the dual cone as:

$$K^*_{\mathrm{SOS}`_1} = \mathrm{cl}\left\{ \begin{array}{r} \mathbf{s} \in \mathrm{R}^{U\times m} : \quad _{\mathrm{SOS}}(\mathbf{s}_1) \quad 0, \\ _{\mathrm{SOS}}(\mathbf{s}_1) - \quad _{\mathrm{SOS}}(\mathbf{s}_i) \quad _{\mathrm{SOS}}(\mathbf{s}_1)^{-1} \quad _{\mathrm{SOS}}(\mathbf{s}_i) \quad 0, \forall i \in \mathrm{J2..}m\mathrm{K} \end{array} \right\}. \tag{5.33}$$

Theorem 5.4.3. *The function* $F : \mathrm{R}^{U\times m} \to \mathrm{R}$ *given by:*

$$\begin{aligned} F(\mathbf{s}) = -\textstyle\sum_{i\in \mathrm{J2::}m\mathrm{K}} \mathrm{logdet}(\ _{\mathrm{SOS}}(\mathbf{s}_1) - \ _{\mathrm{SOS}}(\mathbf{s}_i) \ _{\mathrm{SOS}}(\mathbf{s}_1)^{-1} \ _{\mathrm{SOS}}(\mathbf{s}_i)) - \\ \mathrm{logdet}(\ _{\mathrm{SOS}}(\mathbf{s}_1)) \end{aligned} \tag{5.34}$$

is an Lm*-LHSCB for* $K^*_{\mathrm{SOS}`_1}$.

Proof. It is easy to verify that F is a logarithmically homogeneous barrier, and we show it is an Lm-self-concordant barrier. As in Theorem 5.4.1, we define an auxiliary cone:

$$K^m_{\hat{}_1} = \{(\mathbf{X}_1, \ldots, \mathbf{X}_m) \in \mathrm{S}^L_+ \times (\mathrm{R}^{L\times L})^{m-1} : (\mathbf{X}_1, \mathbf{X}_i) \in K^2_{\hat{}_2} \forall i \in \mathrm{J2..}m\mathrm{K}\}. \tag{5.35}$$

Let $\hat{F} : \mathrm{S}^L_{++} \times (\mathrm{R}^{L\times L})^{m-1} \to \mathrm{R}$ be defined as $\hat{F}(\mathbf{X}_1, \ldots, \mathbf{X}_m) = -\sum_{i\in \mathrm{J2::}m\mathrm{K}} \mathrm{logdet}(\mathbf{X}_1 - \mathbf{X}_i\mathbf{X}_1^{-1}\mathbf{X}_i^{>}) - \mathrm{logdet}(\mathbf{X}_1)$. We argue that $\hat{F}$ is an Lm-self-concordant barrier for $K^m_{\hat{}_1}$. F is a composition of $\hat{F}$ with the same linear map used in Theorem 5.4.1 and self-concordance of F then follows by the same reasoning.

Let $\quad = \mathrm{S}^L_+ \times (\mathrm{R}^{L\times L})^{m-1}$ and $\mathbf{H} : \mathrm{int}(\ ) \to (\mathrm{S}^L_+)^{m-1}$ be defined by:

$$\mathbf{H}(\mathbf{X}_1, \ldots, \mathbf{X}_m) = \left(\mathbf{X}_1 - \mathbf{X}_2\mathbf{X}_1^{-1}\mathbf{X}_2^{>}, \ldots, \mathbf{X}_1 - \mathbf{X}_m\mathbf{X}_1^{-1}\mathbf{X}_m^{>}\right). \tag{5.36}$$

We claim that $\mathbf{H}$ is $((\mathrm{S}^L_+)^{m-1}, 1)$-compatible with the domain $\quad$. This amounts to showing that for all $i \in \mathrm{J2..}m\mathrm{K}$, the mapping $\mathbf{H}_i : \mathrm{S}^L_{++} \times \mathrm{R}^{L\times L} \to \mathrm{S}^L$, $\mathbf{H}_i(\mathbf{X}) = \mathbf{X}_1 - \mathbf{X}_i\mathbf{X}_1^{-1}\mathbf{X}_i^{>}$ is $(\mathrm{S}^L_+, 1)$-compatible with the domain $\mathrm{S}^L_+ \times \mathrm{R}^{L\times L}$ (the requirements

for compatibility decouple for each i). The latter holds since $\mathbf{H}_i$ is equivalent to the function $\mathbf{G}$ from Theorem 5.4.1 with $m = 2$. Then by Nesterov and Nemirovskii [1994, Lemma 5.1.7], $\hat{F}$ is an Lm-self-concordant barrier. □ □

Note that we rely on an analogy of a representation for the ℓ_1-norm cone (see [Coey et al., 2021d, Section 4.1]) in (5.33). From this we derive an LHSCB that is analogous to the ℓ_1-norm cone LHSCB. On the other hand, we are not aware of an e□cient LHSCB for its dual, the ℓ_1-norm cone, so we cannot use the same technique to derive an LHSCB for the dual of a polynomial analogy to the ℓ_1-norm cone.

5.5 Implementation details

In Sections 5.5.1 to 5.5.3 we describe the gradients and Hessians of the LHSCBs for K_{SOSPSD}, $K_{\text{SOS}\grave{}_2}$, and $K_{\text{SOS}\grave{}_1}$, which are required as oracles in an algorithm like Coey et al. [2021d]. We give computational complexities of the Hessian oracles for each cone. All the oracles we describe are implemented in the open-source solver Hypatia [Coey et al., 2021d].[3]

5.5.1 SOS-PSD

To draw comparisons between K_{SOSPSD} and its SOS representation (5.5), let us outline how we modify the representation of K_{SOS} from Section 5.1.1 to account for sparsity in a polynomial of the form $\mathbf{y}^\top \mathbf{Q}(\mathbf{x})\mathbf{y}$.

Suppose we have interpolation points $\mathbf{t}_{i\,2\,\text{J1}::U\text{K}}$ to represent K_{SOS} in $\text{R}[\mathbf{x}]_{n;2d}$. Let $\underline{\mathbf{t}}_{i\,2\,\text{J1}::\,\text{sd}(m)\text{K}}$ represent distinct points in R^m, where at most two components in $\underline{\mathbf{t}}_i$ equal one and the rest equal zero, for all i *2* J1.. sd(m)K. We can check that the Cartesian product of $\mathbf{t}_{i\,2\,\text{J1}::U\text{K}}$ and $\underline{\mathbf{t}}_{i\,2\,\text{J1}::\,\text{sd}(m)\text{K}}$, given by $f(\mathbf{t}_1, \underline{\mathbf{t}}_1), \ldots, (\mathbf{t}_U, \underline{\mathbf{t}}_{\text{sd}(m)})g$ gives U sd(m) unisolvent points. The polynomial $\mathbf{y}^\top \mathbf{Q}(\mathbf{x})\mathbf{y}$ from (5.5) is then characterized by its evaluation at these these points.

Now let $\underline{p}_{i\,2\,\text{J1}::m\text{K}}$ be polynomials in $\text{R}[\mathbf{y}]_{m;1}$ such that $\underline{p}_i(y_1, \ldots, y_m) = y_i$ for all

[3] Available at https://github.com/chriscoey/Hypatia.jl.

$i \in [1..m]$. Recall that for K_{SOS} in $\mathbb{R}[\mathbf{x}]_{n,2d}$, $\mathbf{P}$ is defined by $P_{u,\cdot} = p_\cdot(\mathbf{t}_u)$ for all $u \in [1..U], \ell \in [1..L]$. The new matrix $\underline{\mathbf{P}}$ for the U sd(m)-dimensional SOS cone is given by $\underline{\mathbf{P}} = \mathbf{Y} \otimes_K \mathbf{P}$, where $\mathbf{Y} \in \mathbb{R}^{\text{sd}(m)\times m}$, is a Vandermonde matrix of the polynomials $\underline{p}_{i \in [1::m]}$ and points $\underline{\mathbf{t}}_{i \in [1::\text{sd}(m)]}$. Finally, the lifting operator $\underline{\ }_{\text{SOS}}(\mathbf{s}) = \underline{\mathbf{P}}^T \text{diag}(\mathbf{s})\underline{\mathbf{P}}$ is of the same form as $\ _{\text{SOS}}$.

Lemma 5.5.1. *Computing the Hessian of the LHSCB of K_{SOSPSD} requires $O(LU^2m^3)$ time while the Hessian of the LHSCB in the SOS formulation requires $O(LU^2m^5)$ time if $m < L < U$.*

Proof. Define $\mathbf{T}_{i,j} : \mathbb{R}^{U\times m\times m} \to \mathbb{R}^{U\times U}$ for all $i, j \in [1..m]$:

$$\mathbf{T}_{i,j}(\mathbf{S}) = \mathbf{P}(\ _{\text{SOSPSD}}(\mathbf{S})^{-1})_{i,j}\mathbf{P}^T = \left((\mathbf{I}_m \otimes_K \mathbf{P})\ _{\text{SOSPSD}}(\mathbf{S})^{-1}(\mathbf{I}_m \otimes_K \mathbf{P})^T\right)_{i,j}, \quad (5.37)$$

where the indices i, j reference a $U \times U$ submatrix. For all $i, i', j, j' \in [1..m]$, $u, u' \in [1..U]$, the gradient and Hessian of the barrier are:[4]

$$\frac{dF}{dS_{i,j,u}} = -\mathbf{T}_{i,j}(\mathbf{S})_{u,u}, \quad (5.38)$$

$$\frac{d^2F}{dS_{i,j,u}dS_{i',j',u'}} = \mathbf{T}_{i,j'}(\mathbf{S})_{u,u'}\mathbf{T}_{j,i'}(\mathbf{S})_{u,u'}. \quad (5.39)$$

The lifting operator $\ _{\text{SOSPSD}}$ can be computed blockwise in $O(L^2Um^2)$ operations, while $\underline{\ }_{\text{SOS}}$ requires $O(L^2Um^4)$ operations. To avoid computing the explicit inverse $\ _{\text{SOSPSD}}(\mathbf{s})^{-1}$, we use a Cholesky factorization $\ _{\text{SOSPSD}} = \mathbf{L}\mathbf{L}^T$ to form a block triangular matrix $\mathbf{V} = \mathbf{L}^{-1}(\mathbf{I}_m \otimes_K \mathbf{P})^T$ in $O(L^2Um^2)$ operations, while computing the larger $\underline{\mathbf{V}} = \underline{\mathbf{L}}^{-1}\underline{\mathbf{P}}^T$ where $\underline{\ }_{\text{SOS}}(\mathbf{s}) = \underline{\mathbf{L}}\underline{\mathbf{L}}^T$ for the K_{SOS} formulation requires $O(L^2Um^4)$ operations. We use the product $\mathbf{V}^T\mathbf{V}$ to build $\mathbf{T}_{i,j}$ for all $i, j \in [1..m]$ in $O(LU^2m^3)$ operations, while calculating $\underline{\mathbf{V}}^T\underline{\mathbf{V}}$ requires $O(LU^2m^5)$ operations. Once the blocks $\mathbf{T}_{i,j}$ are built, the time complexity to compute the gradient and Hessian are the same for K_{SOSPSD} as for K_{SOS}. □ □

[4] In practice we only store coefficients from the lower triangle of a polynomial matrix and account for this in the derivatives.

5.5.2 SOS-L2

Lemma 5.5.2. *The Hessian of the LHSCB of $K_{\text{SOS}`_2}$ requires $O(LU^2m^2)$ time while the Hessian of the LHSCB in the SOS formulation requires $O(LU^2m^5)$ time if $m < L < U$.*

Proof. Let $\mathbf{T}_{i;j} : \mathsf{R}^{U\times m} \to \mathsf{R}^{U\times U}$ be defined as in (5.37) for all $i, j \in \mathtt{J1}..m\mathtt{K}$, but replacing SOSPSD with $\text{SOS}`_2$. Let $\mathbf{R} = \mathbf{P}(\text{SOS}(\mathbf{s}_1))^{-1}\mathbf{P}^>$. For all $i, i' \in \mathtt{J1}..m\mathtt{K}$, $u, u' \in \mathtt{J1}..U\mathtt{K}$, the gradient and Hessian of the barrier are:

$$\frac{dF}{ds_{i;u}} = \begin{cases} -\sum_{j\in\mathtt{J1}::m\mathtt{K}} \mathbf{T}_{j;j}(\mathbf{s})_{u;u} + (m-2)\mathbf{R}_{u;u} & i = 1 \\ -2\mathbf{T}_{i;1}(\mathbf{s})_{u;u} & i \neq 1, \end{cases} \tag{5.40}$$

$$\frac{d^2F}{ds_{i;u}ds_{i';u'}} = \begin{cases} \sum_{j\in\mathtt{J1}::m\mathtt{K};k\in\mathtt{J1}::m\mathtt{K}}(\mathbf{T}_{j;k}(\mathbf{s})_{u;u'})^2 - (m-2)(\mathbf{R}_{u;u'})^2 & i = i' = 1 \\ 2\sum_{j\in\mathtt{J1}::m\mathtt{K}} \mathbf{T}_{j;1}(\mathbf{s})_{u;u'}\mathbf{T}_{j;i'}(\mathbf{s})_{u;u'} & i = 1, i' \neq 1 \\ 2\sum_{j\in\mathtt{J1}::m\mathtt{K}} \mathbf{T}_{1;j}(\mathbf{s})_{u;u'}\mathbf{T}_{i;j}(\mathbf{s})_{u;u'} & i \neq 1, i' = 1 \\ 2(\mathbf{T}_{1;1}(\mathbf{s})_{u;u'}\mathbf{T}_{i;i'}(\mathbf{s})_{u;u'} + \mathbf{T}_{i;1}(\mathbf{s})_{u;u'}\mathbf{T}_{1;i'}(\mathbf{s})_{u;u'}) & i \neq 1, i' \neq 1. \end{cases} \tag{5.41}$$

To compute the blocks $\mathbf{T}_{i;j}(\mathbf{s})$ we require an inverse of the matrix $\text{SOS}`_2(\mathbf{s})$. It can be verified that:

$$\text{SOS}`_2(\mathbf{s})^{-1} = \begin{bmatrix} \mathbf{0} & \\ & \mathbf{I}_{m-1} \quad K \quad \text{SOS}(\mathbf{s}_1)^{-1} \end{bmatrix} + \mathbf{U}\ (\mathbf{s})^{-1}\mathbf{U}^>, \tag{5.42}$$

where,

$$\mathbf{U}^> = \begin{bmatrix} -\mathbf{I}_L & \text{SOS}(\mathbf{s}_2)\ \text{SOS}(\mathbf{s}_1)^{-1} & \text{SOS}(\mathbf{s}_3)\ \text{SOS}(\mathbf{s}_1)^{-1} & \dots & \text{SOS}(\mathbf{s}_m)\ \text{SOS}(\mathbf{s}_1)^{-1} \end{bmatrix}.$$

Computing $\mathbf{T}_{i;j}$ for all $i, j \in \mathtt{J1}..m\mathtt{K}$ is the most expensive step in obtaining the Hessian and we do this in $O(LU^2m^2)$ operations. The complexity of computing the Hessian in the SOS formulation of $K_{\mathsf{Arw}\,\text{SOSPSD}}$ is the same as in the SOS formulation of K_{SOSPSD} since the cones have the same dimension. □ □

5.5.3 SOS-L1

Lemma 5.5.3. *The Hessians of the LHSCBs of $K_{\text{SOS}\grave{}_1}$ and its SOS formulation require $O(LU^2m)$ time if $m < L < U$.*

Proof. Let $\mathbf{T}_{i;j}(\mathbf{s}) : \mathsf{R}^{U\times m} \to \mathsf{S}^U$ for all $i \in \mathtt{J2..}m\mathtt{K}, j \in \{1,2\}$ be defined by:

$$\mathbf{T}_{i;j}(\mathbf{s}) = \mathbf{P}(\ _{\text{SOS}\grave{}_2}((s_1,s_i))^{-1})_{1;j}\mathbf{P}^\top \tag{5.43}$$

$$= \left((\mathbf{I}_m \otimes_K \mathbf{P})\ _{\text{SOS}\grave{}_2}((s_1,s_i))^{-1}(\mathbf{I}_m \otimes_K \mathbf{P})^\top\right)_{1;j}. \tag{5.44}$$

For all $i, i' \in \mathtt{J1..}m\mathtt{K}$, $u, u' \in \mathtt{J1..}U\mathtt{K}$, the gradient and Hessian of the barrier are:

$$\frac{dF}{ds_{i;u}} = \begin{cases} -2\sum_{j\in\mathtt{J2::}m\mathtt{K}} \mathbf{T}_{j;1}(\mathbf{s})_{u;u} + (m-2)\mathbf{R}_{u;u} & i = 1 \\ -2\mathbf{T}_{i;2}(\mathbf{s})_{u;u} & i \neq 1, \end{cases} \tag{5.45}$$

$$\frac{d^2F}{ds_{i;u}ds_{i';u'}} = \begin{cases} 2\sum_{j\in\mathtt{J2::}m\mathtt{K};k\in\{1;2\}}(\mathbf{T}_{j;k}(\mathbf{s})_{u;u'})^2 - (m-2)(\mathbf{R}_{u;u'})^2 & i = i' = 1 \\ 4\mathbf{T}_{i;1}(\mathbf{s})_{u;u'}\mathbf{T}_{i;2}(\mathbf{s})_{u;u'} & i \neq 1, i' = 1 \\ 4\mathbf{T}_{i';1}(\mathbf{s})_{u;u'}\mathbf{T}_{i';2}(\mathbf{s})_{u;u'} & i = 1, i' \neq 1 \\ 2\sum_{k\in\{1;2\}}(\mathbf{T}_{i;k}(\mathbf{s})_{u;u'})^2 & i = i' \neq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.46}$$

Calculating $\mathbf{T}_{i;j}(\mathbf{s})$ for all $i \in \mathtt{J2..}m\mathtt{K}, j \in \{1,2\}$ can be done in $O(LU^2m)$ operations. The Hessian of the SOS formulation requires computing $O(m)$ Hessians of SOS cones that require $O(LU^2)$ time. We use the block arrowhead structure of the Hessian when applying its inverse similarly to (5.42). □ □

5.6 Numerical example

For each cone ($K_{\text{SOSPSD}}, K_{\text{SOS}\grave{}_2}, K_{\text{SOS}\grave{}_1}$) we compare the computational time to solve a simple example with its SOS formulation from Section 5.2. We use an example analogous to the polynomial envelope problem from [Papp and Yildiz, 2019, Section

7.2], but replace the nonnegativity constraint by a conic inequality. Let $q_{i2\mathbb{J}2::m\mathbb{K}}(\mathbf{x})$ be randomly generated polynomials in $\mathsf{R}_{n;2d_r}[\mathbf{x}]$. We seek a polynomial that gives the tightest approximation to the ℓ_1 or ℓ_2 norm of $(q_2(\mathbf{x}), \ldots, q_m(\mathbf{x}))$ for all $\mathbf{x} \in [-1,1]^n$:

$$\min_{q_1(\mathbf{x}) \in \mathsf{R}_{n,2d}[\mathbf{x}]} \int_{[-1;1]^n} q_1(\mathbf{x}) d\mathbf{x} : \tag{5.47a}$$

$$q_1(\mathbf{x}) \geq \|(q_2(\mathbf{x}), \ldots, q_m(\mathbf{x}))\|_p \qquad \forall \mathbf{x} \in [-1,1]^n, \tag{5.47b}$$

with $p \in \{1,2\}$ in (5.47b).

To restrict (5.47b) over $[-1,1]^n$, we use *weighted sum of squares* (WSOS) formulations. A polynomial $q(\mathbf{x})$ is WSOS with respect to weights $g_{i2\mathbb{J}1::K\mathbb{K}}(\mathbf{x})$ if it can be expressed in the form of $q(\mathbf{x}) = \sum_{i2\mathbb{J}1::K\mathbb{K}} g_i(\mathbf{x}) p_i(\mathbf{x})$, where $p_{i2\mathbb{J}1::K\mathbb{K}}(\mathbf{x})$ are SOS. Papp and Yildiz [2019, Section 6] show that the dual WSOS cone (we will write K^*_{WSOS}) may be represented by an intersection of K^*_{SOS} cones. We represent the dual *weighted* cones K^*_{WSOSPSD}, $K^*_{\text{WSOS}\grave{}_2}$ and $K^*_{\text{WSOS}\grave{}_1}$ analogously using intersections of K^*_{SOSPSD}, $K^*_{\text{SOS}\grave{}_2}$ and $K^*_{\text{SOS}\grave{}_1}$ respectively.

Let $\mathbf{f}_{i2\mathbb{J}1::m\mathbb{K}}$ denote the coefficients of $q_{i2\mathbb{J}1::m\mathbb{K}}(\mathbf{x})$ and let $\mathbf{w} \in \mathsf{R}^U$ be a vector of quadrature weights on $[-1,1]^n$. A low dimensional representation of (5.47) may be written as:

$$\min_{\mathbf{f}_1 \in \mathsf{R}^U} \mathbf{w}^\top \mathbf{f}_1 : \quad (\mathbf{f}_1, \ldots, \mathbf{f}_m) \in K, \tag{5.48}$$

where K is $K_{\text{WSOS}\grave{}_2}$ or $K_{\text{WSOS}\grave{}_1}$. If $p = 2$, we compare the $K_{\text{WSOS}\grave{}_2}$ formulation with two alternative formulations involving $K_{\text{ArwSOSPSD}}$. We use either K_{WSOSPSD} to model $K_{\text{ArwSOSPSD}}$ as implied in (5.9), or K_{WSOS} as in (5.7). For $p = 1$, we build an

SOS formulation by replacing (5.48) with:

$$\min_{\mathbf{f}_1,\mathbf{g}_2,\ldots,\mathbf{g}_m,\mathbf{h}_2,\ldots,\mathbf{h}_m \in \mathbb{R}^U} \mathbf{w}^\top \mathbf{f}_1 : \tag{5.49a}$$

$$\mathbf{f}_1 - \sum_{i=2}^{m} (\mathbf{g}_i + \mathbf{h}_i) \in K_{\mathrm{WSOS}}, \tag{5.49b}$$

$$\mathbf{f}_i - \mathbf{g}_i + \mathbf{h}_i = 0, \quad \mathbf{g}_i, \mathbf{h}_i \in K_{\mathrm{WSOS}} \qquad \forall i = 2,\ldots,m. \tag{5.49c}$$

We select interpolation points using a heuristic adapted from [Papp and Yildiz, 2019, Sommariva and Vianello, 2009]. We uniformly sample N interpolation points, where $N \geq U$. We form a Vandermonde matrix of the same structure as the matrix $\mathbf{P}$ used to construct the lifting operator, but using the N sampled points for rows. We perform a QR factorization and use the first U indices from the permutation vector of the factorization to select U out of N rows to keep.

All experiments are performed on hardware with an AMD Ryzen 9 3950X 16-Core Processor (32 threads) and 128GB of RAM, running Ubuntu 20.10, and Julia 1.8 [Bezanson et al., 2017]. Optimization models are built using JuMP [Lubin and Dunning, 2015] and solved with Hypatia 0.5.3 using our specialized, predefined cones. Scripts we use to run our experiments and raw results are available in the Hypatia repository.[5] We use default settings in Hypatia and set relative optimality and feasibility tolerances to 10^{-7}.

In Tables 5.2 and 5.3, we show Hypatia's termination status, number of iterations, and solve times for $n \in \{1, 4\}$ and varying values of d_r and m. The termination status (*st*) columns of Tables 5.2 and 5.3 use the following codes to classify solve runs:

co the solver claims the primal-dual certificate returned is optimal given its numerical tolerances,

tl a limit of 1800 seconds is reached,

rl a limit of approximately 120GB of RAM is reached,

[5] Instructions to repeat our experiments are at https://github.com/chriscoey/Hypatia.jl/tree/master/benchmarks/natvsext.

sp the solver terminates due to slow progress during iterations,

er the solver reports a different numerical error,

sk we skip the instance because the solver reached a time or RAM limit on a smaller instance.

If $p = 1$, we let $d = d_r$, where the maximum degree of $q_1(\mathbf{x})$ is $2d$. If $p = 2$, we vary $d \in \{d_r, 2d_r\}$ and add an additional column *obj* in Table 5.2 to show the ratio of the objective value under the K_{WSOS} (or equivalently K_{WSOSPSD}) formulation divided by the objective value under the $K_{\text{WSOS}^1_2}$ formulation. Note that in our setup, the dimension of $K_{\text{WSOS}^1_2}$ only depends on d. A more flexible implementation could allow polynomial components to have different degrees in $K_{\text{WSOS}^1_2}$ for the $d = 2d_r$ case.

For $p = 2$ and $d = 2d_r$, the difference in objective values between $K_{\text{WSOS}^1_2}$ and alternative formulations is less than 1% across all converged instances. For $p = 2$ and $d = d_r$, the difference in the objective values is around 10–43% across converged instances. However, the solve times for $K_{\text{WSOS}^1_2}$ with $d = 2d_r$ are sometimes faster than the solve times of alternative formulations with $d = d_r$ and equal values of n, m, and d_r. This suggests that it may be beneficial to use $K_{\text{WSOS}^1_2}$ in place of SOS formulations, but with higher maximum degree in the $K_{\text{WSOS}^1_2}$ cone. The solve times using K_{WSOSPSD} are slightly faster than the solve times using K_{WSOS}. For the case where $p = 1$, the $K_{\text{WSOS}^1_1}$ formulation is faster than the K_{WSOS} formulation, particularly for larger values of m. We also observe that the number of iterations the algorithm takes for $K_{\text{WSOS}^1_2}$ compared to alternative formulations varies, but larger for $K_{\text{WSOS}^1_1}$ compared to the alternative SOS formulation.

				$K_{\text{SOS}\ell_2}$			K_{SOS}			K_{SOSPSD}			
n	d_r	m	d	st	iter	time	st	iter	time	st	iter	time	obj
1	20	4	20	co	13	0.1	co	17	0.4	co	13	0.2	0.89
			40	co	16	0.2	co	19	1.8	co	15	1.1	0.99
		8	20	co	13	0.1	co	17	2.9	co	14	2.1	0.85
			40	co	19	0.7	co	21	18.0	co	16	10.0	1.00
		16	20	co	14	0.4	co	19	48.0	co	14	27.0	0.80
			40	co	21	2.4	co	20	264.0	co	17	188.0	1.00
		32	20	co	15	1.6	co	22	1189.0	co	17	843.0	0.78
			40	co	23	13.0	tl	3	2033.0	tl	7	2075.0	0.03
		64	20	co	17	8.5	rl	*	*	rl	*	*	*
			40	co	20	59.0	sk	*	*	sk	*	*	*
	40	4	40	co	14	0.2	co	17	1.4	co	14	1.0	0.89
			80	co	19	1.0	co	19	7.7	co	17	6.2	0.99
		8	40	co	16	0.6	co	19	15.0	co	15	9.1	0.82
			80	co	21	3.1	co	21	93.0	co	17	62.0	1.00
		16	40	co	17	2.0	co	20	246.0	co	16	152.0	0.79
			80	co	27	13.0	co	21	1737.0	co	18	1206.0	1.00
		32	40	co	18	7.6	tl	3	2031.0	tl	8	1803.0	0.02
			80	co	27	53.0	rl	*	*	rl	*	*	*
		64	40	co	19	36.0	sk	*	*	sk	*	*	*
			80	co	26	226.0	sk	*	*	sk	*	*	*
4	2	4	2	co	13	0.2	co	18	0.9	co	15	0.6	0.75
			4	co	21	33.0	co	43	133.0	co	37	97.0	1.00
		8	2	co	13	0.4	co	21	11.0	co	18	7.7	0.64
			4	co	21	102.0	tl	49	1816.0	tl	60	1811.0	1.00
		16	2	co	15	2.3	co	30	242.0	co	25	203.0	0.59
			4	co	21	437.0	sk	*	*	sk	*	*	*
		32	2	co	15	10.0	tl	6	1848.0	tl	10	1972.0	15.00
			4	co	22	1707.0	sk	*	*	sk	*	*	*
		64	2	co	15	46.0	sk	*	*	sk	*	*	*
			4	tl	10	1935.0	sk	*	*	sk	*	*	*
	4	4	4	co	17	11.0	co	30	114.0	co	27	93.0	0.69
			8	tl	10	1840.0	rl	*	*	tl	*	*	*
		8	4	co	18	42.0	co	34	1494.0	co	29	1111.0	0.58
		16	4	co	18	174.0	rl	*	*	tl	*	*	*
		32	4	co	16	580.0	sk	*	*	sk	*	*	*
		64	4	tl	10	1853.0	sk	*	*	sk	*	*	*

Table 5.2: Solve time in seconds and number of iterations (iter) for instances with $p = 2$.

			$K_{\mathsf{SOS}\grave{}_1}$			K_{SOS}		
n	d	m	st	iter	time	st	iter	time
1	40	8	co	17	0.5	co	15	0.5
		16	co	21	1.3	co	15	1.9
		32	co	25	3.2	co	15	11.0
		64	co	29	7.6	co	17	87.0
		128	co	32	17.0	co	18	610.0
	80	8	co	21	2.6	co	18	2.6
		16	co	24	5.6	co	17	13.0
		32	co	27	13.0	co	18	89.0
		64	co	31	31.0	co	18	600.0
		128	co	38	83.0	tl	*	*
4	2	8	co	17	0.5	co	17	0.4
		16	co	18	1.0	co	16	1.3
		32	co	24	2.8	co	17	7.8
		64	co	27	6.4	co	17	57.0
		128	co	30	14.0	co	17	400.0
	4	8	co	25	28.0	co	21	54.0
		16	co	28	86.0	co	22	318.0
		32	co	29	198.0	tl	9	1823.0
		64	co	31	423.0	sk	*	*
		128	co	42	1210.0	sk	*	*

Table 5.3: Solve time in seconds and number of iterations (iter) for instances with $p = 1$.

5.7 Conclusions

SOS generalizations of PSD, ℓ_2-norm and ℓ_1-norm constraints can be modeled using specialized cones that are simple to use in a generic interior point algorithm. The characterizations of K_{SOSPSD} and $K_{\text{SOS}\ell_2}$ rely on ideas from Papp and Alizadeh [2013] as well as the use of a Lagrange polynomial basis for e□cient oracles in the multivariate case. For the K_{SOSPSD} barrier, the complexity of evaluating the Hessian is reduced by a factor of $O(m^2)$ from the SOS formulation barrier. This does not result in significant speed improvements since Hessian evaluations are not the bottleneck in an interior point algorithm. In contrast, the dimension and barrier parameter of the $K_{\text{SOS}\ell_2}$ and $K_{\text{SOS}\ell_1}$ cones are lower compared to their SOS formulations, and the complexity of evaluating the Hessian of the $K_{\text{SOS}\ell_2}$ barrier is reduced by a factor of $O(m^3)$ from its SOS formulation. For both $K_{\text{SOS}\ell_2}$ and $K_{\text{SOS}\ell_1}$, the total solve time was generally lower compared to their SOS formulations. While there is no penalty in using lower dimensional representations of SOS-L1 constraints, SOS-L2 formulations give rise to more conservative restrictions than higher dimensional SOS formulations, which is observable in practice.

Chapter 6

E□cient conjugate gradient evaluations

This chapter is based on the submitted paper Kapelevich et al. [2022]. We use slightly di□erent cone symbols from Section 2.5 to clarify the distinctions in Hypatia's power-like cones.

6.1 Introduction

As described in Chapter 1, many of the properties of symmetric PDIPMs are not straightforward to generalize for other conic sets. These properties include access to Nesterov-Todd scaling points, and e□ciently computable oracles for both the primal barrier and its conjugate. In contrast to the algorithm by Skajaa and Ye [2015], the algorithm for nonsymmetric cones by Dahl and Andersen [2021] attempts to generalize some important algorithmic concepts from symmetric cone PDIPMs, to nonsymmetric cones.

Unlike earlier chapters, here we are primarily motivated by the algorithm of Dahl and Andersen [2021], which is implemented in the MOSEK solver. The method is based on a technique by Tunçel [2001], Myklebust and Tunçel [2014] which generalizes the concept of scaling matrices from symmetric cones. Like the algorithm by Skajaa and Ye [2015], the linear systems solved in each iteration are equal in size to those

arising in symmetric algorithms. Unlike the algorithm by Skajaa and Ye [2015], information about the gradient of the conjugate barrier is required to compute search directions. In turn, the search directions satisfy a number of desirable properties such as ensuring that the violations of residuals and complementarity conditions decrease at the same rate. The authors also propose a neighborhood that allows stepping further away from the central path compared to Skajaa and Ye [2015], although there is no proof of polynomial time convergence. One might expect, owing to the use of conjugate barrier information, that the search directions of Dahl and Andersen [2021] would often permit convergence in fewer iterations than the search directions of Skajaa and Ye [2015]. We show some evidence of this in Section 7.1.

The search directions in the algorithm by Dahl and Andersen [2021] relate to those from symmetric algorithms as follows. Given a primal-dual pair of points (s, z) for a symmetric cone, Nesterov and Todd [1997, 1998] show the existence of a unique scaling matrix T satisfying the secant equations $z = Ts$ and $g(s) = Tg^*(z)$, where g and g^* denote the gradients of the primal barrier and its conjugate. A key idea of Myklebust and Tunçel [2014], Dahl and Andersen [2021] is to construct a general positive definite matrix for any cone, satisfying the two secant equations. Dahl and Andersen [2021] choose a specific formula for this scaling matrix that requires calculating the Hessian of a primal barrier function, and adding low rank updates to the Hessian. These low rank updates include adding multiples of the conjugate gradient.

The conjugate gradient can always be evaluated via a numerical procedure (e.g. applying Newton's method to an optimization problem), but this generic approach is computationally slow and can be numerically challenging. In particular, it requires applying the inverse Hessian of the primal barrier in each iteration (which can become a new bottleneck in a PDIPM). A large number of damped Newton iterations may be necessary to get near the region of quadratic convergence towards the end of the PDIPM, when the distance to the cone boundary is small. Our aim is to show e□cient methods of calculating conjugate gradients for seven useful nonsymmetric cones, which an interior point solver could support. Aside from their use in the algorithm by Dahl and Andersen [2021], procedures for evaluating conjugate gradients

are useful due to their applications in the frameworks by Nesterov et al. [1999] and Nesterov [2012]. The cones we study are $K_{\log}$, K_{logdet}, K_{pow}, K_{rtdet}, K_{gpow}, K_{ℓ_1}, and $K_{\ell_{\text{spec}}}$ from Section 2.5. Currently, MOSEK supports three-dimensional variants of the logarithm cone and the radial power cone.

Our conjugate gradient evaluations lead to a second result- by differentiating the procedure to calculate the conjugate gradients we are able to derive closed-form expressions for the inverse Hessian of the primal barrier. This is useful for measuring central path proximity and the linear system methods from Section 2.6. The inverse Hessian is already known for five of the cones above (including the cones from Chapter 3), so we only show it for the hypograph and radial power cones.

6.2 Preliminaries

Let f be a LHSCB for a proper cone K. As a consequence of logarithmic homogeneity (1.3b), the gradient g of f satisfies [Nesterov and Todd, 1997, Equation (2.5)]:

$$\langle -g(w), w\rangle = \nu \qquad \forall w \in K. \tag{6.1}$$

Recall the definition of a convex conjugate from (1.4), and recall that f^* is an LHSCB for K^* if f is an LHSCB for K. The gradient g^* of f^* is given by the unique solutions to the optimization problem in (1.4):

$$g^*(r) := -\arg\max_{w \in \operatorname{int}(K)} \{\langle -r, w\rangle - f(w)\}. \tag{6.2}$$

From (1.4), (6.2) and (6.1):

$$f^*(r) = \langle -r, -g^*(r)\rangle - f(-g^*(r)) = -\nu - f(-g^*(r)). \tag{6.3}$$

We refer to g^* as the *conjugate gradient* (which has no relation to the method of conjugate gradients). The negative gradients of LHSCBs f and f^* are bijective linear

maps between K and K^*. In particular [Myklebust and Tunçel, 2014, Theorem 2.5]:

$$-g^*(-g(w)) = w \qquad \forall w \in K, \tag{6.4a}$$

$$-g(-g^*(r)) = r \qquad \forall r \in K^*. \tag{6.4b}$$

(6.4) characterizes the gradient and conjugate gradient maps as negative inverses of each other. Let H and H^* be the Hessians of f and f^*.

Some of the LHSCBs we use are related to *unitarily invariant* functions. These are a generalization of spectral functions of matrices from Chapter 3. Let $W = U_W \operatorname{diag}(\sigma_W) V_W^\top$ with $\sigma_W \in \mathbb{R}^{d_1}$ be the singular value decomposition of $W \in \mathbb{R}^{d_1 \times d_2}$. If W is symmetric, then σ_W are the eigenvalues of W and $U_W = V_W$. Suppose $F : \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}$ is a function given by $F(W) = f(\sigma_W)$, where $f \in \mathbb{R}^{d_1} \to \mathbb{R}$ is some symmetric function (invariant to the order of its inputs). Then F is unitarily invariant. Let G and g denote the gradients of F and f. In Sections 6.3.1, 6.3.2 and 6.3.4, we use the result by Lewis [1995, Theorem 3.1]:

$$G(W) = U_W \operatorname{Diag}(g(\sigma_W)) V_W^\top. \tag{6.5}$$

6.3 Conjugate gradients

In Sections 6.3.1 to 6.3.4 we offer efficient procedures for evaluating conjugate gradients, characterized by (6.4). We defer some derivations to Section 6.5 to ease readability.

6.3.1 Logarithm cone and log-determinant cone

Let $\varphi : \mathbb{R}^d \to \mathbb{R}$ be the function:

$$\varphi(w) := \sum_{i \in [d]} \log w_i. \tag{6.6}$$

Recall the definition of the logarithm cone from Section 2.5:

$$K_{\log} := \mathrm{cl}\{(u, v, w) \in \mathbb{R} \times \mathbb{R}_> \times \mathbb{R}^d_> : u \le v\varphi\left(\tfrac{w}{v}\right)\}, \tag{6.7}$$

and its $(2 + d)$-LHSCB proved in Section 3.6.4:

$$f(u, v, w) = -\log(v\varphi\left(\tfrac{w}{v}\right) - u) - \log(v) - \textstyle\sum_{i \in [d]} \log(w_i). \tag{6.8}$$

The dual cone is given by:

$$K^*_{\log} := \mathrm{cl}\{(p, q, r) \in \mathbb{R}_< \times \mathbb{R} \times \mathbb{R}^d_> : q \ge p \textstyle\sum_{i \in [d]} \log\left(-\frac{r_i}{p}\right) + pd\}. \tag{6.9}$$

Let ω denote the Wright omega function [Corless and Jeffrey, 2002], which can be well approximated in $O(1)$ time and satisfies:

$$\omega(\beta) + \log(\omega(\beta)) = \beta. \tag{6.10}$$

The Wright omega function is used by Serrano [2015, Chapter 8] for deriving the conjugate barrier of the three-dimensional exponential cone, which is a special case of $K_{\log}$.

Proposition 6.3.1. *The conjugate gradient at* $(p, q, r) \in \mathrm{int}(K^*_{\log})$ *has components:*

$$g^*_p = \frac{-d - 2 + q - p + 2\tau}{p(1-\tau)}, \tag{6.11a}$$

$$g^*_q = -\frac{1}{p(1-\tau)}, \tag{6.11b}$$

$$g^*_{r_i} = \frac{\tau}{r_i(1-\tau)} \qquad \forall i \in [d], \tag{6.11c}$$

where:

$$\omega := d \cdot \omega\left(\tfrac{1}{d}\left(1 + d - \tfrac{q}{p} + \textstyle\sum_{i \in [d]} \log\left(-\tfrac{r_i}{p}\right)\right) - \log(d)\right). \tag{6.12}$$

The proof is given in Section 6.5.1. Note that by substituting (6.11) in (6.3), we

obtain the conjugate barrier:

$$f^*(p,q,r) = -2-d-2\log(-p)-\log\left(\tfrac{(\tau-1)^{d+1}}{\tau^d}\right)-\textstyle\sum_{i\in[\![d]\!]}\log(r_i). \tag{6.13}$$

The conjugate gradient for $K_{\log}$ can be easily modified to obtain a conjugate gradient for the log-determinant cone from Section 2.5 (for simplicity, without vectorizing):

$$K_{\text{logdet}} := \text{cl}\left\{(u,v,W)\in\mathbb{R}\times\mathbb{R}_{>}\times\mathbb{S}^d : u\leq v\,\text{logdet}\left(\tfrac{W}{v}\right)\right\}, \tag{6.14}$$

which has the dual:

$$K^*_{\text{logdet}} := \text{cl}\left\{(p,q,R)\in\mathbb{R}_{<}\times\mathbb{R}\times\mathbb{S}^d : q\geq p(\text{logdet}\left(-\tfrac{R}{p}\right)+d)\right\}. \tag{6.15}$$

Let $W = U_w\,\text{Diag}(\lambda_w)U_w^\top$ be the eigendecomposition of W. K_{logdet} admits the $(2+d)$-LHSCB [Coey et al., 2021a, Section 6]:

$$F(u,v,W) = -\log(v\varphi\left(\tfrac{w}{v}\right)-u)-\log(v)-\textstyle\sum_{i\in[\![d]\!]}\log(\lambda_{w;i}) = f(u,v,\lambda_w). \tag{6.16}$$

Proposition 6.3.2. *Let* $(p,q,R)\in\text{int}(K^*_{\text{logdet}})$ *and let* $R = U_r\,\text{Diag}(\lambda_r)U_r^\top$ *be the eigendecomposition of* R. *The conjugate gradient* G^* *has components:*

$$G^*_p = g^*_p(p,q,\lambda_r) = \tfrac{-d-2+q-p+2\bar{\Omega}}{p(1-\bar{\Omega})}, \tag{6.17a}$$

$$G^*_q = g^*_q(p,q,\lambda_r) = -p^{-1}(1-\quad)^{-1}, \tag{6.17b}$$

$$G^*_R = U_r\,\text{Diag}(g^*_r(p,q,\lambda_r))U_r^\top = \tfrac{\bar{\Omega}}{(1-\bar{\Omega})}R^{-1}, \tag{6.17c}$$

where:

$$:= d\cdot\omega\left(\tfrac{1}{d}\left(1+d-\tfrac{q}{p}+\textstyle\sum_{i\in[\![d]\!]}\log\left(-\tfrac{r_i}{p}\right)\right)-\log(d)\right). \tag{6.18}$$

Proof. For fixed u and v, F is a unitarily invariant function of W. Due to (6.5), the

gradient of F is:

$$G_u = g_u(u, v, \lambda_w), \tag{6.19a}$$

$$G_v = g_v(u, v, \lambda_w), \tag{6.19b}$$

$$G_W = U_w \operatorname{Diag}(g_w(u, v, \lambda_w)) U_w^\top. \tag{6.19c}$$

The result can be verified from (6.4) and (6.19). □

6.3.2 Hypograph power cone and root-determinant cone

Let $\varphi : \mathbb{R}^d \to \mathbb{R}$ be the function:[1]

$$\varphi(w) := \prod_{i \in [d]} w_i^{\alpha_i}, \tag{6.20}$$

parametrized by $\alpha = (\alpha_1, \ldots, \alpha_d)$ such that $\langle e, \alpha \rangle = 1$ and $\alpha \geq 0$. Define the *hypograph-power* cone:

$$K_{\text{hpower}} := \left\{(u, w) \in \mathbb{R} \times \mathbb{R}^d_+ : u \leq \varphi(w)\right\}, \tag{6.21}$$

which admits the $(1 + d)$-LHSCB [Nesterov et al., 2018, Section 5.4.7]:

$$f(u, w) = -\log(\varphi(w) - u) - \textstyle\sum_{i \in [d]} \log(w_i). \tag{6.22}$$

This is the *power mean cone* in Section 2.5. In the special case where $\alpha = {}^{e}/_{d}$, we call K_{hpower} the *hypograph geometric mean* cone, K_{hgeom}. The dual cone is given by [Coey et al., 2021b]:

$$K^*_{\text{hpower}} := \left\{(p, r) \in \mathbb{R}_- \times \mathbb{R}^d_+ : -p \leq \varphi\left(\tfrac{r}{\alpha}\right)\right\}. \tag{6.23}$$

The division in ${}^{r}/_{\alpha}$ should be interpreted componentwise.

Lemma 6.3.3. *Let* $(p, r) \in \operatorname{int}(K^*_{\text{hpower}})$ *parametrized by* α. *The unique root of* $h(y) :=$

[1] We reuse symbols with similar roles across subsections; their meaning should be taken from the definition within each subsection.

$\sum_{i \in [d]} \alpha_i \log(y - p\alpha_i) - \log(\varphi(r))$ *can be easily computed by a quadratically convergent Newton-Raphson method starting at* 0.

The proof is given in Section 6.5.2.

Proposition 6.3.4. *The conjugate gradient at* $(p, r) \in \operatorname{int}(K^*_{\text{hpower}})$ *has components:*

$$g^*_p = -p^{-1} - \hat{y}^{-1}, \tag{6.24a}$$

$$g^*_{r_i} = \frac{p_i \hat{y}^{-1} - 1}{r_i} \qquad \forall i \in [d], \tag{6.24b}$$

where $\hat{y}$ *is the root of* h *from Lemma 6.3.3. In the case where* $\alpha = e/d$*, the conjugate gradient at* $(p, r) \in \operatorname{int}(K^*_{\text{hgeom}})$ *can be written more simply:*

$$g^*_p = -p^{-1} - (\varphi(r) + p/d)^{-1}, \tag{6.25a}$$

$$g^*_{r_i} = -\frac{\varphi'(r)}{r_i(\varphi'(r) + p/d)} \qquad \forall i \in [d]. \tag{6.25b}$$

The proof is given in Section 6.5.2. Substituting (6.25) in (6.3), we obtain a simple expression for the conjugate barrier of K_{hgeom}:

$$f^*(p, r) = -1 - d - d \log\left(\frac{d\varphi'(r) + p}{d\varphi'(r)}\right) - \log(-p) - \sum_{i \in [d]} \log(r_i). \tag{6.26}$$

The conjugate gradient for K_{hgeom} can be easily modified to obtain a conjugate gradient for the root-determinant cone from Section 2.5:

$$K_{\text{rtdet}} := \left\{(u, W) \in \mathbb{R} \times \mathbb{S}^d : u \leq \det(W)^{1/d}\right\}, \tag{6.27}$$

which has the dual [Coey et al., 2021d, Section 4.4]:

$$K^*_{\text{rtdet}} := \left\{(p, R) \in \mathbb{R}_- \times \mathbb{S}^d : -p \leq d \det(R)^{1/d}\right\}. \tag{6.28}$$

Let $W = U_W \operatorname{Diag}(\lambda_W) U_W^\top$ be the eigendecomposition of W. K_{rtdet} admits the $(1 + d)$-

LHSCB [Coey et al., 2021a]:

$$F(u, W) = -\log(\varphi(\lambda_W) - u) - \sum_{i \in [d]} \log(\lambda_{W,i}) = f(u, \lambda_W). \tag{6.29}$$

Proposition 6.3.5. *Let* $(p, R) \in \operatorname{int}(K^*_{\text{rtdet}})$ *and* $R = U_r \operatorname{Diag}(\lambda_r) U_r^\top$ *be the eigendecomposition of* R*. The conjugate gradient* G^* *has components:*

$$G^*_p = g^*_p(p, \lambda_r) = -p^{-1} - (\det(R)^{1/d} + p/d)^{-1}, \tag{6.30a}$$

$$G^*_R = U_r \operatorname{Diag}(g^*_r(p, \lambda_r)) U_r^\top = -\frac{\det(R)^{1/d}}{\det(R)^{1/d} + p/d} R^{-1}. \tag{6.30b}$$

Proof. For fixed u, F is a unitarily invariant function of W. Similar to Proposition 6.3.2, the result can be verified from (6.4) and (6.5). □

6.3.3 Radial power cone

Let $\varphi : \mathbb{R}^d \to \mathbb{R}$ be the function:

$$\varphi(w) := \prod_{i \in [d_2]} w_i^{2^i}, \tag{6.31}$$

parametrized by $\alpha = (\alpha_1, \ldots, \alpha_d)$ such that $\langle e, \alpha \rangle = 1$ and $\alpha \geq 0$. Define the *radial-power* cone:

$$K_{\text{rpower}} := \left\{(u, w) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2}_{\geq} : \|u\| \leq \prod_{i \in [d_2]} w_i^{\alpha_i} = \sqrt{\varphi(w)}\right\}, \tag{6.32}$$

which admits the $(1 + d_2)$-LHSCB [Roy and Xiao, 2021, Theorem 1]:

$$f(u, w) = -\log(\varphi(w) - \|u\|^2) - \sum_{i \in [d_2]} (1 - \alpha_i) \log(w_i). \tag{6.33}$$

This is the *generalized power cone* in Section 2.5. Note that f is not equivalent to the barrier from (6.22), even when $d_1 = 1$. Hence the conjugate barrier and its derivatives take different forms from our results in Section 6.3.2. In the special case where $\alpha = e/d_2$ and $d_1 = 1$, we call K_{rpower} the *radial geometric mean* cone K_{rgeom}.

The dual cone is given by [Chares, 2009, Theorem 4.3.1]:

$$K^*_{\text{rpower}} := \left\{(p, r) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2}_{\geq} : \|p\| \leq \prod_{i \in \llbracket d_2 \rrbracket} \left(\frac{r_i}{\alpha_i}\right)^{\alpha_i}\right\}. \tag{6.34}$$

Lemma 6.3.6. *Let* $(p, r) \in \text{int}(K^*_{\text{hpower}})$ *parametrized by* α *and* $p > 0$*. The unique positive root of:*

$$h(y) := \sum_{i \in \llbracket d_2 \rrbracket} 2\alpha_i \log\left(2\alpha_i y^2 + \frac{2y(1+\alpha_i)}{p}\right) - \log(\varphi(r)) - \log\left(\frac{2y}{p} + y^2\right) - 2\log\left(\frac{2y}{p}\right) \tag{6.35}$$

can be easily computed numerically by a quadratically convergent Newton-Raphson method starting at $p^{-1} + d_2 \frac{p + \sqrt{\varphi'(r)(d_2^2 - p^2)\varphi'(r) + d_2^2 - 1)}}{\varphi'(r)d_2^2 - p^2}$.

The proof is given in Section 6.5.3.

Proposition 6.3.7. *The conjugate gradient at* $(p, r) \in \text{int}(K^*_{\text{rpower}})$ *is given by:*

$$g^*_{p_i} = \begin{cases} 0 & p = 0, \\ -y\frac{p_i}{\|p\|} & p \neq 0, \end{cases} \qquad \forall i \in \llbracket d_1 \rrbracket, \tag{6.36a}$$

$$g^*_{r_i} = -\frac{\alpha_i(1 + py) + 1}{r_i} \qquad \forall i \in \llbracket d_2 \rrbracket, \tag{6.36b}$$

where y *is the positive root of* h *from Lemma 6.3.6. In the case where* $d_1 = 1$ *and* $\alpha = e/d_2$*, the conjugate gradient at* $(p, r) \in \text{int}(K^*_{\text{rgeom}})$ *is:*

$$g^*_p = -p^{-1} + d_2 \frac{p + \sqrt{\varphi'(r)(d_2^2 - p^2)\varphi'(r) + d_2^2 - 1)}}{\varphi'(r)d_2^2 - p^2}, \tag{6.37a}$$

$$g^*_{r_i} = -r_i^{-1}\left(\frac{p^2 + \sqrt{\varphi'(r)(d_2^2\varphi'(r) + d_2^2 p^2 - p^2)}}{\varphi'(r)d_2^2 - p^2} + 1\right) \qquad \forall i \in \llbracket d_2 \rrbracket. \tag{6.37b}$$

The proof is given in Section 6.5.3.

6.3.4 Infinity norm cone and spectral norm cone

Recall the definition of the infinity norm cone from Section 2.5:

$$K_{\infty,1} := \{(u, w) \in \mathbb{R} \times \mathbb{R}^d : u \geq \|w\|_1\}, \tag{6.38}$$

which admits the $(1 + d)$-LHSCB [Güler, 1996, section 7.5]:

$$f(u, w) = -\sum_{i \in [d]} \log(u^2 - w_i^2) + (d - 1)\log(u). \tag{6.39}$$

The dual cone is the epigraph of the ℓ_1 norm function:

$$K^*_{\infty,1} := \{(p, r) \in \mathbb{R} \times \mathbb{R}^d : p \geq \|r\|_1\}. \tag{6.40}$$

Lemma 6.3.8. *Let* $(p, r) \in \operatorname{int}(K^*_{\infty,1})$. *The unique negative root of* $h(y) := py + \sum_{i \in [d]} \sqrt{1 + r_i^2 y^2} + 1$ *can be easily computed by a quadratically convergent Newton-Raphson method starting at* $\max\{-(p - \sum_{i \in [d]} |r_i|)^{-1}, -\frac{d+1}{p}\}$.

Proposition 6.3.9. *The conjugate gradient at* $(p, r) \in \operatorname{int}(K^*_{\infty,1})$ *is given by:*

$$g^*_p = \hat{y}, \tag{6.41a}$$

$$g^*_{r_i} = \begin{cases} 0 & r_i = 0, \\ \dfrac{p\sqrt{1+\hat{y}^2 r_i^2} - 1}{r_i} & r_i \neq 0, \end{cases} \qquad \forall i \in [d], \tag{6.41b}$$

where $\hat{y}$ *is the negative root of* h *from Lemma 6.3.8.*

The proof is given in Section 6.5.4. The conjugate gradient for $K_{\infty,1}$ can be easily modified to obtain a conjugate gradient for the spectral norm cone from Section 2.5 (for simplicity, without vectorizing):

$$K_{\text{spec}} := \{(u, W) \in \mathbb{R} \times \mathbb{R}^{d_1 \times d_2} : u \geq \sigma_{\max}(W)\}, \tag{6.42}$$

where $\sigma_{\max}$ is the maximum singular value function. Let $R = U_r \operatorname{Diag}(\sigma_r) V_r^\top$ be the singular value decomposition of $R \in \mathbb{R}^{d_1 \times d_2}$. The dual cone is the epigraph of the

nuclear norm:

$$K^*_{\text{spec}} := \{(p, R) \in \mathbb{R} \times \mathbb{R}^{d_1 \times d_2} : p \geq \sum_{i \in [d_1]} \sigma_{r;i}\}. \tag{6.43}$$

Let $W = U_w \operatorname{Diag}(\sigma_w) V_w^\top$ be the singular value decomposition of W. K^*_{spec} admits the $(1 + d_1)$-LHSCB:

$$F(u, W) = -\sum_{i \in [d_1]} \log(u^2 - \sigma_{w;i}^2) + (d_1 - 1)\log(u) = f(u, \sigma_w). \tag{6.44}$$

Proposition 6.3.10. *The conjugate gradient G^* at $(p, R) \in \operatorname{int}(K^*_{\text{spec}})$ has components:*

$$G_p^* = g^*_p(p, \sigma_r), \tag{6.45a}$$

$$G_R^* = U_r \operatorname{Diag}(g^*_r(p, \sigma_r)) V_r^\top. \tag{6.45b}$$

Proof. For fixed u, F is a unitarily invariant function of W. Similar to Proposition 6.3.2, the result can be verified using (6.4) and (6.5). □

6.4 Inverse Hessians

In Chapter 3 we derive efficient inverse Hessian operators for a number of cones, including $K_{\log}$, K_{logdet}, K_{hgeom}, and K_{rtdet}. Inverse Hessians for K_{ℓ_1} and K_{spec} are described in a forthcoming paper by Coey.[2] As described in Chapter 2, the inverse Hessians are useful for measuring proximity to the central path in Hypatia's algorithm, as well as some of the linear system solving methods. Our derivations of the conjugate gradients offer an alternative method for deriving inverse Hessian operators. Since these have not been written for K_{hpower} or K_{rpower}, we derive those inverse Hessian operators here.

[2] Implementation by C. Coey can be found at https://github.com/chriscoey/Hypatia.jl/blob/master/src/Cones/epinormspectral.jl.

6.4.1 Hypograph power cone

Proposition 6.4.1. *The inverse Hessian operator at* $\mathfrak{u} = (u, w)$ *2 K*$_{\text{hpower}}$ *parametrized by* α*, in the direction* $\mathfrak{x} = (x, z)$ *2* $\mathbb{R}^{1+d}$ *is:*

$$(H(\mathfrak{u})^{-1} \cdot \mathfrak{x})_u = \big((\varphi(w) - u)^2 + \tfrac{k_2}{k_3}u^2\big)x - \tfrac{'(w)}{k_3}\langle z, \tfrac{w}{k_0}\rangle, \tag{6.46a}$$

$$(H(\mathfrak{u})^{-1} \cdot \mathfrak{x})_{w_i} = \tfrac{w_i^2}{k_{1,i}}z_i + \tfrac{{}_i w_i}{k_{1,i}}\tfrac{'(w)}{k_3}x + \tfrac{g_u\,'(w)}{k_3}\langle z, \tfrac{w}{k_0}\rangle\tfrac{{}_i w_i}{k_{1,i}} \qquad 8i \in [d], \tag{6.46b}$$

where φ *is defined as in Section 6.3.2, and:*

$$k_{1;i} := 1 + \alpha_i\varphi(w)g_u \qquad 8i \in [d], \tag{6.47a}$$

$$k_2 := \textstyle\sum_{i\in[d]} \frac{{}_i^2}{k_{1,i}}, \tag{6.47b}$$

$$k_3 := 1 - \varphi(w)g_u k_2. \tag{6.47c}$$

The proof is given in Section 6.6.1.

6.4.2 Radial power cone

Proposition 6.4.2. *The inverse Hessian operator at* $\mathfrak{u} = (u, w)$ *2 K*$_{\text{rpower}}$ *parametrized by* α*, in the direction* $\mathfrak{x} = (x, z)$ *is:*

$$(H(\mathfrak{u})^{-1} \cdot \mathfrak{x})_{u_i} = \tfrac{}{2}x_i + \tfrac{u_i}{k_3}\Big(- \tfrac{2k_2\,'(w) - k_3}{k_1}\langle x, u\rangle - \langle /_{g_w}, z\rangle\Big) \qquad 8i \in [d_1], \tag{6.48a}$$

$$(H(\mathfrak{u})^{-1} \cdot \mathfrak{x})_{w_i} = -\tfrac{w_i}{g_{w_i}}z_i - \tfrac{{}_i}{k_3 g_{w_i}}\big(\langle x, u\rangle - \tfrac{2kuk^2}{}\langle /_{g_w}, z\rangle\big) \qquad 8i \in [d_2], \tag{6.48b}$$

where φ *is defined as in Section 6.3.3,* $\zeta := \varphi(w) - kuk^2$*, and:*

$$k_1 := \varphi(w) + kuk^2, \tag{6.49a}$$

$$k_2 := \langle /_w, \ /_{g_w}\rangle, \tag{6.49b}$$

$$k_3 := \tfrac{'(w)+kuk^2}{2\,'(w)} + 2k_2\tfrac{kuk^2}{}. \tag{6.49c}$$

The proof is given in Section 6.6.2.

6.5 Proofs of conjugate gradients

6.5.1 Logarithm cone

Proof of Proposition 6.3.1. For convenience, let ζ be the function $\zeta(u, v, w) := v\varphi(w/v) - u$. Let (u, v, w) *2* int($K_{\log}$). Then the gradient of f with respect to components, u, v, and w is:

$$g_u = \zeta(u, v, w)^{-1}, \tag{6.50a}$$

$$g_v = -\zeta(u, v, w)^{-1}(\varphi(\tfrac{w}{v}) - d) - \tfrac{1}{v}, \tag{6.50b}$$

$$g_{w_i} = -\zeta(u, v, w)^{-1}\tfrac{v}{w_i} - \tfrac{1}{w_i} \qquad 8i \; 2 \; \mathrm{J}d\mathsf{K}. \tag{6.50c}$$

Note that $\omega > 1$ since:

$$(p, q, r) \; 2 \; \mathrm{int}(K^*_{\log}) \tag{6.51a}$$

$$) \;\; pd - q + p\textstyle\sum_{i2\mathrm{J}d\mathsf{K}} \log(-\frac{r_i}{p}) < 0 \tag{6.51b}$$

$$, \;\; \tfrac{1}{d}(1 + d - \tfrac{q}{p} + \textstyle\sum_{i2\mathrm{J}d\mathsf{K}} \log(-\frac{r_i}{p})) - \log(d) > \tfrac{1}{d} + \log(\tfrac{1}{d}) \tag{6.51c}$$

$$, \;\; \omega(\tfrac{1}{d}(1 + d - \tfrac{q}{p} + \textstyle\sum_{i2\mathrm{J}d\mathsf{K}} \log(-\frac{r_i}{p})) - \log(d)) > \tfrac{1}{d} \tag{6.51d}$$

$$, \;\; \omega > 1, \tag{6.51e}$$

where (6.51d) follows from (6.51c) by applying ω to both sides, and noting that $\omega(\beta + \log(\beta)) = \beta$ due to (6.10).

We would like to find $g^* := (g^*_p, g^*_q, g^*_r)$ such that $-g(-g^*) = (p, q, r)$. Fix $\zeta := \zeta(-g^*_p, -g^*_q, -g^*_r)$. Then, from (6.50a):

$$p = -\zeta^{-1}. \tag{6.52}$$

Combining (6.50c) with (6.52), we need for all $i \in [d]$:

$$r_i = \zeta^{-1}\frac{g^*_q}{g^*_{r_i}} - \frac{1}{g^*_{r_i}} \tag{6.53a}$$

$$= -p\frac{g^*_q}{g^*_{r_i}} - \frac{1}{g^*_{r_i}} \tag{6.53b}$$

$$\implies g^*_{r_i} = \frac{-pg^*_q - 1}{r_i}. \tag{6.53c}$$

Combining (6.50b) with (6.52), we need (division by r should be interpreted componentwise):

$$q = \zeta^{-1}\left(\varphi\left(\frac{g^*_r}{g^*_q}\right) - d\right) - \frac{1}{g^*_q} \tag{6.54a}$$

$$= -p\left(\varphi\left(\frac{-pg^*_q - 1}{rg^*_q}\right) - d\right) - \frac{1}{g^*_q}. \tag{6.54b}$$

Replacing the definition of φ in (6.54) and rearranging:

$$\frac{q}{p} - d - \sum_{i\in[d]}\log(-\tfrac{r_i}{p}) - 1 = -\frac{1}{pg^*_q} - d\log(1 + \frac{1}{pg^*_q}) - 1 \tag{6.55a}$$

$$\implies -\frac{q}{p} + d + \sum_{i\in[d]}\log(-\tfrac{r_i}{p}) + 1 = 1 + \frac{1}{pg^*_q} + d\log(1 + \frac{1}{pg^*_q}). \tag{6.55b}$$

Note (6.55b) has the form $\beta = a + d\log(a)$. Letting $a = db$:

$$\beta = db + d\log(db) \implies \frac{\beta}{d} - \log(d) = b + \log(b). \tag{6.56}$$

Therefore,

$$d^{-1}\left(1 + d - \frac{q}{p} + \sum_{i\in[d]}\log(-\tfrac{r_i}{p})\right) - \log(d) = 1 + \frac{1}{pg^*_q} + \log\left(1 + \frac{1}{pg^*_q}\right) \tag{6.57a}$$

$$\implies \omega = 1 + \frac{1}{pg^*_q} \tag{6.57b}$$

$$\implies g^*_q = -p^{-1}(1 - \omega)^{-1}. \tag{6.57c}$$

Substituting (6.57c) in (6.53) gives (6.11c). Finally, due to (6.1):

$$g^*_p = \frac{-d - 2 - qg^*_q - \langle r, g^*_r\rangle}{p} = \frac{-d - 2 + \frac{q}{p} + 2\tau}{p(1-\tau)}. \tag{6.58}$$

□

6.5.2 Hypograph power cone

Proof of Lemma 6.3.3. For φ given in (6.20), let y denote the root of:

$$h(y) := \sum_{i \in [d]} \alpha_i \log(y - p\alpha_i) - \log(\varphi(r)), \tag{6.59}$$

which we show is unique. Note that y must satisfy:

$$\varphi(ye - p\alpha_i) = \varphi(r) \tag{6.60a}$$

$$) \quad \varphi\left(\frac{y}{} - pe\right) = \varphi\left(\frac{r}{}\right). \tag{6.60b}$$

Since $(p, r) \in K^*_{\text{hpower}}$, this implies $y > 0$. The derivatives of h are:

$$h'(y) = \sum_{i \in [d]} \frac{i}{y - p_i}, \tag{6.61a}$$

$$h''(y) = \sum_{i \in [d]} -\frac{i}{(y - p_i)^2} < 0. \tag{6.61b}$$

Note that $p < 0$ for $(p, r) \in \text{int}(K^*_{\text{hpower}})$ and therefore $h'(y) > 0$ for all $y > \max_{i \in [d]} \{p\alpha_i\}$, i.e. the domain of h. So the root of h is unique. Since h is concave and increasing, a root-finding Newton-Raphson method will converge quadratically from any initial $y_- < y$ [Süli and Mayers, 2003, Theorem 1.9]. We may pick, for example, $y_- = 0$, which ensures $y_- < y$ and y_- is in the domain of h.

We remark that the solution from the uniform α case can be used as an upper bound on y. To see the validity of the upper bound, consider the function $h(\alpha, y) := \sum_{i \in [d]} \alpha_i \log(y - p\alpha_i) - \log(\varphi(r))$. Observe that $h(\alpha, y)$ is convex and symmetric in α (over the unit simplex). Hence of any fixed y, $h(\alpha, y)$ is minimized at $\alpha = e/d$. Since $h(\alpha, y)$ is also increasing in y, the root of $h(e/d, y) = 0$ upper bounds y. □

Proof of Proposition 6.3.4. For convenience, let ζ be the function $\zeta(u, w) := \varphi(w) - u$, where φ is from (6.20). Let $(u, w) \in \text{int}(K_{\text{hpower}})$. Then the gradient of f with respect

to components u and w is:

$$g_u = \zeta(u,w)^{-1}, \tag{6.62a}$$

$$g_{w_i} = -\zeta(u,w)^{-1}\varphi(w)\alpha_i w_i^{-1} - w_i^{-1} \qquad \forall i \in [d]. \tag{6.62b}$$

We would like to find $g^* := g^*(p,r)$ such that $-g(-g^*) = (p,r)$. Using (6.62a), we need:

$$p = -\zeta(-g^*_p, -g^*_r)^{-1} = -(\varphi(-g^*_r) + g^*_p)^{-1}. \tag{6.63}$$

Using (6.62b) and (6.63), we need for all $i \in [d]$:

$$r_i = \zeta(-g^*_p, -g^*_r)^{-1} \cdot \varphi(-g^*_r)\alpha_i \cdot (-g^{*\,-1}_{r_i}) - g^{*\,-1}_{r_i} \tag{6.64a}$$

$$= -g^{*\,-1}_{r_i}(-p\alpha_i\varphi(-g^*_r) + 1). \tag{6.64b}$$

From (6.63) and (6.64):

$$g^*_p = -p^{-1} - \varphi(-g^*_r), \tag{6.65a}$$

$$g^*_{r_i} = \frac{p_i\varphi(-g^*_r)-1}{r_i} \qquad \forall i \in [d]. \tag{6.65b}$$

It remains to show how to evaluate $\varphi(-g^*_r)$. Applying φ from (6.20) to both sides of (6.64b), after collecting for each i:

$$\varphi(r) = \prod_{i\in[d]}\left(-g^{*\,-1}_{r_i}\left(-p\alpha_i\varphi(-g^*_r) + 1\right)\right)^{\alpha_i} \tag{6.66a}$$

$$= \prod_{i\in[d]}\varphi(-g^*_r)^{-\alpha_i}\left(-p\alpha_i\varphi(-g^*_r) + 1\right)^{\alpha_i} \tag{6.66b}$$

$$= \prod_{i\in[d]}\left(-p\alpha_i + \varphi(-g^*_r)^{-1}\right)^{\alpha_i} \tag{6.66c}$$

$$\Rightarrow \log(\varphi(r)) = \sum_{i\in[d]}\alpha_i \log\left(-p\alpha_i + \varphi(-g^*_r)^{-1}\right) \tag{6.66d}$$

$$\Rightarrow h(\varphi(-g^*_r)^{-1}) = 0. \tag{6.66e}$$

From (6.66e), we can evaluate $\varphi(-g^*_r)^{-1}$ easily due to Lemma 6.3.3. Combining this with (6.65) justifies (6.24). In the special case where $\alpha = e/d$, we can solve $h(y) = 0$

exactly, giving:

$$\eta = \left(\varphi(r) + \tfrac{p}{d}\right)^{-1}. \tag{6.67}$$

Substituting (6.67) in (6.65) gives (6.25). □

6.5.3 Radial power cone

Proof of Lemma 6.3.6. Let η denote the positive root of:

$$h(y) := \sum_{i \in [d_2]} 2\alpha_i \log\left(2\alpha_i y^2 + \tfrac{2y(1+\alpha_i)}{p}\right) - \log(\varphi(r)) - \log\left(\tfrac{2y}{p} + y^2\right) - 2\log\left(\tfrac{2y}{p}\right), \tag{6.68}$$

which we show is unique. In the special case where $\alpha = e/d_2$, we can solve $h(y) = 0$ exactly. It can be verified that $\eta > 0$ is given by:

$$\eta = -p^{-1} + d_2 \frac{p + \sqrt{\varphi'(r)(d_2^2 - p^2)\varphi'(r) + d_2^2 - 1)}}{\varphi'(r)d_2^2 - p^2}, \tag{6.69}$$

where φ is given by (6.31). Note that the denominator is positive since $(p, r) \in \text{int}(K^*_{\text{rgeom}})$. Let us turn to the case of non-uniform α. The first two derivatives of h are:

$$h'(y) = 2\sum_{i \in [d_2]} \frac{\frac{2}{i}}{\alpha_i y + \frac{1+\alpha_i}{p}} - 2\frac{y + \frac{1}{p}}{y(y + \frac{2}{p})}, \tag{6.70a}$$

$$h''(y) = -2\sum_{i \in [d_2]} \frac{\frac{3}{i}}{\left(\alpha_i y + \frac{1+\alpha_i}{p}\right)^2} + \frac{2\left(y^2 + \frac{2}{p^2}\right)}{y^2\left(y + \frac{2}{p}\right)^2}. \tag{6.70b}$$

We have that h is decreasing for $p, y > 0$, since:

$$h'(y) < 2\sum_{i \in [d_2]} \frac{\frac{2}{i}}{a_i y + \frac{1+\alpha_i}{p}} - 2\frac{y}{y\left(y + \frac{2}{p}\right)} \tag{6.71a}$$

$$\leq 2\frac{1}{y + \frac{1+1}{p}} - 2\frac{1}{y + \frac{2}{p}} \tag{6.71b}$$

$$= 0. \tag{6.71c}$$

In the second to last line we use the fact that the right-hand side in the line above is convex in α, and therefore maximized at an extreme point of the simplex that α

belongs to. Similar reasoning shows that $h''(y) > 0$ for $p, y > 0$:

$$h''(y) > -2\sum_{i \in [d_2]} \frac{\frac{3}{i}}{\left({}_i y + \frac{1+{}_i}{p} \right)^2} + \frac{2}{\left(y + \frac{2}{p}\right)^2} \tag{6.72a}$$

$$\geq -\frac{2}{\left(y + \frac{2}{p}\right)^2} + \frac{2}{\left(y + \frac{2}{p}\right)^2} \tag{6.72b}$$

$$= 0. \tag{6.72c}$$

The second inequality follows from the fact that the expression above it is concave in α.

Due to (6.71) and (6.72), a root-finding Newton-Raphson method converges quadratically starting from some y_- such that $y_- \leq \hat{y}$ [Süli and Mayers, 2003, Theorem 1.9]. We may use the solution from the equal powers case, i.e. (6.69) for y_-. To see why, note that the function $h(\alpha, y) = \sum_{i \in [d_2]} 2\alpha_i \log(2\alpha_i y^2 + (1+\alpha_i)\frac{2y}{p}) - \log(\varphi(r)) - \log(\frac{2y}{p} + y^2) - 2\log(\frac{2y}{p})$ is convex and symmetric in α (ignoring y, it can be checked that the Hessian of h is diagonal with nonnegative entries). So for any fixed y, $h(\alpha, y)$ is minimized at $\alpha = {}^e/_{d_2}$. Since h is decreasing, a solution to $h({}^e/_{d_2}, y) = 0$ lower bounds $\hat{y}$. □

Proof of Proposition 6.3.7. For convenience, let ζ be the function $\zeta(u, w) := \varphi(w) - \|u\|^2$, where φ is from (6.31). Let $(u, w) \in \text{int}(K_{\text{rpower}})$. Then the gradient of f with respect to components u and w is:

$$g_{u_i} = \frac{2u_i}{(u;w)} \qquad \forall i \in [d_1], \tag{6.73a}$$

$$g_{w_i} = -\frac{2\ {}_i\varphi'(w)}{w_i\ (u;w)} - \frac{1-{}_i}{w_i} \qquad \forall i \in [d_2]. \tag{6.73b}$$

We would like to find $g^* := g^*(p, r)$ such that $-g(-g^*) = (p, r)$. First, if $p = 0$, it is easy to see from (6.73) that:

$$g^*_{p_i} = 0 \qquad \forall i \in [d_1], \tag{6.74a}$$

$$g^*_{r_i} = -\frac{1+{}_i}{r_i} \qquad \forall i \in [d_2]. \tag{6.74b}$$

For the case $p \neq 0$, let us show that without loss of generality, we may assume $p \in \mathbb{R}_{\geq}$. Fix $\zeta := \zeta(-g^*_p, -g^*_r)$. Let $Q \in \mathbb{R}^{d_1 \times d_1}$ be a suitable Householder transformation mapping $p \in \mathbb{R}^{d_1}$ to a vector of zeros except for one entry that is equal to $\|p\|$. Let $g^*_{\|p\|}$ denote the p-component of the conjugate gradient at $(\|p\|, r) \in \text{int}(K^*_{\text{rpower}})$. Since the function ζ is invariant to orthonormal transformations on the first input,

$$f(u, w) = f(Qu, w). \tag{6.75}$$

It is also easy to see from the definition of the dual gradient in (6.2) that this implies:

$$g^*_p = Q^\top g^*_{Qp}. \tag{6.76}$$

From (6.73a):

$$g^*_p = \frac{p}{2}. \tag{6.77}$$

Due to (6.76) and the invariance of ζ to transformation by $Q^\top$:

$$\zeta = \zeta(-g^*_p, -g^*_r) = \zeta(-Q^\top g^*_{Qp}, -g^*_r) = \zeta(-g^*_{\|p\|}, -g^*_r) = \frac{2g^*_{\|p\|}}{\|p\|}. \tag{6.78}$$

Substituting into (6.77):

$$g^*_p = \frac{g^*_{\|p\|} \cdot p}{\|p\|}, \tag{6.79}$$

where $g^*_{\|p\|} \in \mathbb{R}_{\geq}$ can be computed as for the $p \in \mathbb{R}_{\geq}$ case.

Suppose $p \in \mathbb{R}_{\geq}$ from hereon. Due to (6.73a), we need:

$$\zeta = \frac{2g^*_p}{p}. \tag{6.80}$$

Since $\zeta > 0$, we have $g^*_p > 0$. From (6.73b):

$$r_i = \frac{2\alpha_i \varphi'(-g^*_r)}{-g^*_{r_i}} - \frac{1-\alpha_i}{g^*_{r_i}} \qquad \forall i \in [d_2], \tag{6.81a}$$

$$\Leftrightarrow \; r_i(-g^*_{r_i})\zeta = 2\alpha_i \varphi(-g^*_r) + (1-\alpha_i)\zeta \qquad \forall i \in [d_2]. \tag{6.81b}$$

Applying φ to both sides, across all i:

$$\varphi(r)\varphi(-g^*_r)\zeta^2 = \varphi(2\alpha_i\varphi(-g^*_r) + (1-\alpha_i)\zeta). \tag{6.82}$$

Substituting for $\varphi(-g^*_r) = \zeta + {g^*_p}^2 = \frac{2g^*_p}{p} + {g^*_p}^2$ and the expression for ζ from (6.80):

$$\varphi(r)\Big(\frac{2g^*_p}{p} + {g^*_p}^2\Big)\Big(\frac{2g^*_p}{p}\Big)^2 = \varphi\Big(2\alpha_i\Big(\frac{2g^*_p}{p} + {g^*_p}^2\Big) + (1-\alpha_i)\frac{2g^*_p}{p}\Big). \tag{6.83}$$

We treat this as a root-finding problem for g^*_p. Taking the logarithm of both sides in (6.83) and rearranging, we would like to find a root for $h(g^*_p) = 0$, where $p, g^*_p > 0$. Due to Lemma 6.3.6, this can be done with a quadratically convergent Newton-Raphson method. Due to (6.81b), the solution can be used to compute g^*_r with:

$$g^*_{r_i} = -\frac{1}{r_i}(2\alpha_i\varphi(-g^*_r) + (1-\alpha_i)\zeta) = -\frac{{}_i(1+pg^*_p)+1}{r_i} \qquad 8i \in [d_2], \tag{6.84}$$

which shows (6.36). Combining (6.69) with (6.84) gives (6.37).

Although not necessary for (6.36) and (6.37), let us derive an upper bound on g^*_p. Computationally, we find this bound to provide a good initial estimate while solving $h(g^*_p) = 0$. From (6.81a), we have that:

$$r_i g^*_{r_i} = -\frac{2\ {}_i{}'(-g^*_r)}{} - (1-\alpha_i) \qquad 8i \in [d_2] \tag{6.85a}$$

$$) \ \sum_{j\in[d_2]:j\neq i} -r_j g^*_{r_j}\alpha_i = \sum_{j\in[d_2]:j\neq i} \frac{2\ {}_i\ {}_j{}'(-g^*_r)}{} + \sum_{j\in[d_2]:j\neq i}\alpha_i(1-\alpha_j) \quad 8i \in [d_2]. \tag{6.85b}$$

Multiplying (6.85a) by $(1-\alpha_i)$ and adding (6.85b), we get:

$$(1-\alpha_i)r_i g^*_{r_i} - \sum_{j\in[d_2]:j\neq i}\alpha_i r_j g^*_{r_j} = -1 + d_2\alpha_i \qquad 8i \in [d_2]. \tag{6.86}$$

Combining (6.86) with (6.1):

$$r_i g^*_{r_i} + \frac{-1+d_2 \ _i-(1-\ _i)r_i g^*_{r_i}}{-\ _i} + pg^*_p = -d_2 - 1 \qquad 8i \ 2 \ \mathrm{J}d_2\mathrm{K} \qquad (6.87a)$$

$$) \ -g^*_{r_i} = \frac{_i+1+pg^*_p \ _i}{r_i} \qquad 8i \ 2 \ \mathrm{J}d_2\mathrm{K}. \qquad (6.87b)$$

Next, since $-g^* \ 2 \ \mathrm{int}(K_{\mathrm{rpower}})$, we have that

$$g^*_p < \sqrt{\varphi(-g^*_r)} \qquad (6.88a)$$

$$= \sqrt{\varphi\big(\tfrac{+1+\ pg^*_p}{r}\big)} \qquad (6.88b)$$

$$= \not{p}\tfrac{1}{'(r=\alpha)}\sqrt{\varphi\big(1+\alpha^{-1}+pg^*_p\big)} \qquad (6.88c)$$

$$\leq \not{p}\tfrac{1}{'(r=\alpha)}\textstyle\sum_{i2\mathrm{J}d_2\mathrm{K}}(\alpha_i+1+\alpha_i pg^*_p) \qquad (6.88d)$$

$$= \not{p}\tfrac{1}{'(r=\alpha)}(1+d_2+pg^*_p) \qquad (6.88e)$$

$$) \ g^*_p \leq \not{p}\tfrac{1+d_2}{'(r=\alpha)-p}. \qquad (6.88f)$$

The second inequality comes from the weighted version of the arithmetic-mean-geometric-mean inequality [Chares, 2009, Page 128].

□

6.5.4 Infinity norm cone

Proof of Lemma 6.3.8. Let $\not{y} < 0$ denote the negative root of:

$$h(y) := py + \textstyle\sum_{i2\mathrm{J}d\mathrm{K}}\sqrt{1+r_i^2y^2} + 1. \qquad (6.89)$$

The derivatives of h are:

$$h^{\not{0}}(y) = p + y\textstyle\sum_{i2\mathrm{J}d\mathrm{K}} r_i^2(1+r_i^2y^2)^{-1=2}, \qquad (6.90a)$$

$$h^{\not{0}\not{0}}(y) = \textstyle\sum_{i2\mathrm{J}d\mathrm{K}} r_i^2(1+r_i^2y^2)^{-3=2} \geq 0. \qquad (6.90b)$$

Once again, h can have at most one root η on the halfline $y \leq 0$ since:

$$h'(y) \geq p + y \sum_{i \in [d]} p \frac{r_i^2}{r_i^2 y^2} \tag{6.91a}$$

$$= p + y \sum_{i \in [d]} \frac{r_i^2}{|y||r_i|} \tag{6.91b}$$

$$= p - \sum_{i \in [d]} |r_i| \tag{6.91c}$$

$$> 0. \tag{6.91d}$$

The last line follows from $(p, r) \in \text{int}(K^*_1)$. Since h is increasing and convex, a root-finding Newton-Raphson method will converge quadratically from any $y_+ \geq \eta$ [Süli and Mayers, 2003, Theorem 1.9]. Consider the function:

$$h(y) := y\left(p - \sum_{i \in [d]} |r_i|\right) + 1 \tag{6.92a}$$

$$= py + \sum_{i \in [d]} |r_i||y| + 1 \tag{6.92b}$$

$$\leq h(y). \tag{6.92c}$$

The root of h is at $y = -(p - \sum_{i \in [d]} |r_i|)^{-1}$. Since h is increasing in y, we may use this for y_+. Alternatively, we could use:

$$h(y) := py + d + 1 \leq h(y), \tag{6.93}$$

and its root gives $y_+ = -\frac{d+1}{p}$. □

Proof of Proposition 6.3.9. Let $(u, w) \in \text{int}(K_1)$ and define $\zeta_i(u, w) := u^2 - w_i^2$ for all $i \in [d]$. Then the gradient of f is:

$$g_u = \frac{d-1}{u} - \sum_{i \in [d]} \frac{2u}{\zeta_i(u;w)}, \tag{6.94a}$$

$$g_{w_i} = \frac{2w_i}{\zeta_i(u;w)} \qquad \forall i \in [d]. \tag{6.94b}$$

We would like to find $g^* := g^*(p, r)$ such that $-g(-g^*(p, r)) = (p, r)$. Let us fix

$\zeta_i := g^*_p{}^2 - g^*_{r_i}{}^2$ for all $i \in \mathrm{J}d\mathrm{K}$. From (6.94b), for all $i \in \mathrm{J}d\mathrm{K}$:

$$r_i = \frac{2g^*_{r_i}}{\zeta_i} \tag{6.95a}$$

$$\implies \tfrac{1}{2}(g^*_p{}^2 - g^*_{r_i}{}^2) r_i = g^*_{r_i}. \tag{6.95b}$$

This implies the signs of r_i and $g^*_{r_i}$ equal for all $i \in \mathrm{J}d\mathrm{K}$, and:

$$g^*_{r_i} = \begin{cases} 0 & r_i = 0, \\ \frac{\sqrt{1 + r_i^2 g^*_p{}^2} - 1}{r_i} & r_i \neq 0. \end{cases} \tag{6.96}$$

Substituting into the definition for ζ_i, for all $i \in \mathrm{J}d\mathrm{K}$:

$$\zeta_i = \begin{cases} g^*_p{}^2 & r_i = 0, \\ \frac{-2 + 2\sqrt{1 + r_i^2 g^*_p{}^2}}{r_i^2} & r_i \neq 0, \end{cases} \tag{6.97}$$

From (6.94a):

$$d - 1 - \textstyle\sum_{i \in \mathrm{J}d\mathrm{K}} \frac{2g^*_p{}^2}{\zeta_i} = p g^*_p \tag{6.98a}$$

$$\iff d - 1 - \textstyle\sum_{i \in \mathrm{J}d\mathrm{K}: r_i \neq 0} \frac{2g^*_p{}^2 r_i^2}{-2 + 2\sqrt{1 + r_i^2 g^*_p{}^2}} + \sum_{i \in \mathrm{J}d\mathrm{K}: r_i = 0} 2 = p g^*_p \tag{6.98b}$$

$$\iff d - 1 - \textstyle\sum_{i \in \mathrm{J}d\mathrm{K}: r_i \neq 0} \frac{g^*_p{}^2 r_i^2 (-1 - \sqrt{1 + r_i^2 g^*_p{}^2})}{1 - (1 + r_i^2 g^*_p{}^2)} + \sum_{i \in \mathrm{J}d\mathrm{K}: r_i = 0} 2 = p g^*_p \tag{6.98c}$$

$$\iff p g^*_p + \textstyle\sum_{i \in \mathrm{J}d\mathrm{K}} \sqrt{1 + r_i^2 g^*_p{}^2} + 1 = 0. \tag{6.98d}$$

We treat (6.98d) as a root-finding problem in the variable $g^*_p < 0$, which can be easily solved due to Lemma 6.3.8. The expression for g^*_r is obtained from (6.96). □

6.6 Proofs of inverse Hessian operators

For $K, K^* \subset \mathbb{R}^{1+d}$, where K is either K_{hpower} or K_{rpower} parametrized by α, let $\mathfrak{w} = (u, w) \in \operatorname{int}(K)$, and $\mathfrak{x} = (x, z) \in \mathbb{R}^{1+d}$ be an arbitrary direction. Due to [Nesterov

and Todd, 1997, Equation (2.11)]:

$$H(u)^{-1} \cdot x = H^*(-g(u)) \cdot x. \tag{6.99}$$

Therefore to derive the inverse Hessian operator $H(u)^{-1} \cdot x$, we may derive an expression for $H^*(p) \cdot x$, for arbitrary $p = (p, r)$ 2 int(K^*), and substitute $-g(u)$ for p. In a practical implementation of a PDIPM, $g(u)$ is usually already available at the time when $H(u)^{-1}$ is needed. Note that:

$$H^*(p) \cdot x = \frac{d}{dt} g^*(p + tx)\big|_{t=0}. \tag{6.100}$$

For convenience, we let $p(t) = p + tx$ and we let $g^*(t)$ denote the conjugate gradient at $p(t)$. We use $'$ to denote derivatives with respect to the linearization variable t, i.e. $H^*(p) \cdot x = g^{*\prime}(0)$.

6.6.1 Hypograph power cone

Proof of Proposition 6.4.1. Differentiating (6.63) and (6.64) at $p(t)$ with respect to t gives rise to the nonlinear system (which we wish to solve for $g^{*\prime}(t)$):

$$x = (\varphi(-g^*_r(t)) + g^*_p(t))^{-2}(\tfrac{d}{dt}\varphi(-g^*_r(t)) + g^{*\prime}_p(t)), \tag{6.101a}$$

$$z_i = g^*_{r_i}(t)^{-2} g^{*\prime}_{r_i}(t)(-p\alpha_i\varphi(-g^*_r(t)) + 1) - g^*_{r_i}(t)^{-1}\alpha_i(-x\varphi(-g^*_r(t)) - p\tfrac{d}{dt}\varphi(-g^*_r(t))) \quad 8i \in \mathrm{Jd}\mathsf{K}. \tag{6.101b}$$

From the chain rule:

$$\tfrac{d}{dt}\varphi(-g^*_r(t)) = \varphi(-g^*_r(t)) \textstyle\sum_{i \in \mathrm{Jd}\mathsf{K}} \alpha_i (g^*_{r_i}(t))^{-1} g^{*\prime}_{r_i}(t). \tag{6.102}$$

From hereon let us drop the variable t from our notation for brevity. Define:

$$K := \textstyle\sum_{i \in \mathrm{Jd}\mathsf{K}} \alpha_i g^*_{r_i}{}^{-1} g^{*\prime}_{r_i}. \tag{6.103}$$

From (6.101b), for all $i \in [d]$:

$$g^*_{r_i}{}^{(1)} = g^*_{r_i}{}^{2} \cdot \frac{z_i + g^*_{r_i}{}^{-1} \, {}_i(-x'(-g^*_r) - p'(-g^*_r)K)}{-p_i'(-g^*_r) + 1}. \tag{6.104}$$

Multiplying (6.104) by $\alpha_i g^*_{r_i}{}^{-1}$ and summing over all $i \in [d]$ gives:

$$K = \sum_{i \in [d]} \frac{z_i \, {}_i g^*_{r_i} + \frac{2}{i}'(-g^*_r)(-x - pK)}{-p_i'(-g^*_r) + 1} \tag{6.105a}$$

$$\Rightarrow K = \Big(1 + \varphi(-g^*_r)p \sum_{j \in [d]} \frac{\frac{2}{j}}{1 - p_j'(-g^*_r)}\Big)^{-1} \sum_{i \in [d]} \frac{z_i \, {}_i g^*_{r_i} + \frac{2}{i}'(-g^*_r)(-x)}{1 - p_i'(-g^*_r)}. \tag{6.105b}$$

We remark that for the uniform $\alpha = e/d$ case, K takes the simple form:

$$K = d^{-1}\Big(\sum_{i \in [d]} z_i g^*_{r_i} - x\varphi(-g^*_r)\Big). \tag{6.106}$$

From (6.101a) and (6.104) we can obtain $g^*_p{}^{(1)}$ and $g^*_r{}^{(1)}$ in closed form. Let:

$$k_{1,i} := 1 - \alpha_i \varphi(-g^*_r)p \qquad \forall i \in [d], \tag{6.107a}$$

$$k_2 := \sum_{i \in [d]} \frac{\frac{2}{i}}{k_{1,i}}, \tag{6.107b}$$

$$k_3 := 1 + \varphi(-g^*_r)pk_2. \tag{6.107c}$$

Then from (6.101) and (6.104) (multiplication between vectors below should be interpreted elementwise):

$$g^*_p{}^{(1)} = (\varphi(-g^*_r) + g^*_p)^2 x - K\varphi(-g^*_r) \tag{6.108a}$$

$$= \big(\zeta(-g^*_p, -g^*_r)^2 + \frac{k_2 g^*_p{}^2}{k_3}\big)x - \frac{'(-g^*_r)}{k_3}\langle z, \frac{g^*_r}{k_0}\rangle, \tag{6.108b}$$

$$g^*_{r_i}{}^{(1)} = g^*_{r_i}{}^{2} \cdot \frac{z_i + g^*_{r_i}{}^{-1} \, {}_i(-x'(-g^*_r) - p'(-g^*_r)K)}{k_{1,i}} \tag{6.108c}$$

$$= \frac{g^*_{r_i}{}^2}{k_{1,i}} z_i - \frac{{}_i g^*_{r_i}}{k_{1,i}} \frac{'(-g^*_r)}{k_3} x - \frac{p'(-g^*_r)}{k_3}\langle z, \frac{g^*_r}{k_0}\rangle \frac{{}_i g^*_{r_i}}{k_{1,i}} \qquad \forall i \in [d]. \tag{6.108d}$$

Applying (6.99) gives the desired result. □

6.6.2 Radial power cone

Proof of Proposition 6.4.2. Let:

$$K := \sum_{i \in [d_2]} \alpha_i (g^*_{r_i}(t))^{-1} g^*_{r_i}(t). \tag{6.109}$$

For φ defined in (6.31), we have that:

$$\begin{aligned} \frac{d}{dt}\varphi(-g^*_r(t)) &= \sum_{i \in [d_2]} \varphi(-g^*_r(t)) \cdot 2\alpha_i (-g^*_{r_i}(t))^{-1} (-g^*_{r_i}(t)) && \text{(6.110a)} \\ &= 2\varphi(-g^*_r) K. && \text{(6.110b)} \end{aligned}$$

We fix $\zeta := \zeta(-g^*_p, -g^*_r)$ for convenience. We start by differentiating (6.80) at $p(t)$ with respect to a linearization variable t, which we omit for brevity:

$$x_i = 2\Big(\frac{g^*_{p_i}}{} - \frac{g^*_{p_i}}{2}(2\varphi(-g^*_r)K - 2\langle g^*_p, g^*_p \rangle)\Big) \qquad \forall i \in [d_1]. \tag{6.111}$$

We use (6.111) to obtain a new expression for $\langle g^*_p, g^*_p \rangle$:

$$\begin{aligned} \langle x, g^*_p \rangle &= 2\Big(\frac{\langle g^*_p, g^*_p \rangle}{} - \frac{\|g^*_p\|^2}{2}(2\varphi(-g^*_r)K - 2\langle g^*_p, g^*_p \rangle)\Big) && \text{(6.112a)} \\ \Leftrightarrow \langle g^*_p, g^*_p \rangle &= \frac{2}{2k_1}\Big(\langle x, g^*_p \rangle + \frac{4\|g^*_p\|^2}{2} \cdot \varphi(-g^*_r)K\Big), && \text{(6.112b)} \end{aligned}$$

where we define:

$$k_1 := \varphi(-g^*_r) + \|g^*_p\|^2. \tag{6.113}$$

Next, we differentiate (6.81b), for all $i \in [d_2]$:

$$\begin{aligned} &- z_i g^*_{r_i} \zeta - r_i g^*_{r_i} \zeta - 2r_i g^*_{r_i}(\varphi(-g^*_r)K - \langle g^*_p, g^*_p \rangle) = \\ &4\alpha_i \varphi(-g^*_r)K + 2(1-\alpha_i)(\varphi(-g^*_r)K - \langle g^*_p, g^*_p \rangle), \end{aligned} \tag{6.114}$$

whence, for all $i \in \mathcal{Jd}_2\mathcal{K}$ (replacing $\langle g^*_p, g^{*0}_p \rangle$ using (6.112b)):

$$-r_i g^*_{r_i}{}^0 \zeta = 4\alpha_i \varphi(-g^*_r) K + 2(1 - \alpha_i + r_i g^*_{r_i})(\varphi(-g^*_r) K - \langle g^*_p, g^{*0}_p \rangle) + z_i g^*_{r_i} \zeta \tag{6.115a}$$

$$= 4\alpha_i \varphi(-g^*_r) K - 4\alpha_i \frac{\varphi'(-g^*_r)}{k_1} (\varphi(-g^*_r) K - \langle g^*_p, g^{*0}_p \rangle) + z_i g^*_{r_i} \zeta \tag{6.115b}$$

$$= \frac{2\alpha_i \varphi'(-g^*_r)}{k_1} \left(2 g^*_p k^2 K + \zeta \langle g^*_p, x \rangle\right) + z_i g^*_{r_i} \zeta. \tag{6.115c}$$

In the second equality we use the identity, for all $i \in \mathcal{Jd}_2\mathcal{K}$:

$$1 - \alpha_i + r_i g^*_{r_i} = \alpha_i \left(\langle p, -g^*_p \rangle - 2\right) = -2\alpha_i \left(\frac{g^*_p k^2}{} + 1\right) = -2\alpha_i \frac{\varphi'(-g^*_r)}{}, \tag{6.116}$$

which can be derived by the same logic as in (6.87), then using (6.80). In the third equality we substitute (6.112b).

Substituting into $g^*_{r_i}{}^0$ from (6.115c) into (6.109):

$$K = -\sum_{i \in \mathcal{Jd}_2\mathcal{K}} \alpha_i (g^*_{r_i})^{-1} \frac{2\alpha_i \varphi'(-g^*_r) k_1^{-1} \left(2 g^*_p k^2 K + \langle g^*_p, x \rangle\right) - z_i (-g^*_{r_i})}{r_i} \tag{6.117a}$$

$$= \frac{-2k_2 k_1^{-1} \varphi'(-g^*_r) \langle x, g^*_p \rangle - \langle \alpha, z \rangle}{1 + 4k_2 k_1^{-1} \varphi'(-g^*_r) g^*_p k^2}, \tag{6.117b}$$

where:

$$k_2 := \langle r, g^*_r \rangle. \tag{6.118}$$

We now replace $\langle g^*_p, g^{*0}_p \rangle$ using (6.112b) and substitute for K in (6.111) to obtain $g^*_p{}^0$. Define for convenience:

$$k_3 := \frac{k_1}{2\varphi'(-g^*_r)} + 2k_2 \frac{g^*_p k^2}{}, \tag{6.119}$$

then:

$$g^*_p{}^0 = \zeta \left(\frac{x}{2} + \frac{g^*_p}{2} (2\varphi(-g^*_r) K - 2\langle g^*_p, g^{*0}_p \rangle)\right) \tag{6.120a}$$

$$= \frac{}{2} x - \frac{g^*_p}{k_3} \left(\frac{2k_2 \varphi'(-g^*_r) + k_3}{k_1} \langle x, g^*_p \rangle + \langle r, z \rangle\right). \tag{6.120b}$$

Finally, substituting for K in (6.115c) and rearranging, for all i 2 Jd_2K:

$$g^*_{r_i}{}^{0} = -\frac{g^*_{r_i}}{r_i} z_i + \frac{i}{k_3 r_i}\Big(- hx, g^*_p i + \frac{2kg^*_p k^2}{} h\ /r, zi\Big). \tag{6.121}$$

Applying (6.99) to (6.120) and (6.121) gives the desired result. □

Chapter 7

Computational value of conjugate gradients

In the first part of this chapter we empirically compare our conjugate gradient procedures from Chapter 6 with a generic Newton method (in terms of numerical accuracy and computational speed at numerically challenging points). This is done independently to any interior point method. In the second part of this chapter our aim is to empirically compare Hypatia's default algorithm with the algorithm described by Dahl and Andersen [2021] (that we occasionally refer to as *MOSEK's algorithm* for short) together with the conjugate gradient oracles from Chapter 6. In this part of the chapter, we describe our Julia implementation of MOSEK's algorithm (as a stepping procedure in Hypatia).[1] Our implementation includes a minor enhancement. We give expressions for additional oracles required by the cones studied in Chapter 6 for this implementation. We describe a test set of problems and report on the numerical stability and iteration counts from both algorithms. We provide some concluding remarks and opportunities for future work in the final part of the chapter.

[1] The full algorithm implemented by the MOSEK solver is not transparent to us and in this chapter we only make comparisons between our implementations and not the MOSEK solver.

7.1 Comparison with a generic approach

Conjugate gradients can always be evaluated by solving (6.2) using Newton's method, which we refer to as the *generic approach*. Although it should be unsurprising that for $K_{\log}$, K_{logdet}, K_{hgeom}, K_{rtdet}, and K_{rgeom}, the closed form conjugate gradient expressions are more numerically stable and computationally cheaper to evaluate than the generic approach, this may not be obvious for K_{hpower}, K_{rpower}, $K_{\text{`}_1}$, and $K_{\text{`}_{\text{spec}}}$, where a numerical procedure is still needed. In this section we compare the numerical properties and computational e□cacy of the conjugate gradient procedures from Sections 6.3.1 to 6.3.4 against the generic approach. We test both approaches by evaluating conjugate gradients at randomly generated points in the dual cone. Since the procedures for K_{logdet}, K_{rtdet}, and $K_{\text{`}_{\text{spec}}}$ mirror those of $K_{\log}$, $K_{\text{hgeom}}$, and $K_{\text{`}_1}$, we omit the matrix variants for brevity.

We implement the generic approach as follows. The objective in (6.2) is self-concordant. As described by Nesterov et al. [2018, Section 5.2], the objective can be optimized by combining the damped Newton's method with the full Newton's method. In particular, the Hessian H of f induces a local norm at any $w \in K$ that is given by $\|z\|_w = \sqrt{\langle z, H(w)^{-1}z\rangle}$ for all z. When the local norm of the gradient of the objective, that is $\|r + g(w)\|_w$ is less than $\frac{3-\sqrt{5}}{2}$, Newton's method begins to converge quadratically. We stop iterating when the local norm of the gradient is lower than some tolerance ϵ. In our experiments we set ϵ to 1000 times machine zero. Alternatively, we stop if we detect insu□cient progress between consecutive iterations (this happens when numerical error causes Newton's method to stall). The procedure is summarized in Algorithm 1.

For each of the cones $K_{\log}$, K_{hgeom}, K_{hpower}, K_{rgeom}, K_{rpower}, we generate a random dual point by generating a random r_i variable in $(0, 1)$ for $i \in \{1, \ldots, d\}$ and for $K_{\log}$, we also generate a random p variable. We generate the epigraph variable q by letting the relative violation on the epigraph or hypograph inequality in the cone equal some o□set fraction $o \in (0, 1)$. For example, for $K_{\log}$, given randomly generated r and p,

Algorithm 1 Newton's method for computing a conjugate gradient.

procedure compute $g^*(r)$
Require: r, ϵ, w_0, oracles for $g(w)$, $H(w)^{-1}$
  $w \leftarrow w_0$
  $\lambda_1 \leftarrow \sqrt{\langle g(w) + r, H(w)^{-1}(g(w) + r)\rangle} = \sqrt{\nu - 2\langle w, r\rangle + \langle r, H(w)^{-1}r\rangle}$
  $j \leftarrow 1$
  while $\lambda_j > \epsilon$ **do**
    if $\lambda_j > \frac{3-\sqrt{5}}{2}$ **then**
      $\alpha \leftarrow (1 + \lambda_j)^{-1}$ ▷ damped Newton's method
    else
      $\alpha \leftarrow 1$ ▷ full Newton's method
    end if
    $w \leftarrow w - \alpha H(w)^{-1}(g(w) + r)$
    $\lambda_j \leftarrow \sqrt{\langle g(w) + r, H(w)^{-1}(g(w) + r)\rangle} = \sqrt{\nu - 2\langle w, r\rangle + \langle r, H(w)^{-1}r\rangle}$
    if $\lambda_j > \frac{3-\sqrt{5}}{2}$ and $\lambda_j > 1000\left(\frac{j-1}{1-(j-1)}\right)^2$ **then**
      break ▷ slow progress
    end if
    $j \leftarrow j + 1$
  end while
  return $-w$
end procedure

we let:

$$q = p\textstyle\sum_{i \in [\![d]\!]} \log(-\frac{r_i}{p}) + pd, \tag{7.1a}$$

$$q = q \times (1 + \operatorname{sgn}(q) \times o), \tag{7.1b}$$

which ensures $(p, q, r) \in \operatorname{int}(K^*_{\log})$ and o controls proximity to the boundary. At each dual point, we measure violation on (6.1):

$$|\langle g^*(z), z\rangle + \nu|, \tag{7.2}$$

where $g^*(z)$ is obtained by one of two procedures. Note that the term inside the absolute value of (7.2) should be positive at $-g^*(z) \in K$, but could be negative from numerical approximation. To measure computational cost, we count the number of Newton iterations taken by the generic approach and if applicable, the number of steps taken by the Newton-Raphson methods from Lemmas 6.3.3, 6.3.6 and 6.3.8.

In Table 7.1 we give the average number of iterations from each approach over 10 randomly generated points (and randomly generated α for K_{hpower} and K_{rpower}). Recall that an iteration of the generic Newton's method requires applying an inverse Hessian matrix of side dimension d. It is clear that each iteration of the generic Newton's method is much more expensive then an iteration of a univariate Newton-Raphson method or evaluating the conjugate gradient in closed form. Additionally, in all experiments the generic Newton's method requires many more iterations than the univariate Newton-Raphson method, and the number of iterations for the generic approach grows as d increases and the offset from the boundary o decreases. For the specialized conjugate gradient procedures, the iteration counts remain fairly constant (usually 2–6) for different d. See Table 7.1 for details. We also plot the violation on (7.2) against number of Newton iterations in Figure 7-1, for the first randomly generated dual point. The final violation from the specialized methods is marked by an 'o'. Unlike the specialized methods, more iterations of the damped Newton method are generally required as the offset from the cone boundary decreases. Finally, in Table 7.2 we show the mean value of (7.2) across 10 randomly generated points. In general, the residuals are similar or slightly more favorable (i.e. lower) for the specialized methods, and in a few cases the multivariate Newton method didn't converge (from numerical issues). The code for these experiments can be found at `https://github.com/lkapelevich/conjgradexperiments`. In our implementation we take the Newton-Raphson step in higher precision (using Julia's `BigFloat` number type), which is relatively inexpensive (since it only involves elementary operations on scalars and not any matrices or vectors) and particularly helpful near the end of the Newton-Raphson method for $K_{\backprime_1}$.

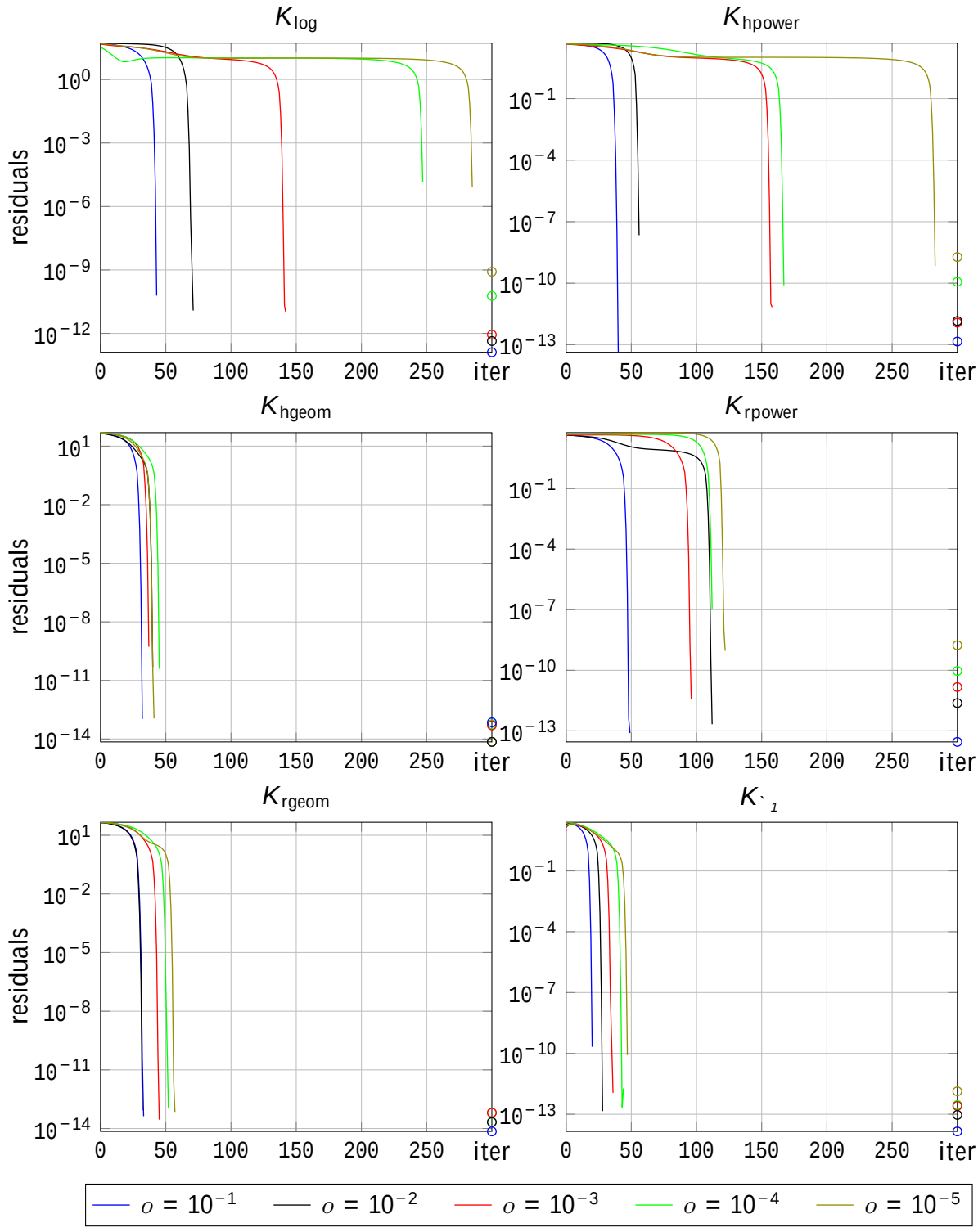

Figure 7-1: Value of (7.2) against iterations of damped and full Newton's method at the first randomly generated dual point with $d = 60$.

d	o	$K_{\log}$	K_{hpower}		K_{hgeom}	K_{rpower}		K_{rgeom}	$K_{\grave{}_1}$	
		g	g	s	g	g	s	g	g	s
20	10^{-5}	98.	88.	2.0	27.	88.	3.0	27.	41.	2.0
	10^{-4}	84.	75.	2.0	27.	74.	3.0	27.	36.	3.0
	10^{-3}	70.	61.	3.0	27.	60.	3.0	27.	30.	4.0
	10^{-2}	56.	46.	3.0	27.	46.	4.0	26.	24.	5.0
	10^{-1}	40.	31.	4.0	25.	31.	4.0	24.	17.	5.0
40	10^{-5}	196.	130.	2.0	37.	127.	3.0	37.	46.	3.0
	10^{-4}	160.	108.	2.0	37.	105.	3.0	37.	40.	4.0
	10^{-3}	124.	85.	3.0	37.	83.	3.0	37.	34.	5.0
	10^{-2}	90.	63.	3.0	37.	62.	3.4	36.	27.	6.0
	10^{-1}	57.	41.	4.0	33.	41.	4.0	33.	19.	5.0
60	10^{-5}	301.	218.	2.0	46.	211.	3.0	45.	50.	3.1
	10^{-4}	240.	170.	2.0	46.	164.	3.0	45.	44.	4.3
	10^{-3}	180.	124.	3.0	46.	120.	3.0	45.	38.	5.5
	10^{-2}	124.	83.	3.0	45.	81.	3.0	44.	30.	6.0
	10^{-1}	76.	50.	4.0	40.	50.	4.0	39.	21.	5.0

Table 7.1: Mean number of iterations using a generic Newton method (g), and when applicable, univariate Newton-Raphson from specialized methods (s) over 10 random points.

d	o	$K_{\log}$		K_{hpower}		K_{hgeom}		K_{rpower}		K_{rgeom}		$K_{\grave{}_1}$	
		g	s	g	s	g	s	g	s	g	s	g	s
20	10^{-5}	-8.7	-9.6	-9.5	-9.7	-14.	-14.	-9.5	-9.9	-13.	-14.	-11.	-11.
	10^{-4}	-9.9	-10.	-11.	-11.	-13.	-14.	-11.	-11.	-13.	-14.	-12.	-12.
	10^{-3}	-11.	-11.	-12.	-12.	-14.	-14.	-12.	-12.	-13.	-14.	-13.	-13.
	10^{-2}	-12.	-12.	-13.	-13.	-14.	-14.	-13.	-13.	-14.	-14.	-13.	-14.
	10^{-1}	-13.	-13.	-13.	-14.	-14.	-14.	-14.	-14.	-14.	-14.	-14.	-14.
40	10^{-5}	-8.1	-9.3	-9.2	-9.2	-13.	-14.	-9.1	-9.4	-13.	-14.	-11.	-11.
	10^{-4}	-9.2	-9.9	-10.	-10.	-14.	-14.	-10.	-10.	-13.	-14.	-12.	-12.
	10^{-3}	-10.	-11.	-11.	-11.	-14.	-14.	-11.	-12.	-13.	-14.	-12.	-13.
	10^{-2}	-11.	-12.	-12.	-12.	-13.	-14.	-12.	-12.	-14.	-14.	-13.	-14.
	10^{-1}	-12.	-13.	-13.	-14.	-14.	-14.	-13.	-13.	-14.	-14.	-13.	-14.
60	10^{-5}	-0.5	-8.4	-9.1	-9.0	-13.	-13.	-9.1	-9.0	-13.	-13.	-11.	-11.
	10^{-4}	-9.0	-9.5	-10.	-10.	-13.	-13.	-10.	-10.	-13.	-13.	-12.	-12.
	10^{-3}	-9.7	-11.	-11.	-11.	-13.	-13.	-11.	-11.	-13.	-14.	-12.	-13.
	10^{-2}	-11.	-12.	-12.	-12.	-13.	-13.	-12.	-12.	-13.	-13.	-13.	-13.
	10^{-1}	-12.	-12.	-13.	-13.	-13.	-13.	-13.	-13.	-13.	-14.	-13.	-14.

Table 7.2: $\log_{10}$ of mean value of (7.2) using a generic Newton method (g) and specialized methods (s) over 10 random points.

7.2 Practical implementation considerations

We give a high-level description of MOSEK's algorithm and describe some of the practical considerations we make while implementing the algorithm in Hypatia's framework. In some places, our implementation deviates from the algorithm described by Dahl and Andersen [2021].

7.2.1 Summary of MOSEK's algorithm

We start by describing the high-level steps of the algorithm proposed by Dahl and Andersen [2021]. We use the same notation as in Chapter 2. Recall that in each iteration of Hypatia's algorithm, search directions are computed by solving (2.25) with different RHSs. In MOSEK's algorithm, (2.25) takes the form:

$$E\delta = r_E, \tag{7.3a}$$

$$\delta_{\bar{z};k} + H_k \delta_{\bar{s};k} = r_k \quad \forall k \in \{1, \ldots, K\}. \tag{7.3b}$$

Critically, $\mu H_k(s_k)$ is replaced by H_k in the final equation. H_k is a *scaling matrix*.

Recall that the scaling matrices MOSEK's algorithm uses are based on a technique by Myklebust and Tunçel [2014], where low-rank updates are applied to the Hessian or inverse Hessian, to obtain a matrix H satisfying:

$$Hs = z, \tag{7.4a}$$

$$Hg^*(z) = g(s). \tag{7.4b}$$

Dahl and Andersen [2021] use:

$$H = \mu H(s) + \tfrac{1}{2}(v_1 v_2^\top + v_2 v_1^\top) - \mu \frac{\left(H g^*(z)\right)\left(H g^*(z)\right)^\top - \left(H^{-1} g(s)\right)\left(g^*(z)\right)^\top}{\left\langle g^*(z), H g^*(z)\right\rangle - \left\langle H^{-1} g(s), g^*(z)\right\rangle}, \tag{7.5}$$

where:

$$v_1 = z + \mu g(s), \tag{7.6a}$$

$$v_2 = z - \mu g(s) + \frac{1}{-1\hbar g(s);g^*(z)i-1} v_1, \tag{7.6b}$$

$$v_3 = -Hg^*(z) + \nu^{-1}\hbar g(s), g^*(z)ig(s). \tag{7.6c}$$

If our problem includes symmetric cones, we use the Nesterov Todd scaling point to obtain H instead, in the manner described by Vandenberghe [2010].

A high-level description of the *stepper* proposed by Dahl and Andersen [2021] can be summarized as follows (with terminology from Chapter 2). At each iterate ω:

1. Solve (7.3) for a centering direction, δ^c, with $r_E = 0$ and $r_k = -z_k - \mu g_k(s_k)$ for all $k \in \mathrm{J}K\mathrm{K}$

2. Solve (7.3) for a prediction direction, δ^p, with $r_E = -E\omega$ and $r_k = -z_k$ for all $k \in \mathrm{J}K\mathrm{K}$

3. Solve (7.3) for a prediction adjustment direction, δ^{pt}, with $r_E = 0$ and $r_k = \frac{1}{2} r^3 f_k\left[\delta^p_{s;k}, (H_k(s_k))^{-1}\delta^p_{z;k}\right]$ for all $k \in \mathrm{J}K\mathrm{K}$

4. Compute (to tolerance) the maximum stepping distance to the cone boundary α_p in the direction δ^p and use it to obtain a centering factor $\gamma = (1-\alpha_p)\min\{(1-\alpha_p)^2, 0.25\}$

5. For the combined direction $\delta := (1-\gamma)\delta^p + \delta^{pt} + \gamma\delta^c$, estimate the maximum stepping distance $\hat{\alpha} := \max\{\alpha : \pi_?(\omega^{i-1} + \alpha\delta) \geq \beta\}$, where, $\pi_?(\omega) := \max_{k\in\mathrm{J}\bar{K}\mathrm{K}}\{\pi_{?;k}(\omega)\}$, and,

$$\pi_{?;k}(\omega) := \begin{cases} \frac{k}{(!)\hbar g_k(\bar{s}_k);g_k^*(\bar{z}_k)i} & \text{if } \mu(\omega) > 0, s_k \in \mathrm{int}\left(K_k\right), z_k \in \mathrm{int}\left(K_k^*\right), \\ 1 & \text{otherwise} \end{cases} \tag{7.7}$$

6. Update the current iterate as $\omega + \hat{\alpha}\delta$

Let us highlight some key differences to Hypatia's algorithm. There is no centering adjustment direction δ^{ct} and the combined direction δ is computed from δ^p, δ^{pt} and δ^c by a different heuristic (with different weights on each direction). The directions δ^p, δ^{pt} and δ^c are computed from a different LHS (due to (7.3)) and additionally, δ^{pt} uses a different RHS to Hypatia. In particular, this new RHS requires evaluating $r^3 f_k\left[\delta^p_{s;k}, (H_k(s_k))^{-1}\delta^p_{z;k}\right]$, which requires derivation (in Hypatia, we apply the third order derivative to a single direction and this leads to some simplification). The value α_p is not calculated using any neighborhood and instead relies on estimating distances to the cone boundary. Finally, the neighborhood used to compute the final step length $\hat{\alpha}$ is different to $\pi_{\mathcal{N}_2}$ and $\pi_{\mathcal{N}_1}$ that we define for Hypatia's algorithm in Chapter 2. The function $\pi_?$ requires checking dual feasibility and calculating the conjugate gradient for each cone.

We implement the above procedure as a *stepper* in Hypatia (the *combined* stepper from Chapter 2 is the default stepper in Hypatia).[2] Our implementation includes a few differences that simplify the steps above and improve the counts of iterations or robustness on Hypatia's test problems. Instead of doing two separate linesearches for α_p and $\hat{\alpha}$, we do a curve search similar to Section 2.4.5. The curve search is carried out over the parameter $1 - \gamma$ and similar to Section 2.4.5, we check different values in the schedule A. The combined direction with curve search that we use is:

$$\delta = (1 - \gamma^{1=3})\big((1 - \gamma)\delta^p + \delta^{pt} + \gamma\delta^c\big). \tag{7.8}$$

Whereas curve search mainly improved iteration counts in Section 2.7, using curve search with MOSEK's algorithm increases robustness and allows us to solve more of Hypatia's (continuous integration) test instances. Additionally, this strategy avoids the need to estimate the distance to the cone boundary (that is needed for α_p), which can be expensive for some cones. The term $1 - \gamma^{1=3}$ is a heuristic that reverses MOSEK's choice of $\gamma = (1-\alpha_p)\min\{(1-\alpha_p)^2, 0.25\}$ from item 4 and does not restrict the amount of centering applied. In our implementation we use $\beta = 0.01$.

[2] The code is at https://github.com/lkapelevich/Hypatia.jl/tree/conjgrads2.

7.2.2 Third order oracles for different directions

As described in the previous section, calculating δ^{pt} requires evaluating the term $r^3 f_k\left[\delta^p_{s;k}, (H_k(s_k))^{-1}\delta^p_{z;k}\right]$, which is not one of the oracles that Hypatia uses. We provide some simplified expressions for $T := r^3 f[\delta^1, \delta^2]$, for arbitrary directions δ^1 and δ^2, to help implement MOSEK's algorithm for some of the cones studied in Chapter 6 (T here is redefined from Chapter 3).

Logarithm and power-like cones

For $K_{\log}$, K_{logdet}, K_{hpower}, and K_{rtdet}, the term T can be inferred by deriving a more general version for the barrier $f(u, v, w) = -\log(u - v\varphi(w/v)) - \text{logdet}(w)$, where φ is a function on a cone of squares of a Euclidean Jordan algebra, as in Chapter 3. In other words, let us generalize our expressions from (3.49) for two different directions. For K_{hpower} and K_{rtdet}, terms involving the perspective variable should be ignored. Let φ be either the sum of logaithms or the power-mean function, (u, v, w) an interior point, and:

$$\begin{aligned}
\xi(\delta) &:= v^{-1}(\delta_w - \delta_v v^{-1} w),\\
\sigma &:= \varphi(w/v) - r\,\varphi[w/v],\\
\chi(\delta) &:= \delta_u - \delta_v \sigma - r\,\varphi[\delta_w],\\
\tau &:= \tfrac{1}{2}\zeta^{-1}\big(r^2\varphi[\xi(\delta^1)(\zeta^{-1}\chi(\delta^2) + \delta^2_v/v) + \xi(\delta^2)(\zeta^{-1}\chi(\delta^1) + \delta^1_v/v)] -\\
&\qquad r^3\varphi[\xi(\delta^1), \xi(\delta^2)]\big).
\end{aligned}$$

Then:

$$\begin{aligned}
T_u &= -2\zeta^{-3}\chi(\delta^1)\chi(\delta^2) - v\zeta^{-2} r^2\varphi[\xi(\delta^1), \xi(\delta^2)],\\
T_v &= 2T_u\sigma + 2\langle\tau, w/v\rangle - \zeta^{-1} r^2\varphi[\xi(\delta^2), \xi(\delta^2)] - 2q^2 v^{-3},\\
T_w &= 2T_u\, r\,\varphi - 2\tau - 2P(w^{-1/2})(P(w^{-1/2})\delta^1_w P(w^{-1/2})\delta^2_w).
\end{aligned}$$

Radial power cone

Let $\varphi(w) = \prod_{i \in [d]} w^{2\ i}$, $(u, w) \in \text{int}(K_{\text{rpower}})$, and:

$$c_1 := 4(2\frac{\varphi'(w)}{\varphi(w)} - 1)h_{\overline{w}}, \delta_w^1 \rangle h_{\overline{w}}, \delta_w^1 \rangle + 2\langle h_{\overline{w}}, \delta_w^2 \rangle + 2\sum_{i \in [d]} \alpha_i \frac{\frac{1}{w,i}\ \frac{2}{w,i}}{w_i^2}, \tag{7.11a}$$

$$c_2 := 2\big(h_{\overline{w}}, \delta_w^1 \rangle h u, \delta_u^2 \rangle + h_{\overline{w}}, \delta_w^2 \rangle h u, \delta_u^1 \rangle\big), \tag{7.11b}$$

$$c_3 := 4\frac{\langle hu;\ \frac{1}{u} \rangle hu;\ \frac{2}{u} \rangle}{\varphi(w)} + h\delta_u^1 . \delta_u^2 \rangle, \tag{7.11c}$$

where division between vectors should be interpreted componentwise. Then:

$$T_u = 2\zeta^{-2}\big(\varphi(w)(w(c_1 - \frac{4c_2}{\varphi(w)}) - 2h_{\overline{w}}, \delta_w^1 \rangle \delta_u^2 \rangle - 2h_{\overline{w}}, \delta_w^2 \rangle \delta_u^1 \rangle) + 2(c_3 w + hu, \delta_u^1 \rangle \delta_u^2 \rangle + hu, \delta_u^2 \rangle \delta_u^1 \rangle)\big), \tag{7.12a}$$

$$T_w = 2\big(\frac{\varphi'(w)}{\varphi(w)}(1 - \frac{\varphi'(w)}{\varphi(w)})\frac{}{w}(c_1 + \frac{2}{w}(h_{\overline{w}}, \delta_w^1 \rangle \delta_w^2 \rangle + h_{\overline{w}}, \delta_w^2 \rangle \delta_w^1 \rangle)) + (\frac{-1}{w} - 2\frac{\varphi'(w)}{\varphi(w)}\frac{}{w})\frac{\frac{1}{w}}{w}\frac{\frac{2}{w}}{w} + 2\frac{\varphi'(w)}{\varphi(w)}\frac{}{w}(\frac{\varphi'(w)+u}{\varphi(w)}c_2 + w^{-1}(hu, \delta_u^1 \rangle \delta_w^2 \rangle + hu, \delta_u^2 \rangle \delta_w^1 \rangle) - c_3)\big). \tag{7.12b}$$

Infinity norm cone

Let $(u, w) \in \text{int}(K_{\infty,1})$ and:

$$\mu := \frac{w}{u}, \tag{7.13a}$$

$$c_{1;i} := \zeta^{-1}(\delta_{w;i}^2 \mu_i - \delta_u^2)(\delta_u^1 - \delta_{w;i}^1 \mu_i) + \frac{1}{2u}(\delta_u^1 \delta_u^2 - \delta_{w;i}^1 \delta_{w;i}^2) \qquad \forall i \in [d], \tag{7.13b}$$

$$c_{2;i} := -\frac{1}{2u}(\delta_u^1 \delta_{w;i}^2 + \delta_u^2 \delta_{w;i}^1) \qquad \forall i \in [d]. \tag{7.13c}$$

Then:

$$T_u = 2\delta_u^1 \delta_u^2 \frac{(d-1)}{u^3} + 2\sum_{i \in [d]} \zeta_i^{-2}(c_{1;i} + \mu_i c_{2;i} + \frac{\frac{1}{u}\ \frac{2}{u}}{u}), \tag{7.14a}$$

$$T_{w;i} = 2\zeta_i^{-2}(c_{2;i} - \mu_i(c_{1;i} - \frac{\frac{1}{w,i}\ \frac{2}{w,i}}{u})) \qquad \forall i \in [d]. \tag{7.14b}$$

Spectral norm cone

Let $(u, W) \in \text{int}(K_{\text{spec}})$ (ignoring Hypatia's vectorization for W) and:

$$Z := uI - W^\top W, \tag{7.15a}$$
$$\tau := Z^{-1}W, \tag{7.15b}$$
$$c_1 := \delta_W^1 \tau \delta_W^2 + \delta_W^2 \tau \delta_W^1, \tag{7.15c}$$
$$c_2 := \delta_W^1 (W^\top Z^{-1} W + I) \delta_W^2, \tag{7.15d}$$
$$c_3 := \delta_u^1 \delta_W^2 W^\top Z^{-1} + \delta_u^2 \delta_W^1 W^\top Z^{-1}, \tag{7.15e}$$
$$c_4 := \delta_u^1 Z^{-1} \delta_W^2 + \delta_u^2 Z^{-1} \delta_W^1. \tag{7.15f}$$

Then:

$$T_u = 2\big(\delta_u^1 \delta_u^2 (6u\,\text{tr}(Z^{-2}) - 8u^3\,\text{tr}(Z^{-3}) + \tfrac{d_1 - 1}{u^3}) + \tag{7.16a}$$
$$\langle h c_3, 4u^2 Z^{-1}\tau - \tau i - 2uh\delta_W^2, Z^{-2}\delta_W^1 (W^\top \tau + I)\rangle + \tag{7.16b}$$
$$(Z^{-1}\delta_W^1 \tau^\top + \tau \delta_W^1 Z^{-1} + Z^{-1}\tau \delta_W^{1\top})\tau i\big), \tag{7.16c}$$
$$T_w = 2\big(\delta_u^1 \delta_u^2 (8u^2 Z^{-3} W - 2Z^{-2} W) + \tag{7.16d}$$
$$Z^{-1}(-2u(c_3 + c_3^\top) + c_2 + c_2^\top)\tau + Z^{-1}(-2uc_4 + c_1)(W^\top \tau + I) + \tag{7.16e}$$
$$\tau(c_1 + c_1^\top)(W^\top \tau + I) + \tau(c_1^\top - 2uc_4)\tau\big). \tag{7.16f}$$

7.2.3 Implementing an ℓ_1 norm cone

Among K_{log}, K_{hpower}, and K_{rpower}, the primal and dual cones have similar modeling power. However $K_{\ell_1}^*$ models a very different set to K_{ℓ_1}. Recall that Hypatia is able to support both ℓ_1 and ℓ_1 norm constraints using the oracles of K_{ℓ_1}. In other words, if $K_{\ell_1}^*$ is the kth cone in K, then $K_k \in K_{\text{pr}}$ from Section 2.4.1. Analogous treatment in MOSEK's algorithm would mean (7.3b) looks like:

$$\delta_{s;k} + H_k(z_k)\delta_{z;k} = r_k, \tag{7.17}$$

if $K_k = K_{\ell_1}$. Empirically, we see better performance by implementing $K^*_{\ell_1} = K_{\ell_1}$ as a *primal cone* i.e. treat it as a cone in K_{pr}. This is straightforward using various oracles we already derived for K_{ℓ_1} in these last two chapters. The gradient oracle for K_{ℓ_1} is the conjugate gradient oracle for K_{ℓ_1}. The Hessian oracle may be obtained using (6.99). T is also easy to calculate, using:

$$\nabla^3 f(s)[\delta^1, \delta^2] = H(s) \cdot \nabla^3 f^*(-g(s))[H(s)\delta^1, H(s)\delta^2], \tag{7.18}$$

which is derived from (6.99). Analogously, the nuclear norm cone can be supported as a primal cone, and we refer to it by K_{nuc} in Table 7.3.

7.2.4 Test problems

We compare the performance of Hypatia's default algorithm (the *combined* stepper from Chapter 2) with an implementation of MOSEK's algorithm as a stepper in Hypatia. Our primary interest is in comparing iteration counts between the two steppers, and the stability of the algorithms (the number of times a certificate is returned successfully) to empirically compare the quality of search directions. For this reason we do not report on solve times (and our implementation of MOSEK's algorithm has not been optimized for computational speed). We are only interested in problems that include cones studied in Chapter 6. As previously argued, problem instances with at least one cone that requires a generic multivariate Newton method for the g^* oracle would be better suited to Hypatia's algorithm than MOSEK's. We use two sets of test problems. The first set includes all CBLIB [Friberg, 2018] problems with at least one three-dimensional exponential cone or a three-dimensional power cone (all other problems in the CBLIB library include only symmetric cones). If integer variables are present, we solve the convex relaxation. Overall, 204 test instances are retained. The second set includes all the test instances from Section 2.7.1 that include a combination of cones studied in Chapter 6 and symmetric cones. Again, we exclude instances that include only symmetric cones. Making comparisons on those instances is not didactic and it is expected that MOSEK's algorithm with symmetric search directions would

perform better than Hypatia's combined stepper. Overall, 105 test instances from Hypatia's benchmark set are retained, and at least one of the stepping procedures converges in 100 of the instances. We summarize the number of instances and cone types from each example (that has at least some instances meeting our criteria on cone types) in Table 7.3.

In Figure 7-2 we plot histograms of the logarithm of the number of iterations taken on converged instances for Hypatia's default stepper (Hyp) and the MOSEK stepper (Mos). For both the CBLIB test set and Hypatia's benchmark set, the histogram corresponding to MOSEK's algorithm is shifted to left compared to Hypatia's default stepper. In Tables 7.4 and 7.5 we give the number of converged instances, the number of *nearly converged* instances (this is a status that is returned by Hypatia), and shifted geometric means of counts of iterations (using the same groupings as in Section 2.7.2 to handle convergence failures and the shifted geometric mean formula (2.41)). On the CBLIB instances, both steppers solve the same set of problems. On Hypatia's benchmark set, our implementation of MOSEK's algorithm results in 16 fewer converged instances than Hypatia's default stepper. Five of these failing instances include nuclear norm constraints. We expect that an implementation of the adjustment term for the spectral norm cone that is based on a singular value decomposition (analogous to the implementation very recently proposed by Coey [2022]) instead of (7.16) would remedy these failures and slightly reduce the gap in number of solved instances. Overall, our implementation of MOSEK's algorithm gives lower geometric means of iteration counts and fewer iterations on the majority of instances, but is less reliable on Hypatia's benchmark problems than the combined stepper. The improvement in iteration counts is around 18%–22% in the *every* set.

example	#	cones in at least one instance
central polynomial matrix	12	$K_{\geq}$ K K_{rpower} K_{rtdet} $K_{\log}$ K_{logdet}
classical-quantum capacity	4	$K_{\geq}$ K $K_{\log}$
covariance estimation	9	$K_{\geq}$ K K_{rpower} K_{rtdet} $K_{\log}$
density estimation	12	$K_{\geq}$ K K_{hgeom} $K_{\log}$
discrete maximum likelihood	3	$K_{\geq}$ K_{hpower} $K_{\log}$
D-optimal design	13	$K_{\geq}$ K $K_{`_1}$ $K_{`_2}$ K_{hgeom} K_{rtdet} $K_{\log}$ K_{logdet}
experiment design	8	$K_{\geq}$ K K_{rpower} K_{rtdet} $K_{\log}$ K_{logdet}
matrix completion	8	$K_{\geq}$ K $K_{`_{\text{spec}}}$ K_{nuc} K_{rpower} K_{hgeom} $K_{\log}$
matrix regression	3	K $K_{`_1}$ $K_{`_2}$ K_{sqr} K_{nuc}
maximum volume hypercube	8	$K_{`_1}$ $K_{`_1}$ $K_{`_2}$ K_{hgeom}
nonparametric distribution	6	$K_{\geq}$ K_{hgeom} $K_{\log}$
portfolio	4	$K_{`_1}$ $K_{`_1}$
robust geometric programming	3	$K_{\geq}$ $K_{\log}$
shape constrained regression	2	K $K_{`_1}$
signomial minimization	2	$K_{\geq}$ $K_{\log}$
sparse principal components	3	K $K_{`_1}$

Table 7.3: For each example, the count of instances and list of exotic cones (defined in Section 2.5) used in at least one instance (and at least one stepping procedure converged).

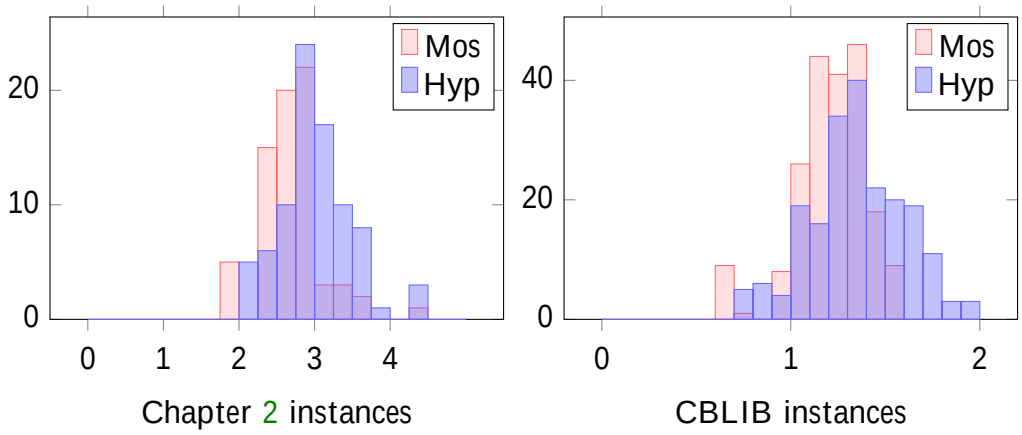

Figure 7-2: Overlayed histograms of $\log_{10}$ of iteration counts, excluding instances that fail to converge.

			geomean iters		
	count converged	count nearly	every	this	all
Hyp	100	0	21.9	21.9	21.9
Mos	84	16	17.4	17.4	21.0

Table 7.4: For each algorithm, the number of converged instances, nearly converged instances, and geometric mean of iteration counts using 100 of Hypatia's benchmark problems.

			geomean iters		
	count converged	count nearly	every	this	all
Hyp	202	1	22.3	22.3	22.5
Mos	202	1	16.2	16.2	16.3

Table 7.5: For each algorithm, the number of converged instances, nearly converged instances, and geometric mean of iteration counts using 204 CBLIB problems.

7.3 Conclusions and future work

The stepper using MOSEK's search directions performed very well on problems with three-dimensional exponential and power cones, and had some advantages for other problems with cones from Chapter 6. An interesting question is whether we can use ideas from those search directions to improve iteration counts in problems for other exotic cones. We were not able to make progress towards e□cient procedures for evaluating the conjugate gradient of several classes of cones- including Hypatia's separable spectral function cone, the LMI cone, the dual cone of sum-of-squares polynomials, or the sparse PSD matrix cone with general sparsity patterns. Indeed, e□cient procedures for the conjugate gradients of these cones would give us e□cient procedures for evaluating the corresponding conjugate barriers, which are not known in the literature. It may be possible to create hybrid stepping strategies if the cones in the problem are a combination of cones that admit e□cient conjugate gradients and cones that do not. That is, the components of the search direction corresponding to cones with e□ciently computable conjugate gradients could be calculated as in MOSEK's algorithm, and those that do not could be computed di□erently. However, such a strategy would lose some of the properties proved by Dahl and Andersen [2021], such as ensuring the violations on the residuals gap and complementarity gap decrease at the same rate. In addition, since the neighborhood definitions in Hypatia's algorithm and MOSEK's algorithm are well suited for their respective search directions, a hybrid strategy may require choosing di□erent step lengths α for di□erent cones, which makes the choice of step length on the x and y variables ambiguous.

Bibliography

Akshay Agrawal, Steven Diamond, and Stephen Boyd. Disciplined geometric programming. *Optimization Letters*, 13(5):961–976, 2019.

Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical programming*, 95(1):3–51, 2003.

Erling D Andersen, Cornelis Roos, and Tamas Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Mathematical Programming*, 95(2):249–277, 2003.

Martin Andersen, Joachim Dahl, Zhang Liu, Lieven Vandenberghe, S Sra, S Nowozin, and SJ Wright. Interior-point methods for large-scale cone programming. *Optimization for Machine Learning*, 5583, 2011.

Martin S Andersen, Joachim Dahl, and Lieven Vandenberghe. Logarithmic barriers for sparse matrix cones. *Optimization Methods and Software*, 28(3):396–423, 2013.

Erin M Aylward, Sleiman M Itani, and Pablo A Parrilo. Explicit SOS decompositions of univariate polynomial matrices and the Kalman-Yakubovich-Popov lemma. In *2007 46th IEEE Conference on Decision and Control*, pages 5660–5665. IEEE, 2007.

Erin M Aylward, Pablo A Parrilo, and Jean-Jacques E Slotine. Stability and robustness analysis of nonlinear systems via contraction metrics and SOS programming. *Automatica*, 44(8):2163–2170, 2008.

Michel Baes. Convexity and di□erentiability properties of spectral functions and spectral mappings on Euclidean Jordan algebras. *Linear algebra and its applications*, 422(2-3):664–700, 2007.

Ahron Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, volume 2. SIAM, 2001.

Je□ Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

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

Brian Borchers. CSDP, a C library for semidefinite programming. *Optimization Methods and Software*, 11(1-4):613–623, 1999.

Stephen Boyd. EE363 review session 4: Linear matrix inequalities. University Lecture, 2009. URL https://stanford.edu/class/ee363/sessions/s4notes.pdf.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. SIAM, 1994.

Samuel Burer. Semidefinite programming in the space of partial positive semidefinite matrices. *SIAM Journal on Optimization*, 14(1):139–172, 2003.

John Burkardt. Polynomials for global optimization tests, 2016. URL https://people.sc.fsu.edu/~jburkardt/py_src/polynomials/polynomials.html. Accessed: 2021-03-22.

Eric Carlen. Trace inequalities and quantum entropy: an introductory course. *Entropy and the quantum*, 529:73–140, 2010.

Venkat Chandrasekaran and Parikshit Shah. Relative entropy relaxations for signomial optimization. *SIAM Journal on Optimization*, 26(2):1147–1173, 2016.

Venkat Chandrasekaran and Parikshit Shah. Relative entropy optimization and its applications. *Mathematical Programming*, 161(1-2):1–32, 2017.

Robert Chares. Cones and interior-point algorithms for structured convex optimization involving powers andexponentials, 2009.

Yanqing Chen, Timothy A Davis, William W Hager, and Sivasankaran Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3): 1–14, 2008.

Chris Coey. *Interior point and outer approximation methods for conic optimization*. PhD thesis, Massachusetts Institute of Technology, 2022.

Chris Coey, Lea Kapelevich, and Juan Pablo Vielma. Conic optimization with spectral functions on Euclidean Jordan algebras. *arXiv preprint arXiv:2103.04104*, 2021a.

Chris Coey, Lea Kapelevich, and Juan Pablo Vielma. Hypatia cones reference. 2021b.

Chris Coey, Lea Kapelevich, and Juan Pablo Vielma. Performance enhancements for a generic conic interior point algorithm. *arXiv preprint arXiv:2107.04262*, 2021c.

Chris Coey, Lea Kapelevich, and Juan Pablo Vielma. Solving natural conic formulations with Hypatia.jl. *arXiv preprint arXiv:2005.01136*, 2021d.

Robert M Corless and David J Je□rey. The wright ω function. In *Artificial intelligence, automated reasoning, and symbolic computation*, pages 76–89. Springer, 2002.

Joachim Dahl and Erling D Andersen. A primal-dual interior-point algorithm for nonsymmetric exponential-cone optimization. *Mathematical Programming*, pages 1–30, 2021.

Alexandre d'Aspremont, Laurent El Ghaoui, Michael I Jordan, and Gert RG Lanckriet. A direct formulation for sparse pca using semidefinite programming. *SIAM review*, 49(3):434–448, 2007.

Chandler Davis. All convex invariant functions of Hermitian matrices. *Archiv der Mathematik*, 8(4):276–278, 1957.

Chun Yuan Deng. A generalization of the Sherman–Morrison–Woodbury formula. *Applied Mathematics Letters*, 24(9):1561–1564, 2011.

Steven Diamond and Stephen Boyd. CVXPY: a Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

Andrew C Doherty, Pablo A Parrilo, and Federico M Spedalieri. Complete family of separability criteria. *Physical Review A*, 69(2):022308, 2004.

Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

Alexander Domahidi, Eric Chu, and Stephen Boyd. ECOS: an SOCP solver for embedded systems. In *2013 European Control Conference (ECC)*, pages 3071–3076. IEEE, 2013.

Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: a modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

Jacques Faraut and Adam Koranyi. Analysis on symmetric cones. *Bull. Amer. Math. Soc*, 35:77–86, 1998.

Hamza Fawzi and Omar Fawzi. E□cient optimization of the quantum relative entropy. *Journal of Physics A: Mathematical and Theoretical*, 51(15):154003, 2018.

Hamza Fawzi, James Saunderson, and Pablo A. Parrilo. Semidefinite approximations of the matrix logarithm. *Foundations of Computational Mathematics*, 2018. Package cvxquad at https://github.com/hfawzi/cvxquad.

Hamza Fawzi, James Saunderson, and Pablo A Parrilo. Semidefinite approximations of the matrix logarithm. *Foundations of Computational Mathematics*, 19(2):259–296, 2019.

Leonid Faybusovich. Self-concordant barriers for cones generated by chebyshev systems. *SIAM Journal on Optimization*, 12(3):770–781, 2002.

Leonid Faybusovich and Takashi Tsuchiya. Matrix monotonicity and self-concordance: how to handle quantum entropy in optimization problems. *Optimization Letters*, 11(8):1513–1526, 2017.

Leonid Faybusovich and Cunlu Zhou. Long-step path-following algorithm for quantum information theory: Some numerical aspects and applications. *Numerical Algebra, Control & Optimization*, 2021.

Philip J Fleming and John J Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *Communications of the ACM*, 29(3):218–221, 1986.

Henrik A Friberg. CBLIB 2014: A benchmark library for conic mixed-integer and continuous optimization. *Mathematical Programming Computation*, 8(2):191–214, 2016.

Henrik A. Friberg. The conic benchmark format version 3. Technical report, 2018.

Takayuki Furuta. Concrete examples of operator monotone functions obtained by an elementary method without appealing to Löwner integral representation. *Linear algebra and its applications*, 429(5-6):972–980, 2008.

Yves Genin, Yvan Hachez, Yu Nesterov, and Paul Van Dooren. Optimization problems over positive pseudopolynomial matrices. *SIAM Journal on Matrix Analysis and Applications*, 25(1):57–79, 2003.

Nicholas Gould and Jennifer Scott. A note on performance profiles for benchmarking software. *ACM Transactions on Mathematical Software (TOMS)*, 43(2):1–5, 2016.

Michael Grant and Stephen Boyd. CVX: MATLAB software for disciplined convex programming, version 2.1, 2014.

Michael Grant, Stephen Boyd, and Yinyu Ye. Disciplined convex programming. In *Global optimization*, pages 155–210. Springer, 2006.

Osman Güler. Barrier functions in interior point methods. *Mathematics of Operations Research*, 21(4):860–885, 1996.

Osman Güler and Levent Tunçel. Characterization of the barrier parameter of homogeneous convex cones. *Mathematical programming*, 81(1):55–76, 1998.

Georgina Hall. Engineering and business applications of sum of squares polynomials. *arXiv preprint arXiv:1906.07961*, 2019.

Didier Henrion and Milan Korda. Convex computation of the region of attraction of polynomial control systems. *IEEE Transactions on Automatic Control*, 59(2): 297–312, 2013.

Didier Henrion and J-B Lasserre. Convergent relaxations of polynomial matrix inequalities and static output feedback. *IEEE Transactions on Automatic Control*, 51(2):192–202, 2006.

Michael E Ho□man and William Douglas Withers. Generalized Chebyshev polynomials associated with a□ne Weyl groups. *Transactions of the American Mathematical Society*, 308(1):91–104, 1988.

Camile WJ Hol and Carsten W Scherer. Sum of squares relaxations for polynomial semidefinite programming. In *Proc. Symp. on Mathematical Theory of Networks and Systems (MTNS), Leuven, Belgium*. Citeseer, 2004.

Lea Kapelevich, Chris Coey, and Juan Pablo Vielma. Sum of squares generalizations for conic sets. *arXiv preprint arXiv:2103.11499*, 2021.

Lea Kapelevich, Erling D Andersen, and Juan Pablo Vielma. Computing conjugate barrier information for nonsymmetric cones. *arXiv preprint arXiv:2201.04121*, 2022.

Mehdi Karimi and Levent Tunçel. Domain-Driven Solver (DDS) Version 2.0: a MATLAB-based software package for convex optimization problems in domain-driven form, 2020.

Masakazu Kojima. *Sums of squares relaxations of polynomial semidefinite programs*. Inst. of Technology, 2003.

Masakazu Kojima and Masakazu Muramatsu. An extension of sums of squares relaxations to polynomial optimization problems over symmetric cones. *Mathematical Programming*, 110(2):315–336, 2007.

Milan Korda, Didier Henrion, and Colin N Jones. Controller design and value function approximation for nonlinear dynamical systems. *Automatica*, 67:54–66, 2016.

Man Kam Kwong. Some results on matrix monotone functions. *Linear Algebra and Its Applications*, 118:129–153, 1989.

Yann Labit, Dimitri Peaucelle, and Didier Henrion. SeDuMi interface 1.02: a tool for solving LMI problems with SeDuMi. In *Proceedings. IEEE International Symposium on Computer Aided Control System Design*, pages 272–277. IEEE, 2002.

Jean B Lasserre. Homogeneous functions and conjugacy. *Journal of Convex Analysis*, 5:397–404, 1998.

Monique Laurent and Teresa Piovesan. Conic approach to quantum graph parameters using linear optimization over the completely positive semidefinite cone. *SIAM Journal on Optimization*, 25(4):2461–2493, 2015.

Benoît Legat, Christopher Coey, Robin Deits, Joey Huchette, and Amelia Perry. Sum-of-squares optimization in Julia. In *JuMP Developers Meetup/Workshop*, 2017.

Benoit Legat, Oscar Dowson, Joaquim Dias Garcia, and Miles Lubin. MathOptInterface: a data structure for mathematical optimization problems. *arXiv preprint arXiv:2002.03447*, 2020.

Adrian S Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analysis*, 2(1):173–183, 1995.

Karl Löwner. Über monotone matrixfunktionen. *Mathematische Zeitschrift*, 38(1): 177–216, 1934.

Miles Lubin and Iain Dunning. Computing in Operations Research using Julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015. doi: 10.1287/ijoc.2014.0623.

Rahul Mazumder, Arkopal Choudhury, Garud Iyengar, and Bodhisattva Sen. A computational framework for multivariate convex regression and its variants. *Journal of the American Statistical Association*, 114(525):318–331, 2019.

Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

MOSEK ApS. Modeling Cookbook revision 3.2.2, 2020. URL https://docs.mosek.com/modeling-cookbook/index.html.

Riley Murray, Venkat Chandrasekaran, and Adam Wierman. Signomial and polynomial optimization via relative entropy and partial dualization. *Mathematical Programming Computation*, pages 1–39, 2020.

Tor Myklebust and Levent Tunçel. Interior-point algorithms for convex optimization based on primal-dual metrics. *arXiv preprint arXiv:1411.2129*, 2014.

Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. Studies in Applied Mathematics. Society for Industrial and Applied Mathematics, 1994.

Yu E Nesterov and Michael J Todd. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations research*, 22(1):1–42, 1997.

Yu E Nesterov and Michael J Todd. Primal-dual interior-point methods for self-scaled cones. *SIAM Journal on optimization*, 8(2):324–364, 1998.

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

Yuri Nesterov. Towards non-symmetric conic optimization. *Optimization Methods and Software*, 27(4-5):893–917, 2012.

Yurii Nesterov, Michael J Todd, and Yinyu Ye. Infeasible-start primal-dual methods and infeasibility detectors for nonlinear programming problems. *Mathematical Programming*, 84(2):227–268, 1999.

Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

Dominique Orban. BenchmarkProfiles.jl, 2019. URL https://doi.org/10.5281/zenodo.4630955.

David Papp and Farid Alizadeh. Semidefinite characterization of sum-of-squares cones in algebras. *SIAM Journal on Optimization*, 23(3):1398–1423, 2013.

Dávid Papp and Farid Alizadeh. Shape-constrained estimation using nonnegative splines. *Journal of Computational and Graphical Statistics*, 23(1):211–231, 2014.

Dávid Papp and Sercan Yıldız. On "A homogeneous interior-point algorithm for nonsymmetric convex conic optimization". *arXiv preprint arXiv:1712.00492*, 2017.

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

Dávid Papp and Sercan Yıldız. alfonso: ALgorithm FOr Non-Symmetric Optimization, 2020. URL https://github.com/dpapp-github/alfonso.

Dávid Papp and Sercan Yıldız. alfonso: Matlab package for nonsymmetric conic optimization. *INFORMS Journal on Computing*, 2021.

Pablo A Parrilo. Chapter 3: Polynomial optimization, sums of squares, and applications. In *Semidefinite Optimization and Convex Algebraic Geometry*, pages 47–157. SIAM, 2012.

Frank Permenter, Henrik A Friberg, and Erling D Andersen. Solving conic optimization problems via self-dual embedding and facial reduction: a unified approach. *SIAM Journal on Optimization*, 27(3):1257–1282, 2017.

Ralph Tyrell Rockafellar. *Convex analysis*. Princeton university press, 2015.

Scott Roy and Lin Xiao. On self-concordant barriers for generalized power cones. *Optimization Letters*, pages 1–14, 2021.

Hristo S Sendov. The higher-order derivatives of spectral functions. *Linear algebra and its applications*, 424(1):240–281, 2007.

Santiago Akle Serrano. *Algorithms for unsymmetric cone optimization and an implementation for problems with the exponential cone*. PhD thesis, Stanford University, 2015.

Anders Skajaa and Yinyu Ye. A homogeneous interior-point algorithm for nonsymmetric convex conic optimization. *Mathematical Programming*, 150(2):391–422, 2015.

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

R Subramanian and KV Bhagwat. On a theorem of Wigner on products of positive matrices. *Proceedings of the Indian Academy of Sciences-Section A. Part 3, Mathematical Sciences*, 88(1):31–34, 1979.

Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

Defeng Sun and Jie Sun. Löwner's operator and spectral functions in Euclidean Jordan algebras. *Mathematics of Operations Research*, 33(2):421–445, 2008.

Yifan Sun and Lieven Vandenberghe. Decomposition methods for sparse matrix nearness problems. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1691–1717, 2015.

David Sutter, Tobias Sutter, Peyman Mohajerin Esfahani, and Renato Renner. E□-cient approximation of quantum channel capacities. *IEEE Transactions on Information Theory*, 62(1):578–598, 2015.

Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. SDPT3—a matlab software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4):545–581, 1999.

Levent Tunçel. Generalization of primal—dual interior-point methods to convex optimization problems in conic form. *Foundations of computational mathematics*, 1 (3):229–254, 2001.

Madeleine Udell, Karanveer Mohan, David Zeng, Jenny Hong, Steven Diamond, and Stephen Boyd. Convex optimization in Julia. In *Proceedings of the 1st First Workshop for High Performance Technical Computing in Dynamic Languages*, pages 18–28. IEEE Press, 2014.

Lieven Vandenberghe. The CVXOPT linear and quadratic cone program solvers, 2010. URL https://www.seas.ucla.edu/~vandenbe/publications/coneprog.pdf.

Manuel V.C. Vieira. *Jordan Algebraic approach to symmetric optimization*. PhD thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, TU Delft, NL 2628 CD, Delft, The Netherlands, November 2007.

Manuel VC Vieira. Derivatives of eigenvalues and Jordan frames. *Numerical Algebra, Control & Optimization*, 6(2):115, 2016.

Xiaojie Xu, Pi-Fang Hung, and Yinyu Ye. A simplified homogeneous and self-dual linear programming algorithm and its implementation. *Annals of Operations Research*, 62(1):151–171, 1996.

Makoto Yamashita, Katsuki Fujisawa, and Masakazu Kojima. Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0). *Optimization Methods and Software*, 18(4):491–505, 2003.

Shuzhong Zhang. A new self-dual embedding method for convex programming. *Journal of Global Optimization*, 29(4):479–496, 2004.