Performance enhancements for a generic conic interior point algorithm

Mathematical Programming Computation
https://doi.org/10.1007/s12532-022-00226-0

Performance enhancements for a generic conic interior point algorithm

Chris Coey[1] · **Lea Kapelevich**[1] · **Juan Pablo Vielma**[2]

Received: 8 July 2021 / Accepted: 29 July 2022

Abstract
In recent work, we provide computational arguments for expanding the class of proper cones recognized by conic optimization solvers, to permit simpler, smaller, more natural conic formulations. We define an exotic cone as a proper cone for which we can implement a small set of tractable (i.e. fast, numerically stable, analytic) oracles for a logarithmically homogeneous self-concordant barrier for the cone or for its dual cone. Our extensible, open-source conic interior point solver, Hypatia, allows modeling and solving any conic problem over a Cartesian product of exotic cones. In this paper, we introduce Hypatia's interior point algorithm, which generalizes that of Skajaa and Ye (Math. Program. 150(2):391–422, 2015) by handling exotic cones without tractable primal oracles. To improve iteration count and solve time in practice, we propose four enhancements to the interior point stepping procedure of Skajaa and Ye: (1) loosening the central path proximity conditions, (2) adjusting the directions using a third order directional derivative barrier oracle, (3) performing a backtracking search on a curve, and (4) combining the prediction and centering directions. We implement 23 useful exotic cones in Hypatia. We summarize the complexity of computing oracles for these cones and show that our new third order oracle is not a bottleneck. From 37 applied examples, we generate a diverse benchmark set of 379 problems. Our computational testing shows that each stepping enhancement improves Hypatia's iteration count and solve time. Altogether, the enhancements reduce the geometric means of iteration count and solve time by over 80% and 70% respectively.

✉ Lea Kapelevich
lkap@mit.edu

Chris Coey
coey@mit.edu

Juan Pablo Vielma
jvielma@google.com

[1] Operations Research Center, MIT, Cambridge, MA, USA

[2] Google Research and MIT Sloan School of Management, Cambridge, MA, USA

Published online: 17 September 2022

Keywords Conic optimization · Logarithmically homogeneous self-concordant barrier functions · Higher-order derivatives · Interior point methods

Mathematics Subject Classification 90-08 · 90C25 · 90C51

1 Introduction

Any convex optimization problem may be represented as a conic problem that minimizes a linear function over the intersection of an affine subspace with a Cartesian product of primitive proper cones (i.e. irreducible, closed, convex, pointed, and full-dimensional conic sets). Under certain conditions, a conic problem has a simple and easily checkable certificate of optimality, primal infeasibility, or dual infeasibility [60]. Most commercial and open-source conic solvers (such as CSDP [8], CVXOPT [3], ECOS [24, 63], MOSEK [42], SDPA [68], Alfonso [56]) implement primal-dual interior point methods (PDIPMs) based on the theory of logarithmically homogeneous self-concordant barrier (LHSCB) functions. Compared to first order conic methods (see [54] on SCS solver), idealized PDIPMs typically exhibit higher per-iteration cost, but have a lower iteration complexity of $\mathcal{O}(\sqrt{\nu}\log(1/\varepsilon))$ iterations to converge to ε tolerance, where ν is the barrier parameter of the LHSCB. We limit the scope of this paper to conic PDIPMs, but note that there are other notions of duality and PDIPMs for convex problems outside of the conic realm (see e.g. [37]).

1.1 Conic optimization with Hypatia solver

Hypatia [20] is an open-source, extensible conic primal-dual interior point solver.[1] Hypatia is written in the Julia language [7] and is accessible through a flexible, low-level native interface or the modeling tool JuMP [25]. A key feature of Hypatia is a generic cone interface that allows users to define new *exotic cones*. An exotic cone is a proper cone for which we can implement a small set of tractable LHSCB oracles (listed in Sect. 1.2) for either the cone or its dual cone. 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. Several cones are new and required the development of LHSCBs and efficient procedures for oracle evaluations (see [18, 35]).

Advanced conic solvers such as MOSEK 9 currently recognize at most only a handful of standard cones: nonnegative, (rotated) second order, positive semidefinite (PSD), and three-dimensional exponential and power cones. In [20], we show across seven applied examples that modeling with the much larger class of exotic cones often permits simpler, smaller, more natural conic formulations. Our computational experiments with these examples demonstrate the potential advantages, especially in terms of solve time and memory usage, of solving natural formulations with Hypatia compared to solving standard conic extended formulations with either Hypatia or

[1] Hypatia is available at github.com/chriscoey/Hypatia.jl under the MIT license. See [19] for documentation, examples, and instructions for using Hypatia.

MOSEK 9. However, a description of Hypatia's algorithms was outside the scope of [20]. In contrast, the main purpose of this paper is to introduce some key features of our exotic conic PDIPM that enable it to be both general and performant.

1.2 The Skajaa-Ye algorithm

Most conic PDIPM solvers use efficient algorithms specialized for symmetric cones, in particular, the nonnegative, (rotated) second order, and PSD cones. Although non-symmetric conic PDIPMs proposed in [47, 50] can handle a broader class of cones, they have several disadvantages compared to specialized symmetric methods (e.g. requiring larger linear systems, strict feasibility of the initial iterate, or conjugate LHSCB oracles). The algorithm by Skajaa and Ye [64], henceforth referred to as *SY*, addresses these issues by approximately tracing the central path of the homogeneous self-dual embedding (HSDE) [2, 67] to an approximate solution for the HSDE. This final iterate provides an approximate conic certificate for the conic problem, if a conic certificate exists. The SY algorithm relies on an idea by Nesterov [47] that a high quality prediction direction (enabling a long step and rapid progress towards a solution) can be computed if the current iterate is in close proximity to the central path (i.e. it is an approximate scaling point). To restore centrality after each prediction step, SY performs a series of centering steps.

By using a different definition of central path proximity to [47], SY avoids needing conjugate LHSCB oracles.[2] Indeed, a major advantage of SY is that it only requires access to a few tractable oracles for the primal cone: an initial interior point, feasibility check, and gradient and Hessian evaluations for the LHSCB. In our experience, for a large class of proper cones, these oracles can be evaluated analytically, i.e. without requiring the implementation of iterative numerical procedures (such as optimization) that can be expensive and may need numerical tuning. Conjugate LHSCB oracles in general require optimization, and compared to the analytic oracles, they are often significantly less efficient and more prone to numerical instability.

1.3 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 [20]). In this case, SY can solve neither the primal problem nor its conic dual, as SY requires primal oracles. Our algorithm generalizes SY to allow a conic formulation over any Cartesian product of exotic cones.

The focus of [64] is demonstrating that SY has the best known iteration complexity for conic PDIPMs. This complexity analysis was corrected by Papp and Yıldız [56], who implemented SY in their recent MATLAB solver Alfonso [58, 59]. It is well known that performant PDIPM implementations tend to violate assumptions used in

[2] Some proposed techniques such as the Hessian scaling updates and central path proximity definitions of [21, 46] require conjugate LHSCB oracles.

iteration complexity analysis, so in this paper 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 in [21, 41] that also use higher order LHSCB information.[3] 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 in [21], and we leverage this for fast and numerically stable evaluations. Whereas the method of [41] only applies to symmetric cones, and the technique in [21] is tested 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 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 paper 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.

[3] To avoid confusion, we do not use the term 'corrector' in this paper. In the terminology of [21, 41] our TOA approach is a type of 'higher order corrector' technique, but also our unadjusted centering direction is referred to by [56, 64] as the 'corrector' direction.

For the purposes of this paper, we only describe and test one set of (default) options for these components.

1.4 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.

1.5 Overview

In Sect. 2, we define our mathematical notation. In Sect. 3, we define exotic cones, LHSCBs, and our required cone oracles (including the TOO). In Sect. 4, we describe Hypatia's general primal-dual conic form, associated conic certificates, and the HSDE. In Sect. 5, we define the central path of the HSDE and central path proximity measures, and we outline Hypatia's high level algorithm. We also derive the prediction and centering directions and our new TOA directions, and we describe the SY-like stepping procedure and our series of four enhancements to this procedure. In Appendices A to B, we discuss advanced procedures for preprocessing and initial point finding, solving structured linear systems for directions, and performing efficient backtracking searches and proximity checks. In Sect. 6, we briefly introduce Hypatia's predefined exotic cones and show that our TOO is relatively cheap to compute, and in Appendix C we describe some TOO implementations. In Sect. 7, we summarize our applied examples and exotic conic benchmark instances, and finally we present our computational results demonstrating the practical efficacy of our stepping enhancements.

2 Notation

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

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

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

For a vector or matrix A, the transpose is A' and the trace is $\mathrm{tr}(A)$. $I(d)$ is the identity matrix in $\mathbb{R}^{d \times d}$. We use the standard inner product on $\mathbb{R}^d$, i.e. $s'z = \sum_{i \in [\![d]\!]} s_i z_i$ for $s, z \in \mathbb{R}^d$, which equips $\mathbb{R}^d$ with the standard norm $\|s\| = (s's)^{1/2}$. The linear operators vec and mat preserve inner products, e.g. $\mathrm{vec}(S)' \mathrm{vec}(Z) = \mathrm{tr}(S'Z)$ for $S, Z \in \mathbb{R}^{d_1 \times d_2}$ or $S, Z \in \mathbb{S}^d$. $\mathrm{Diag} : \mathbb{R}^d \to \mathbb{S}^d$ is the diagonal matrix of a given vector, and $\mathrm{diag} : \mathbb{S}^d \to \mathbb{R}^d$ is the vector of the diagonal of a given matrix. For dimensions implied by context, e is a vector of 1s, e_i is the ith unit vector, and 0 is a vector or matrix of 0s.

$|x|$ 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 $\mathrm{logdet}(X)$ is the log-determinant of $X \succ 0$. For a vector $x \in \mathbb{R}^d$, $\|x\|_\infty = \max_{i \in [\![d]\!]} |x_i|$ is the ℓ_∞ norm and $\|x\|_1 = \sum_{i \in [\![d]\!]} |x_i|$ is the ℓ_1 norm.

Suppose the function $f : \mathrm{int}(\mathcal{C}) \to \mathbb{R}$ is strictly convex and three times continuously differentiable on the interior of a set $\mathcal{C} \subset \mathbb{R}^d$. For a point $p \in \mathrm{int}(\mathcal{C})$, we denote the gradient and Hessian of f at p as $\nabla f(p) \in \mathbb{R}^d$ and $\nabla^2 f(p) \in \mathbb{S}^d_{\succ}$. Given an $h \in \mathbb{R}^d$, the first, second, and third order directional derivatives of f at p in direction h are $\nabla f(p)[h] \in \mathbb{R}$, $\nabla^2 f(p)[h, h] \in \mathbb{R}_{\geq}$, and $\nabla^3 f(p)[h, h, h] \in \mathbb{R}$.

3 Exotic cones and oracles

Let $\mathcal{K}$ be a proper cone in $\mathbb{R}^q$, i.e. a conic subset of $\mathbb{R}^q$ that is closed, convex, pointed, and full-dimensional (see [64]). Note that requiring $\mathcal{K}$ to be a subset of $\mathbb{R}^q$ simplifies our notation but is not restrictive, e.g. for the PSD cone, we use the standard svec vectorization (see Sect. 2). The dual cone of $\mathcal{K}$ is $\mathcal{K}^*$, which is also a proper cone in $\mathbb{R}^q$:

$$\mathcal{K}^* \coloneqq \{z \in \mathbb{R}^q : s'z \geq 0, \forall s \in \mathcal{K}\}. \tag{1}$$

Following [49, Sections 2.3.1 and 2.3.3], $f : \text{int}(\mathcal{K}) \to \mathbb{R}$ is a ν-LHSCB for $\mathcal{K}$, where $\nu \geq 1$ is the *LHSCB parameter*, if it is three times continuously differentiable, strictly convex, satisfies $f(s_i) \to \infty$ along every sequence $s_i \in \text{int}(\mathcal{K})$ converging to the boundary of $\mathcal{K}$, and:

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

$$f(\theta s) = f(s) - \nu \log(\theta) \qquad \forall s \in \text{int}(\mathcal{K}), \theta \in \mathbb{R}_>. \tag{2b}$$

Following [61, Section 3.3], we define the *conjugate* of f, $f^* : \text{int}(\mathcal{K}^*) \to \mathbb{R}$, as:

$$f^*(z) \coloneqq -\inf_{s \in \text{int}(\mathcal{K})}\{s'z + f(s)\}, \tag{3}$$

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

A Cartesian product $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K$ of K proper cones is a proper cone, and its dual cone is $\mathcal{K}^* = \mathcal{K}_1^* \times \cdots \times \mathcal{K}_K^*$. In this case, if f_k is a ν_k-LHSCB for $\mathcal{K}_k$, then $\sum_{k \in [\![K]\!]} f_k$ is an LHSCB for $\mathcal{K}$ with parameter $\sum_{k \in [\![K]\!]} \nu_k$ [49, Proposition 2.3.3]. We call $\mathcal{K}$ a primitive cone if it cannot be written as a Cartesian product of two or more lower-dimensional cones (i.e. K must equal 1). Note $\mathcal{K}^*$ is primitive if and only if $\mathcal{K}$ is primitive. Primitive proper cones are the fundamental building blocks of conic formulations.

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

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

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

$$g(s) \coloneqq \nabla f(s) \in \mathbb{R}^q, \tag{4a}$$

$$H(s) \coloneqq \nabla^2 f(s) \in \mathbb{S}^q_\succ. \tag{4b}$$

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

$$\mathrm{T}(s, \delta_s) \coloneqq -\tfrac{1}{2}\nabla^3 f(s)[\delta_s, \delta_s] \in \mathbb{R}^q. \tag{5}$$

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

In Sect. 6, we introduce Hypatia's predefined cones and discuss the time complexity of computing the feasibility check, gradient, Hessian, and TOO oracles. In Appendix

C, we describe efficient and numerically stable techniques for computing these oracles for a handful of our cones. Although Hypatia's generic cone interface allows specifying additional oracles that can improve speed and numerical performance (e.g. a dual cone feasibility check, Hessian product, and inverse Hessian product), these optional oracles are outside the scope of this paper.

For the initial interior point (which Hypatia only calls once, when finding an initial iterate), we prefer to use the *central point* of $\mathcal{K}$. This is the unique point satisfying $t \in \text{int}(\mathcal{K}) \cap \text{int}(\mathcal{K}^*)$ and $t = -g(t)$ [21]. For the nonnegative cone $\mathcal{K} = \mathbb{R}^n_+$, $f(s) = -\log(s)$ 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.

4 General conic form and certificates

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

$$\begin{aligned} \inf_x \quad & c'x : && \text{(6a)} \\ & b - Ax = 0, && \text{(6b)} \\ & h - Gx \in \mathcal{K}, && \text{(6c)} \end{aligned}$$

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

Once a proper cone $\mathcal{K}_k$ is defined through Hypatia's generic cone interface, both $\mathcal{K}_k$ and $\mathcal{K}_k^*$ may be used in any combination with other cones recognized by Hypatia to construct the Cartesian product cone $\mathcal{K}$ in (6c). The primal form (6) matches CVXOPT's form, however CVXOPT only recognizes symmetric cones [66]. Unlike the conic form used in [59, 64], 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 (6), over variables $y \in \mathbb{R}^p$ and $z \in \mathbb{R}^q$ associated with (6b) and (6c), is:

$$\begin{aligned} \sup_{y,z} \quad & -b'y - h'z : && \text{(7a)} \\ & c + A'y + G'z = 0, && \text{(7b)} \\ & z \in \mathcal{K}^*, && \text{(7c)} \end{aligned}$$

where (7b) is associated with the primal variable $x \in \mathbb{R}^n$.

Under certain conditions, there exists a simple conic certificate providing an easily verifiable proof of infeasibility of the primal (6) or dual (7) problem (via the conic

generalization of Farkas' lemma) or optimality of a given primal-dual solution (via conic weak duality).

A primal improving ray x is a feasible direction for the primal along which the objective improves (hence it certifies dual infeasibility):

$$c'x < 0, \quad -Ax = 0, \quad -Gx \in \mathcal{K}. \tag{8}$$

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

$$-b'y - h'z > 0, \quad A'y + G'z = 0, \quad z \in \mathcal{K}^*. \tag{9}$$

A complementary solution (x, y, z) satisfies the primal-dual feasibility conditions (6b), (6c), (7b) and (7c), and has equal and attained primal and dual objective values:

$$c'x = -b'y - h'z. \tag{10}$$

One of these certificates exists if neither the primal nor the dual is *ill-posed*. Intuitively, according to [42, Section 7.2], a conic problem is ill-posed if a small perturbation of the problem data can change the feasibility status of the problem or cause arbitrarily large perturbations to the optimal solution (see [60] for more details).

The *homogeneous self-dual embedding* (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 [66, Section 6]), derived from a homogenization of the primal-dual optimality conditions (6b), (6c), (7b), (7c) and (10):

$$\begin{bmatrix} 0 \\ 0 \\ s \\ \kappa \end{bmatrix} = \begin{bmatrix} 0 & A' & G' & c \\ -A & 0 & 0 & b \\ -G & 0 & 0 & h \\ -c' & -b' & -h' & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \tau \end{bmatrix}, \tag{11a}$$

$$(z, \tau, s, \kappa) \in \left(\mathcal{K}^* \times \mathbb{R}_{\geq} \times \mathcal{K} \times \mathbb{R}_{\geq}\right). \tag{11b}$$

For convenience we let $\omega := (x, y, z, \tau, s, \kappa) \in \mathbb{R}^{n+p+2q+2}$ represent a point. We define the structured 4×6 block matrix $E \in \mathbb{R}^{(n+p+q+1)\times \dim(\omega)}$ such that (11a) is equivalent to $E\omega = 0$. Here we assume E has full row rank; in Appendix A we discuss preprocessing techniques that handle linearly dependent rows. Note that $\omega = 0$ satisfies (11), so the HSDE is always feasible. A point ω is called an interior point if it is strictly feasible for (11b), i.e. $(z, \tau, s, \kappa) \in \operatorname{int}\left(\mathcal{K}^* \times \mathbb{R}_{\geq} \times \mathcal{K} \times \mathbb{R}_{\geq}\right)$.

Suppose a point ω is feasible for the HSDE (11). From skew symmetry of the square 4×4 block matrix in (11a), we have $s'z + \kappa\tau = 0$. From (11b) and the dual cone inequality (1), we have $s'z \geq 0$ and $\kappa\tau \geq 0$. Hence $s'z = \kappa\tau = 0$. We consider an exhaustive list of cases below [66, Section 6.1].

Optimality. If $\tau > 0, \kappa = 0$, then $(x, y, z)/\tau$ is a complementary solution.
Infeasibility. If $\tau = 0, \kappa > 0$, then $c'x + b'y + h'z < 0$.

Of dual. If $c'x < 0$, then x is a primal improving ray.
Of primal. If $b'y + h'z < 0$, then (y, z) is a dual improving ray.

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

According to Skajaa and Ye [64, Section 2], if the primal and dual problems are feasible and have zero duality gap, SY (their algorithm) finds an HSDE solution with $\tau > 0$ (yielding a complementary solution), and if the primal or dual (possibly both) is infeasible, SY finds an HSDE solution with $\kappa > 0$ (yielding an infeasibility certificate). This implies that if SY finds a solution with $\kappa = \tau = 0$, then $\kappa = \tau = 0$ for all solutions to the HSDE; in this case, no complementary solution or improving ray exists, and the primal or dual (possibly both) is ill-posed [60]. The algorithm we describe in Sect. 5 is an extension of SY that inherits these properties.

5 Central path following algorithm

In Sect. 5.1, we describe the central path of the HSDE, and in Sect. 5.2 we define central path proximity measures. In Sect. 5.3, 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 Sect. 5.4, we derive prediction and centering directions and our corresponding TOA directions using the TOO. Finally in Sect. 5.5, we summarize an SY-like stepping procedure and describe our sequence of four enhancements to this procedure.

5.1 Central path of the HSDE

We define the HSDE in (11). Recall that $\mathcal{K}$ in our primal conic form (6) is a Cartesian product $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K$ of K exotic cones. We partition the exotic cone indices $[\![K]\!]$ into two sets: K_{pr} for cones with primal oracles (i.e. for $\mathcal{K}_k$) and K_{du} for cones with dual oracles (i.e. for $\mathcal{K}_k^*$). For convenience, we append the τ and κ variables onto the s and z variables. Letting $\bar{K} = K + 1$, we define for $k \in [\![\bar{K}]\!]$:

$$\bar{\mathcal{K}}_k \coloneqq \begin{cases} \mathcal{K}_k & k \in K_{\text{pr}}, \\ \mathcal{K}_k^* & k \in K_{\text{du}}, \\ \mathbb{R}_{\geq} & k = \bar{K}, \end{cases} \tag{12a}$$

$$(\bar{z}_k, \bar{s}_k) \coloneqq \begin{cases} (z_k, s_k) & k \in K_{\text{pr}}, \\ (s_k, z_k) & k \in K_{\text{du}}, \\ (\kappa, \tau) & k = \bar{K}. \end{cases} \tag{12b}$$

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

by $\mu > 0$, satisfying:

$$E\omega_\mu = \mu E\omega^0, \tag{13a}$$
$$\bar{z}_{\mu,k} + \mu g_k(\bar{s}_{\mu,k}) = 0 \quad \forall k \in [\![\bar{K}]\!], \tag{13b}$$
$$(\bar{z}_\mu, \bar{s}_\mu) \in \operatorname{int}(\bar{\mathcal{K}}^* \times \bar{\mathcal{K}}). \tag{13c}$$

When all exotic cones have primal oracles (i.e. K_{du} is empty), our definition (13) exactly matches the central path defined in [66, Equation 32], and only differs from the definition in [64, Equations 7-8] in the affine form (i.e. the variable names and affine constraint structure). Unlike SY, our central path condition (13b) allows cones with dual oracles (K_{du} may be nonempty).

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

$$\left(\bar{z}_k^0, \bar{s}_k^0\right) = (-g_k(t_k), t_k) \quad \forall k \in [\![\bar{K}]\!], \tag{14}$$

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

$$\min_{x \in \mathbb{R}^n} \quad \|x\| : \tag{15a}$$
$$-Ax + b\tau^0 = 0, \tag{15b}$$
$$-Gx + h\tau^0 - s^0 = 0, \tag{15c}$$

and we let y^0 be the solution of:

$$\min_{y \in \mathbb{R}^p} \quad \|y\| : \tag{16a}$$
$$A'y + G'z^0 + c\tau^0 = 0. \tag{16b}$$

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

Like [64, Section 4.1], we define the *complementarity gap* function:

$$\mu(\omega) := \bar{s}'\bar{z} / \textstyle\sum_{k \in [\![\bar{K}]\!]} \nu_k, \tag{17}$$

where ν_k is the LHSCB parameter of the LHSCB f_k for $\bar{\mathcal{K}}_k$ (see (2b)). Note that $\mu(\omega) > 0$ if $(\bar{z}, \bar{s}) \in \operatorname{int}(\bar{\mathcal{K}}^*) \times \operatorname{int}(\bar{\mathcal{K}})$, by a strict version of the dual cone inequality (1). From (14), $\mu(\omega^0) = 1$, since in (17) we have $(\bar{s}^0)'\bar{z}^0 = \sum_{k \in [\![\bar{K}]\!]} t_k'(-g_k(t_k))$, and $t_k'(-g_k(t_k)) = \nu_k$ by logarithmic homogeneity of f_k [49, Proposition 2.3.4]. Hence ω^0 satisfies the central path conditions (13) 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.

5.2 Central path proximity

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

$$\pi_k(\omega) := \begin{cases} \left\| (H_k(\bar{s}_k))^{-1/2}(\bar{z}_k/\mu(\omega) + g_k(\bar{s}_k)) \right\| & \text{if } \mu(\omega) > 0, \bar{s}_k \in \operatorname{int}\left(\bar{\mathcal{K}}_k\right), \\ \infty & \text{otherwise.} \end{cases} \tag{18}$$

Hence π_k is a measure of the distance from $\bar{s}_k$ and $\bar{z}_k$ to the surface defined by the central path condition (13b) (compare to [64, Equation 9] and [52, Section 4]).

In Lemma 1, we show that for exotic cone $k \in [\![\bar{K}]\!]$, if $\pi_k(\omega) < 1$, then $\bar{s}_k \in \operatorname{int}\left(\bar{\mathcal{K}}_k\right)$ and $\bar{z}_k \in \operatorname{int}\left(\bar{\mathcal{K}}_k^*\right)$. This condition is sufficient but not necessary for strict cone feasibility. If it holds for all $k \in [\![\bar{K}]\!]$, then ω is an interior point (by definition) and (13c) is satisfied. From (18), $\pi_k(\omega)$ can be computed by evaluating the feasibility check, gradient, and Hessian oracles for $\bar{\mathcal{K}}_k$ at $\bar{s}_k$.

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

Proof We adapt the proof of [56, Lemma 15]. Fix $\mu = \mu(\omega)$ for convenience, and suppose $\pi_k(\omega) < 1$ for exotic cone $k \in [\![\bar{K}]\!]$. Then by (18), $\mu > 0$ and $\bar{s}_k \in \operatorname{int}\left(\bar{\mathcal{K}}_k\right)$. By [56, Theorem 8], $\bar{s}_k \in \operatorname{int}\left(\bar{\mathcal{K}}_k\right)$ implies $-g_k(\bar{s}_k) \in \operatorname{int}\left(\bar{\mathcal{K}}_k^*\right)$. Let f_k be the LHSCB for $\bar{\mathcal{K}}_k$, and let $H_k^* := \nabla^2 f_k^*$ denote the Hessian operator for the conjugate f_k^* (see (3)) of f_k. By [56, Equation 13], $H_k^*(-g_k(\bar{s}_k)) = (H_k(\bar{s}_k))^{-1}$, so:

$$\left\| (H_k^*(-g_k(\bar{s}_k)))^{1/2}(\bar{z}_k/\mu + g_k(\bar{s}_k)) \right\| \tag{19a}$$

$$= \left\| (H_k(\bar{s}_k))^{-1/2}(\bar{z}_k/\mu + g_k(\bar{s}_k)) \right\| \tag{19b}$$

$$= \pi_k(\omega) < 1. \tag{19c}$$

So by [56, Definition 1], $\bar{z}_k/\mu \in \operatorname{int}\left(\bar{\mathcal{K}}_k^*\right)$, hence $\bar{z}_k \in \operatorname{int}\left(\bar{\mathcal{K}}_k^*\right)$. □

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

$$\pi_{\ell_2}(\omega) := \left\| (\pi_k(\omega))_{k \in [\![\bar{K}]\!]} \right\|. \tag{20}$$

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

$$\pi_{\ell_\infty}(\omega) := \left\| (\pi_k(\omega))_{k \in [\![\bar{K}]\!]} \right\|_\infty. \tag{21}$$

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

5.3 High level algorithm

We describe a high level algorithm for approximately solving the HSDE. The method starts at the initial interior point ω^0 with complementarity gap $\mu(\omega^0) = 1$ and approximately tracks the central path trajectory (13) through a series of iterations. It maintains feasibility for the linear equality conditions (13a) and strict cone feasibility conditions (13c), but allows violation of the nonlinear equality conditions (13b). On the ith iteration, the current interior point is ω^{i-1} satisfying $\pi_k(\omega^{i-1}) < 1, \forall k \in [\![\bar{K}]\!]$, and the complementarity gap is $\mu(\omega^{i-1})$. The method searches for a new point ω^i that maintains the proximity condition $\pi_k(\omega^i) < 1, \forall k \in [\![\bar{K}]\!]$ (and hence is an interior point) and either has a smaller complementarity gap $\mu(\omega^i) < \mu(\omega^{i-1})$ or a smaller aggregate proximity value $\pi(\omega^i) < \pi(\omega^{i-1})$ (where π is π_{ℓ_2} or π_{ℓ_∞}), or both. As the complementarity gap decreases towards zero, the RHS of (13a) approaches the origin, so the iterates approach a solution of the HSDE (11).

To detect an approximate conic certificate and terminate the iterations, we check whether the current iterate ω satisfies any of the following numerical convergence criteria. These conditions use positive tolerance values for feasibility ε_f, infeasibility ε_i, absolute gap ε_a, relative gap ε_r, and ill-posedness ε_p. The criteria and default tolerance values are similar to those described by MOSEK in [43, Section 13.3.2] and CVXOPT in [44], and implemented in Alfonso [58]. In Sect. 7.2, we describe the tolerance values we use for computational testing in this paper.

Optimality. We terminate with a complementary solution $(x, y, z)/\tau$ approximately satisfying the primal-dual optimality conditions (6b), (6c), (7b), (7c) and (10) if:

$$\max\left(\frac{\|A'y + G'z + c\tau\|_\infty}{1 + \|c\|_\infty}, \frac{\|-Ax + b\tau\|_\infty}{1 + \|b\|_\infty}, \frac{\|-Gx + h\tau - s\|_\infty}{1 + \|h\|_\infty}\right) \leq \varepsilon_f \tau, \tag{22a}$$

and at least one of the following two conditions holds:

$$s'z \leq \varepsilon_a, \tag{22b}$$

$$\min(s'z/\tau, |c'x + b'y + h'z|) \leq \varepsilon_r \max(\tau, \min(|c'x|, |b'y + h'z|)). \tag{22c}$$

Note that (22b) and (22c) are absolute and relative optimality gap conditions respectively.

Primal infeasibility. We terminate with a dual improving ray (y, z) approximately satisfying (9) if:

$$b'y + h'z < 0, \qquad \|A'y + G'z\|_\infty \leq -\varepsilon_i(b'y + h'z). \tag{23}$$

Dual infeasibility. We terminate with a primal improving ray x approximately satisfying (8) if:

$$c'x < 0, \qquad \max(\|Ax\|_\infty, \|Gx + s\|_\infty) \leq -\varepsilon_i c'x. \tag{24}$$

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

$$\mu(\omega) \leq \varepsilon_p, \qquad \tau \leq \varepsilon_p \min(1, \kappa). \tag{25}$$

The high level path following algorithm below computes an approximate solution to the HSDE. In Sect. 5.5, we describe specific stepping procedures for Line 5.

1: **procedure** SOLVEHSDE
2: compute initial interior point ω^0
3: $i \leftarrow 1$
4: **while** ω^{i-1} does not satisfy any of the convergence conditions (22) to (25) **do**
5: $\omega^i \leftarrow \text{STEP}(\omega^{i-1})$
6: $i \leftarrow i + 1$
7: **end while**
8: **return** ω^i
9: **end procedure**

5.4 Search directions

At a given iteration of the path following method, let ω be the current interior point and fix $\mu = \mu(\omega)$ for convenience. The stepping procedures we describe in Sect. 5.5 first compute one or more search directions, which depend on ω. We derive the *centering* direction in Sect. 5.4.1 and the *prediction* direction in Sect. 5.4.2. The goal of centering is to step to a point with a smaller aggregate central path proximity than the current point, i.e. to step towards the central path. The goal of prediction is to step to a point with a smaller complementarity gap, i.e. to step closer to a solution of the HSDE. The centering and prediction directions match those used by SY. We associate with each of these directions a new *third order adjustment* (TOA) direction, which depends on the TOO and helps to correct the corresponding unadjusted direction (which must be computed before the TOA direction). Hence we derive four types of directions here.

Each direction is computed as the solution to a linear system with a structured square block matrix left hand side (LHS) and a particular right hand side (RHS) vector. The LHS, which depends only on ω and the problem data, is the same for all four directions at a given iteration. We let $r \coloneqq (r_E, r_1, \ldots, r_{\bar{K}}) \in \mathbb{R}^{\dim(\omega)}$ represent an RHS, where $r_E \in \mathbb{R}^{n+p+q+1}$ corresponds to the linear equalities (13a) and $r_k \in \mathbb{R}^{q_k}, \forall k \in [\![\bar{K}]\!]$ corresponds to the nonlinear equalities (13b). The direction $\delta \coloneqq (\delta_x, \delta_y, \delta_z, \delta_\tau, \delta_s, \delta_\kappa) \in \mathbb{R}^{\dim(\omega)}$ corresponding to r is the solution to:

$$E\delta = r_E, \tag{26a}$$

$$\delta_{\bar{z},k} + \mu H_k(\bar{s}_k)\delta_{\bar{s},k} = r_k \quad \forall k \in [\![\bar{K}]\!]. \tag{26b}$$

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 Appendix A, we describe a particular method for solving (26).

5.4.1 Centering

The centering direction δ^c is analogous to the definition of [64, Section 3.2]. It reduces the violation on the central path nonlinear equality condition (13b) (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 (13a), we ensure $E\delta^c = E\delta^{ct} = 0$ in (26a).

Dropping the index $k \in [\![\bar{K}]\!]$ for conciseness, recall that (13b) expresses $\bar{z}+\mu g(\bar{s}) = 0$. A first order approximation of this condition gives:

$$\bar{z} + \delta_{\bar{z}} + \mu(g(\bar{s}) + H(\bar{s})\delta_{\bar{s}}) = 0 \tag{27a}$$

$$\Rightarrow \quad \delta_{\bar{z}} + \mu H(\bar{s})\delta_{\bar{s}} = -\bar{z} - \mu g(\bar{s}), \tag{27b}$$

which matches the form of (26b). Hence we let the centering direction δ^c be the solution to:

$$E\delta = 0, \tag{28a}$$

$$\delta_{\bar{z},k} + \mu H_k(\bar{s}_k)\delta_{\bar{s},k} = -\bar{z}_k - \mu g_k(\bar{s}_k) \quad \forall k \in [\![\bar{K}]\!]. \tag{28b}$$

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

$$\bar{z} + \delta_{\bar{z}} + \mu\big(g(\bar{s}) + H(\bar{s})\delta_{\bar{s}} + \tfrac{1}{2}\nabla^3 f(\bar{s})[\delta_{\bar{s}}, \delta_{\bar{s}}]\big) = 0 \tag{29a}$$

$$\Rightarrow \quad \delta_{\bar{z}} + \mu H(\bar{s})\delta_{\bar{s}} = -\bar{z} - \mu g(\bar{s}) + \mu \mathrm{T}(\bar{s}, \delta_{\bar{s}}), \tag{29b}$$

where (29b) uses the definition of the TOO in (5). Note that the RHSs of (27b) and (29b) differ only by $\mu\mathrm{T}(\bar{s}, \delta_{\bar{s}})$, which depends on $\delta_{\bar{s}}$. To remove this dependency, we substitute the centering direction δ^c, which we assume is already computed, into the RHS of (29b). Hence we let the centering TOA direction δ^{ct}, which adjusts the centering direction, be the solution to:

$$E\delta = 0, \tag{30a}$$

$$\delta_{\bar{z},k} + \mu H_k(\bar{s}_k)\delta_{\bar{s},k} = \mu \mathrm{T}_k\big(\bar{s}_k, \delta^c_{\bar{s},k}\big) \quad \forall k \in [\![\bar{K}]\!]. \tag{30b}$$

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).

5.4.2 Prediction

The prediction direction δ^p reduces the complementarity gap and is analogous to the definition of [64, Section 3.1]. We derive δ^p and its corresponding TOA direction δ^{pt} by considering the central path conditions (13) as a dynamical system parametrized by $\mu > 0$, and differentiating the linear and nonlinear equalities (13a) and (13b).

Differentiating (13a) once gives:

$$E\dot{\omega}_\mu = E\omega^0. \tag{31}$$

Rescaling (31) by $-\mu$ and substituting (13a) gives:

$$E(-\mu\dot{\omega}_\mu) = -\mu E\omega^0 = -E\omega_\mu. \tag{32}$$

Dropping the index $k \in [\![\bar{K}]\!]$ for conciseness, we differentiate $\bar{z}_\mu + \mu g(\bar{s}_\mu) = 0$ from (13b) once to get:

$$\dot{\bar{z}}_\mu + g(\bar{s}_\mu) + \mu H(\bar{s}_\mu)\dot{\bar{s}}_\mu = 0. \tag{33}$$

Rescaling (33) by $-\mu$ and substituting $\bar{z}_\mu = -\mu g(\bar{s}_\mu)$ from (13b) gives:

$$-\mu\dot{\bar{z}}_\mu + \mu H(\bar{s}_\mu)(-\mu\dot{\bar{s}}_\mu) = -\bar{z}_\mu. \tag{34}$$

The direction $\dot{\omega}_\mu$ is tangent to the central path. Like SY, we interpret the prediction direction as $\delta^p = -\mu\dot{\omega}_\mu$, so (32) and (34) become:

$$E\delta^p = -E\omega_\mu, \tag{35a}$$

$$\delta^p_{\bar{z}} + \mu H(\bar{s}_\mu)\delta^p_{\bar{s}} = -\bar{z}_\mu, \tag{35b}$$

which matches the form (26). So we let δ^p be the solution to:

$$E\delta = -E\omega, \tag{36a}$$

$$\delta_{\bar{z},k} + \mu H_k(\bar{s}_k)\delta_{\bar{s},k} = -\bar{z}_k \quad \forall k \in [\![\bar{K}]\!]. \tag{36b}$$

Differentiating (13a) twice and rescaling by $\frac{1}{2}\mu^2$ gives:

$$E\left(\tfrac{1}{2}\mu^2\ddot{\omega}_\mu\right) = 0. \tag{37}$$

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

$$\ddot{\bar{z}}_\mu + 2H(\bar{s}_\mu)\dot{\bar{s}}_\mu + \mu\nabla^3 f(\bar{s}_\mu)[\dot{\bar{s}}_\mu, \dot{\bar{s}}_\mu] + \mu H(\bar{s}_\mu)\ddot{\bar{s}}_\mu = 0. \tag{38}$$

Rescaling (38) by $\frac{1}{2}\mu^2$ and substituting the TOO definition (5), we have:

$$\tfrac{1}{2}\mu^2\ddot{\bar{z}}_\mu + \mu H(\bar{s}_\mu)\left(\tfrac{1}{2}\mu^2\ddot{\bar{s}}_\mu\right) = \mu H(\bar{s}_\mu)(-\mu\dot{\bar{s}}_\mu) - \tfrac{1}{2}\mu\nabla^3 f(\bar{s}_\mu)[-\mu\dot{\bar{s}}_\mu, -\mu\dot{\bar{s}}_\mu] \tag{39a}$$

$$= \mu H(\bar{s}_\mu)(-\mu\dot{\bar{s}}_\mu) + \mu\mathrm{T}(\bar{s}_\mu, -\mu\dot{\bar{s}}_\mu). \tag{39b}$$

We interpret the prediction TOA direction, which adjusts the prediction direction, as $\delta^{pt} = \frac{1}{2}\mu^2\ddot{\omega}$. The RHS of (39b) depends on $\dot{\bar{s}}_\mu$, so we remove this dependency

by substituting the prediction direction $\delta^p = -\mu\dot{\omega}_\mu$, which we assume is already computed. Hence using (37) and (39b), we let δ^{pt} be the solution to:

$$E\delta = 0, \tag{40a}$$

$$\delta_{\bar{z},k} + \mu H_k(\bar{s}_k)\delta_{\bar{s},k} = \mu H_k(\bar{s}_k)\delta^p_{\bar{s},k} + \mu \mathrm{T}_k\big(\bar{s}_k, \delta^p_{\bar{s},k}\big) \quad \forall k \in [\![\bar{K}]\!]. \tag{40b}$$

We note that the RHS in (40b) differs from the 'higher order corrector' RHS of Dahl and Andersen [21, Equation 16], which has the form $\frac{1}{2}\nabla^3 f_k\big[\delta^p_{\bar{s},k}, (H_k(\bar{s}_k))^{-1}\delta^p_{\bar{z},k}\big]$. For example, our form does not satisfy all of the properties in [21, Lemmas 3 and 4].

5.5 Stepping procedures

A stepping procedure computes one or more directions from Sect. 5.4 and uses the directions to search for a new interior point. Recall from Line 5 of the high level PDIPM in Sect. 5.3 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 Sect. 5.5.1, we describe a baseline stepping procedure mirroring that of Alfonso, which is an implementation of the SY algorithm with worst-case polynomial time iteration complexity. This procedure, which we call *basic*, alternates between prediction and centering steps and does not use the TOA directions. In Sects. 5.5.2 to 5.5.5, we describe a sequence of four cumulative enhancements to the *basic* procedure. The goal is to improve iteration counts or per-iteration computational efficiency in practice, without regard for theoretical iteration complexity guarantees. Our computational testing in Sect. 7 assesses the value of these enhancements on a diverse set of benchmark instances.

5.5.1 Basic stepping procedure

First, we decide whether to perform a centering step or a prediction step. If the current iterate ω^{i-1} (at the ith iteration) is very close to the central path, i.e. if the sum proximity (20) does not exceed $\eta = 0.0332$, or if the most recent $N = 4$ steps have all been centering steps, then we compute the prediction direction δ^p (note these parameter values are taken directly from Alfonso and are based on the theoretical analysis of [56]). Otherwise, we compute the centering direction δ^c from (28). Letting j be the number of consecutive centering steps taken immediately before the current ith iteration, the search direction is:

$$\delta := \begin{cases} \delta^p & \text{if } \pi_{\ell_2}(\omega^{i-1}) \le \eta \text{ or } j \ge N, \\ \delta^c & \text{otherwise.} \end{cases} \tag{41}$$

Next, we perform a backtracking line search in the direction δ. The search finds a step length $\hat{\alpha} \in (0, 1)$ from a fixed schedule of decreasing values $\mathcal{A} = \{\alpha_l\}_{l \in [\![L]\!]}$, where $L = 18$, $\alpha_1 = 0.9999$, and $\alpha_L = 0.0005$. The next iterate $\omega^i = \omega^{i-1} + \hat{\alpha}\delta$ becomes the first point in the backtracking line search that satisfies $\pi_{\ell_2}(\omega^i) \le \beta_1$ for

$\beta_1 = 0.2844$, which guarantees interiority by Lemma 1 (note β_1 is again taken directly from Alfonso and is based on the theoretical analysis of [56]). 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. In Appendix B we discuss our implementation of the proximity check that we run for each candidate point in the backtracking search.

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

procedure BASICSTEP(ω^{i-1}, j)
  if $\pi_{\ell_2}(\omega^{i-1}) \leq \eta$ or $j \geq N$ **then** ▷ choose predict or center
    $\delta \leftarrow \delta^p$ from (36) ▷ compute prediction direction
    $j \leftarrow 0$
  else
    $\delta \leftarrow \delta^c$ from (28) ▷ compute centering direction
    $j \leftarrow j + 1$
  end if
  $\hat{\alpha} \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_2}(\omega^{i-1} + \alpha\delta) \leq \beta_1\}$ ▷ compute step length by backtracking search
  $\omega^i \leftarrow \omega^{i-1} + \hat{\alpha}\delta$ ▷ update current iterate
end procedure

5.5.2 Less restrictive proximity

The *basic* stepping procedure in Sect. 5.5.1 requires iterates to remain in close proximity to the central path and usually only takes prediction steps from iterates that are very close to the central path. Although conservative proximity conditions are used to prove polynomial iteration complexity in [56], they may be too restrictive from the perspective of practical performance. To allow prediction steps from a larger neighborhood of the central path, we use the π_{ℓ_∞} proximity measure from (21) instead of π_{ℓ_2} to compute the proximity of ω^{i-1}, though we do not change the proximity bound η. To allow the step length to be as large as possible, we use π_{ℓ_∞} instead of π_{ℓ_2} for the backtracking search proximity checks and we replace β_1 from Sect. 5.5.1 by a larger proximity bound $\beta_2 = 0.99$. By Lemma 1, $\beta_2 < 1$ guarantees interiority, and our offline sensitivity analysis on β_2 suggests that 0.99 is a reasonable choice.[4]

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

procedure PROXSTEP(ω^{i-1}, j)
  if $\pi_{\ell_\infty}(\omega^{i-1}) \leq \eta$ or $j \geq N$ **then** ▷ use less restrictive proximity measure π_{ℓ_∞}
    $\delta \leftarrow \delta^p$ from (36)
    $j \leftarrow 0$
  else

[4] These results are available from the Hypatia wiki at https://raw.githubusercontent.com/wiki/chriscoey/Hypatia.jl/files/betas.pdf, and are run on our benchmark set from Sect. 7.

6: $\delta \leftarrow \delta^c$ from (28)
7: $j \leftarrow j + 1$
8: **end if**
9: $\hat{\alpha} \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_\infty}(\omega^{i-1} + \alpha\delta) \leq \beta_2\}$ ▷ use π_{ℓ_∞} and larger proximity bound β_2
10: $\omega^i \leftarrow \omega^{i-1} + \hat{\alpha}\delta$
11: **end procedure**

5.5.3 Third order adjustments

We modify the *prox* stepping procedure in Sect. 5.5.2 to incorporate the new TOA directions associated with the prediction and centering directions. In symmetric conic IPMs, it is common to compute a step length in the unadjusted prediction direction, use this step length to compute an adjusted direction, and then compute a step length in this final direction (see e.g. [66, Section 5.1] for CVXOPT's approach using the Mehrotra correction). Our approach is similar.

After deciding whether to predict or center (using the same criteria as *prox*), we compute the unadjusted direction δ^u (i.e. δ^p or δ^c) and its associated TOA direction δ^t (i.e. δ^{pt} or δ^{ct}). We perform a backtracking line search in direction δ^u (like *prox*) and use this unadjusted step length $\hat{\alpha}^u \in (0, 1)$ to scale down the TOA direction, letting the adjusted direction be $\delta^u + \hat{\alpha}^u\delta^t$. We perform a second backtracking line search in this final direction to compute the final step length $\hat{\alpha}$, using the same techniques and proximity condition as the first line search. If we think of $\hat{\alpha}^u$ as an approximation of $\hat{\alpha}$, then essentially the final step applies an adjustment of $\hat{\alpha}^2\delta^t$ to $\hat{\alpha}\delta^u$. Our derivations of the adjustment directions in Sect. 5.4 (particularly the centering direction) suggest that this is a reasonable heuristic for adjustment.

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

1: **procedure** TOASTEP(ω^{i-1}, j)
2: **if** $\pi_{\ell_\infty}(\omega^{i-1}) \leq \eta$ or $j \geq N$ **then**
3: $\delta^u \leftarrow \delta^p$ from (36)
4: $\delta^t \leftarrow \delta^{pt}$ from (40) ▷ compute prediction TOA direction
5: $j \leftarrow 0$
6: **else**
7: $\delta^u \leftarrow \delta^c$ from (28)
8: $\delta^t \leftarrow \delta^{ct}$ from (30) ▷ compute centering TOA direction
9: $j \leftarrow j + 1$
10: **end if**
11: $\hat{\alpha}^u \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_\infty}(\omega^{i-1} + \alpha\delta^u) \leq \beta_2\}$ ▷ perform line search for unadjusted direction
12: $\delta \leftarrow \delta^u + \hat{\alpha}^u\delta^t$ ▷ compute final direction
13: $\hat{\alpha} \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_\infty}(\omega^{i-1} + \alpha\delta) \leq \beta_2\}$
14: $\omega^i \leftarrow \omega^{i-1} + \hat{\alpha}\delta$
15: **end procedure**

5.5.4 Curve search

The *TOA* stepping procedure in Sect. 5.5.3 performs two backtracking line searches, which can be quite expensive. We propose using a single backtracking search along a curve that is quadratic in the step parameter α and linear in the unadjusted and TOA directions. Recall from Line 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** $\pi_{\ell_\infty}(\omega^{i-1}) \leq \eta$ or $j \geq N$ **then**
3: $\delta^u \leftarrow \delta^p$ from (36)
4: $\delta^t \leftarrow \delta^{pt}$ from (40)
5: $j \leftarrow 0$
6: **else**
7: $\delta^u \leftarrow \delta^c$ from (28)
8: $\delta^t \leftarrow \delta^{ct}$ from (30)
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: $\hat{\alpha} \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_\infty}(\hat{\omega}(\alpha)) \leq \beta_2\}$
13: $\omega^i \leftarrow \hat{\omega}(\hat{\alpha})$
14: **end procedure**

5.5.5 Combined directions

Unlike [59, 64], most conic PDIPMs combine the prediction and centering phases (e.g. [21, 66]). We propose using a single search on a curve that is quadratic in the step parameter α and linear in all four directions δ^c, δ^{ct}, δ^p, δ^{pt} from Sect. 5.5.3. Intuitively, we can step further in a convex combination of the prediction and centering directions than we can in just the prediction direction. In practice, a step length of one is usually ideal for the centering phase, so we can imagine performing a backtracking search from the point obtained from a pure prediction step (with step length one) towards the point obtained from a pure centering step, terminating when we are close enough to the centering point to satisfy the proximity condition. This approach fundamentally differs from the previous procedures we have described because the search trajectory does not finish at the current iterate ω^{i-1}. If $\hat{\omega}^p(\alpha)$ and $\hat{\omega}^c(\alpha)$ are the prediction and centering curve search trajectories from Line 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 δ^c, δ^{ct}, δ^p, δ^{pt} from (28), (30), (36) and (40) ▷ 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: $\hat{\alpha} \leftarrow \max\{\alpha \in \mathcal{A} : \pi_{\ell_\infty}(\hat{\omega}(\alpha)) \leq \beta_2\}$
5: $\omega^i \leftarrow \hat{\omega}(\hat{\alpha})$
6: **end procedure**

6 Oracles for predefined exotic cones

Below we list 23 exotic cone types that we have predefined through Hypatia's generic cone interface (see Sect. 3). Each of these cones is represented in the benchmark set of conic instances that we introduce in Sect. 7.1. Recall that we write any exotic cone $\mathcal{K}$ in vectorized form, i.e. as a subset of $\mathbb{R}^q$, where $q = \dim(\mathcal{K}) \geq 1$ is the cone dimension. For cones typically defined using symmetric matrices, we use the standard svec vectorization (see Sect. 2) 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 each cone, we define the LHSCB Hypatia uses below, and we list the associated barrier parameter ν in Table 1. 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.

Nonnegative cone. $\mathcal{K}_{\geq} := \mathbb{R}^d_{\geq}$ is the (self-dual) nonnegative real vectors (note for $d > 1$, $\mathcal{K}_{\geq}$ is not a primitive cone). We use the LHSCB $f(w) = -\log(w)$ [51, Section 2.1].

PSD cone. $\mathcal{K}_{\succeq} := \left\{w \in \mathbb{R}^{\mathrm{sd}(d)} : \mathrm{mat}(w) \in \mathbb{S}^d_{\succeq}\right\}$ is the (self-dual) PSD matrices of side dimension d. We use the LHSCB $f(w) = -\operatorname{logdet}(\mathrm{mat}(w))$ [51, Section 2.2].

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

Sparse PSD cone. $\mathcal{K}_{\mathrm{sPSD}}$ is the PSD matrices of side dimension s with a fixed sparsity pattern $\mathcal{S}$ containing $d \geq s$ nonzeros (including all diagonal elements); see Appendix C.4. The dual cone $\mathcal{K}^*_{\mathrm{sPSD}}$ is the symmetric matrices with pattern $\mathcal{S}$ for which there exists a PSD completion, i.e. an assignment of the elements not in $\mathcal{S}$ such that the full matrix is PSD. For simplicity, the complexity estimates in Table

1 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 [4]. We use the LHSCB in Appendix C.4.

Linear matrix inequality cone. $\mathcal{K}_{\text{LMI}} := \left\{w \in \mathbb{R}^d : \sum_{i\in[\![d]\!]} w_i P_i \in \mathbb{S}^s_{\succeq}\right\}$ are the vectors for which the matrix pencil of d matrices $P_i \in \mathbb{S}^s, \forall i \in [\![d]\!]$ is PSD. We assume $P_1 \succ 0$ so that we can use the initial interior point e_1. We use the LHSCB in Appendix C.2.

Infinity norm cone. $\mathcal{K}_{\ell_\infty} := \{(u, w) \in \mathbb{R}_{\geq} \times \mathbb{R}^d : u \geq \|w\|_\infty\}$ is the epigraph of the ℓ_∞ norm on $\mathbb{R}^d$. The dual cone $\mathcal{K}^*_{\ell_\infty}$ is the epigraph of the ℓ_1 norm. We use the LHSCB $f(u, w) = (d-1)\log(u) - \sum_{i\in[\![d]\!]} \log(u^2 - w_i^2)$ [32, Section 7.5].

Euclidean norm cone. $\mathcal{K}_{\ell_2} := \{(u, w) \in \mathbb{R}_{\geq} \times \mathbb{R}^d : u \geq \|w\|\}$ is the (self-dual) epigraph of the ℓ_2 norm on $\mathbb{R}^d$ (AKA second-order cone). We use the LHSCB $f(u, w) = -\log(u^2 - \|w\|^2)$ [51, Section 2.3].

Euclidean norm square cone. $\mathcal{K}_{\text{sqr}} := \{(u, v, w) \in \mathbb{R}_{\geq} \times \mathbb{R}_{\geq} \times \mathbb{R}^d : 2uv \geq \|w\|^2\}$ is the (self-dual) epigraph of the perspective of the square of the ℓ_2 norm on $\mathbb{R}^d$ (AKA rotated second-order cone). We use the LHSCB $f(u, v, w) = -\log(2uv - \|w\|^2)$ [51, Section 2.3].

Spectral norm cone. $\mathcal{K}_{\ell_{\text{spec}}} := \{(u, w) \in \mathbb{R}_{\geq} \times \mathbb{R}^{rs} : u \geq \sigma_1(W)\}$, where $W := \text{mat}(w)$ and σ_1 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, $\mathcal{K}^*_{\ell_{\text{spec}}}$ is the epigraph of the matrix nuclear norm (i.e. the sum of singular values). We use the LHSCB $f(u, w) = (r-1)\log(u) - \text{logdet}(u^2 I(r) - WW')$ [49, Section 5.4.6].

Matrix square cone. $\mathcal{K}_{\text{matsqr}} := \left\{(u, v, w) \in \mathbb{R}^{\text{sd}(r)} \times \mathbb{R}_{\geq} \times \mathbb{R}^{rs} : U \in \mathbb{S}^r_{\succeq}, 2Uv \succeq WW'\right\}$, 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 [33]. We use the LHSCB $f(u, v, w) = (r-1)\log(v) - \text{logdet}(2vU - WW')$ [5].

Generalized power cone. $\mathcal{K}_{\text{gpow}} := \left\{(u, w) \in \mathbb{R}^r_{\geq} \times \mathbb{R}^s : \prod_{i\in[\![r]\!]} u_i^{\alpha_i} \geq \|w\|\right\}$, parametrized by exponents $\alpha \in \mathbb{R}^r_{>}$ with $e'\alpha = 1$, is the generalized power cone [16, Section 3.1.2]. We use the LHSCB $f(u, w) = -\log\left(\prod_{i\in[\![r]\!]} u_i^{2\alpha_i} - \|w\|^2\right) - \sum_{i\in[\![r]\!]}(1-\alpha_i)\log(u_i)$ [62].

Power mean cone. $\mathcal{K}_{\text{pow}} := \left\{(u, w) \in \mathbb{R} \times \mathbb{R}^d_{\geq} : u \leq \prod_{i\in[\![d]\!]} w_i^{\alpha_i}\right\}$, parametrized by exponents $\alpha \in \mathbb{R}^d_{>}$ with $e'\alpha = 1$, is the hypograph of the power mean on $\mathbb{R}^d_{\geq}$. We use the LHSCB $f(u, w) = -\log\left(\prod_{i\in[\![d]\!]} w_i^{\alpha_i} - u\right) - \sum_{i\in[\![d]\!]} \log(w_i)$ [48, Section 5.4.7].

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

Root-determinant cone. $\mathcal{K}_{\text{rtdet}} := \left\{(u, w) \in \mathbb{R} \times \mathbb{R}^{\text{sd}(d)} : W \in \mathbb{S}^d_{\succeq}, u \leq (\det(W))^{1/d}\right\}$, where $W := \text{mat}(w)$, is the hypograph of the dth-root-determinant on $\mathbb{S}^d_{\succeq}$. We use the LHSCB $f(u, w) = -\log((\det(W))^{1/d} - u) - \text{logdet}(W)$ [18, Proposition 7.1].

Logarithm cone. $\mathcal{K}_{\log} := \mathrm{cl}\{(u, v, w) \in \mathbb{R} \times \mathbb{R}_> \times \mathbb{R}^d_> : u \leq \sum_{i \in [\![d]\!]} v \log(w_i / v)\}$ is the hypograph of the perspective of the sum of logarithms on $\mathbb{R}^d_>$. We use the LHSCB $f(u, v, w) = -\log\left(\sum_{i \in [\![d]\!]} v \log(w_i/v) - u\right) - \log(v) - \sum_{i \in [\![d]\!]} \log(w_i)$ [18, Proposition 6.1].

Log-determinant cone. $\mathcal{K}_{\mathrm{logdet}} := \mathrm{cl}\{(u, v, w) \in \mathbb{R} \times \mathbb{R}_> \times \mathbb{R}^{\mathrm{sd}(d)} : W \in \mathbb{S}^d_>, u \leq v \,\mathrm{logdet}(W/v)\}$, where $W := \mathrm{mat}(w)$, is the hypograph of the perspective of the log-determinant on $\mathbb{S}^d_>$. We use the LHSCB $f(u, v, w) = -\log(v \,\mathrm{logdet}(W/v) - u) - \log(v) - \mathrm{logdet}(W)$ [18, Proposition 6.1].

Separable spectral function cone. $\mathcal{K}_{\mathrm{sepspec}} := \mathrm{cl}\{(u, v, w) \in \mathbb{R} \times \mathbb{R}_> \times \mathrm{int}(\mathcal{Q}) : u \geq v\varphi(w/v)\}$, where $\mathcal{Q}$ is $\mathcal{K}_{\geq}$ or $\mathcal{K}_{\geq}$ (a cone of squares of a Jordan algebra), is the epigraph of the perspective of a convex separable spectral function $\varphi : \mathrm{int}(\mathcal{Q}) \to \mathbb{R}$, such as the sum or trace of the negative logarithm, negative entropy, or power in $(1, 2]$ (see [18] for more details). The complexity estimates in Table 1 depend on whether $\mathcal{Q}$ is $\mathcal{K}_{\geq}$ or $\mathcal{K}_{\geq}$. We use the LHSCB $f(u, v, w) = -\log(u - v\varphi(w/v)) - \log(v) - \mathrm{logdet}(w)$ [18, Proposition 6.1].

Relative entropy cone. $\mathcal{K}_{\mathrm{relent}} := \mathrm{cl}\{(u, v, w) \in \mathbb{R} \times \mathbb{R}^d_> \times \mathbb{R}^d_> : u \geq \sum_{i \in [\![d]\!]} w_i \log(w_i/v_i)\}$ is the epigraph of vector relative entropy. We use the LHSCB $f(u, v, w) = -\log\left(u - \sum_{i \in [\![d]\!]} w_i \log(w_i/v_i)\right) - \sum_{i \in [\![d]\!]} (\log(v_i) + \log(w_i))$ [36, Appendix E].

Matrix relative entropy cone. $\mathcal{K}_{\mathrm{matrelent}} := \mathrm{cl}\{(u, v, w) \in \mathbb{R} \times \mathbb{R}^{\mathrm{sd}(d)} \times \mathbb{R}^{\mathrm{sd}(d)} : V \in \mathbb{S}^d_>, W \in \mathbb{S}^d_>, u \geq \mathrm{tr}(W(\log(W) - \log(V)))\}$, where $V := \mathrm{mat}(v)$ and $W := \mathrm{mat}(w)$, is the epigraph of matrix relative entropy. We use the LHSCB $f(u, v, w) = -\log(u - \mathrm{tr}(W(\log(W) - \log(V)))) - \mathrm{logdet}(V) - \mathrm{logdet}(W)$ [27, Theorem 1.5].

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 [57]. For simplicity, the complexity estimates in Table 1 assume $s_l = s, \forall l \in [\![r]\!]$. Note that $s < d \leq s^2$. We define $\mathcal{K}_{\mathrm{SOS}}$ and $\mathcal{K}_{\mathrm{matSOS}}$ in [20], and $\mathcal{K}_{\ell_1\mathrm{SOS}}$ and $\mathcal{K}_{\ell_2\mathrm{SOS}}$ in [35, Equations 2.7 and 4.9].

We use LHSCBs for the dual cones of these WSOS cones. For $\mathcal{K}^*_{\mathrm{SOS}}$ and $\mathcal{K}^*_{\mathrm{matSOS}}$, we discuss the LHSCBs in Appendix C.3. For $\mathcal{K}^*_{\ell_1\mathrm{SOS}}$ and $\mathcal{K}^*_{\ell_2\mathrm{SOS}}$, the LHSCBs require more complex notation, so we refer the reader to [35].

Scalar WSOS cone. $\mathcal{K}_{\mathrm{SOS}}$ is a cone of polynomials that are guaranteed to be nonnegative pointwise on the domain.

Symmetric matrix WSOS cone. $\mathcal{K}_{\mathrm{matSOS}}$ is a cone of polynomial symmetric matrices (in an svec-like format) of side dimension t that are guaranteed to belong to $\mathbb{S}_{\geq}$ pointwise on the domain. We let $m := st + d$ in Table 1 for succinctness.

ℓ_1 **epigraph WSOS cone.** $\mathcal{K}_{\ell_1\mathrm{SOS}}$ is a cone of polynomial vectors of length $1 + t$ that are guaranteed to belong to $\mathcal{K}^*_{\ell_\infty}$ pointwise on the domain.

ℓ_2 **epigraph WSOS cone.** $\mathcal{K}_{\ell_2\mathrm{SOS}}$ is a cone of polynomial vectors of length $1 + t$ that are guaranteed to belong to $\mathcal{K}_{\ell_2}$ pointwise on the domain.

Table 1 Cone dimension $\dim(\mathcal{K})$, LHSCB parameter ν, and time complexity estimates (ignoring constants) for our feasibility check, gradient, Hessian, and TOO implementations, for the exotic cones defined in Sect. 6

Cone	$\dim(\mathcal{K})$	ν	Feasibility	Gradient	Hessian	TOO
$\mathcal{K}_{\geq}$	d	d	d	d	d	d
$\mathcal{K}_{\geq}$	$\mathrm{sd}(d)$	d	d^3	d^3	d^4	d^3
$\mathcal{K}_{\mathrm{DNN}}$	$\mathrm{sd}(d)$	$\mathrm{sd}(d)$	d^3	d^3	d^4	d^3
$\mathcal{K}_{\mathrm{sPSD}}$	d	s	JD^2l	JD^2l	dJD^2l	JD^2l
$\mathcal{K}_{\mathrm{LMI}}$	d	s	$ds^2 + s^3$	ds^3	d^2s^2	$ds^2 + s^3$
$\mathcal{K}_{\ell_\infty}$	$1 + d$	$1 + d$	d	d	d	d
$\mathcal{K}_{\ell_2}, \mathcal{K}_{\mathrm{sqr}}$	$1 + d$	2	d	d	d^2	d
$\mathcal{K}_{\ell_{\mathrm{spec}}}$	$1 + rs$	$1 + r$	$r^2s + r^3$	$r^2s + r^3$	r^2s^2	rs^2
$\mathcal{K}_{\mathrm{matsqr}}$	$\mathrm{sd}(r) + 1 + rs$	$1 + r$	$r^2s + r^3$	$r^2s + r^3$	r^2s^2	rs^2
$\mathcal{K}_{\mathrm{gpow}}$	$r + s$	$1 + r$	$r + s$	$r + s$	$r^2 + s^2$	$r + s$
$\mathcal{K}_{\mathrm{pow}}, \mathcal{K}_{\mathrm{geo}}$	$1 + d$	$1 + d$	d	d	d^2	d
$\mathcal{K}_{\mathrm{rtdet}}$	$1 + \mathrm{sd}(d)$	$1 + d$	d^3	d^3	d^4	d^3
$\mathcal{K}_{\log}$	$2 + d$	$2 + d$	d	d	d^2	d
$\mathcal{K}_{\mathrm{logdet}}$	$2 + \mathrm{sd}(d)$	$2 + d$	d^3	d^3	d^4	d^3
$\mathcal{K}_{\mathrm{sepspec}}$-$\mathcal{K}_{\geq}$	$2 + d$	$2 + d$	d	d	d^2	d
$\mathcal{K}_{\mathrm{sepspec}}$-$\mathcal{K}_{\geq}$	$2 + \mathrm{sd}(d)$	$2 + d$	d^3	d^3	d^5	d^3
$\mathcal{K}_{\mathrm{relent}}$	$1 + 2d$	$1 + 2d$	d	d	d^2	d
$\mathcal{K}_{\mathrm{matrelent}}$	$1 + 2\,\mathrm{sd}(d)$	$1 + 2d$	d^3	d^3	d^5	d^4
$\mathcal{K}^*_{\mathrm{SOS}}$	d	sr	ds^2r	ds^2r	d^2sr	ds^2r
$\mathcal{K}^*_{\mathrm{matSOS}}$	$d\,\mathrm{sd}(t)$	str	ms^2t^2r	ds^2t^2r	d^2st^3r	ms^2t^2r
$\mathcal{K}^*_{\ell_1\mathrm{SOS}}$	$d(1 + t)$	str	ds^2tr	ds^2tr	d^2str	ds^2tr
$\mathcal{K}^*_{\ell_2\mathrm{SOS}}$	$d(1 + t)$	$2sr$	ds^2tr	ds^2tr	d^2st^2r	ds^2t^2r

For each cone, we have an analytic form for the feasibility check, gradient, Hessian, and TOO oracles defined in Sect. 3. That is, we always avoid iterative numerical procedures such as optimization, which are typically slow, numerically unstable, and require tuning. Hypatia's algorithm always evaluates the feasibility check before the gradient, Hessian, and TOO (which are only defined at strictly feasible points), and the gradient is evaluated before the Hessian and TOO. For most of these cones, the feasibility check and gradient oracles compute values and factorizations that are also useful for computing the Hessian and TOO, so this data is cached in the cone data structures and reused where possible. In Table 1, we estimate the time complexities (ignoring constants) of these four oracles for each cone, counting the cost of cached values and factorizations only once (for the oracle that actually computes them). Table 1 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 Sect. 7.3 demonstrate that the TOO is very rarely an algorithmic bottleneck in practice.

Our TOO in (5) is distinct from the 'higher order corrector' terms proposed by Mehrotra [41] or Dahl and Andersen [21]. The method of [41] only applies to symmetric cones, and the technique in [21] is tested only for the standard exponential cone. Compared to the third order term proposed by Dahl and Andersen [21], our TOO has a simpler and more symmetric structure, as it relies on only one direction $\delta_{\bar{s}}$ rather than two. Like the gradient and Hessian oracles, our TOO is additive for sums of LHSCBs, which can be useful for cones (such as $\mathcal{K}_{\text{DNN}}$ and $\mathcal{K}^*_{\text{SOS}}$) that are defined as intersections of other cones. We leverage these properties to obtain fast and numerically stable TOO implementations.

To illustrate, in Appendix C.1 we define LHSCBs and derive efficient TOO procedures for a class of cones that can be characterized as intersections of slices of the PSD cone $\mathcal{K}_{\succeq}$. We consider $\mathcal{K}_{\text{LMI}}$ in Appendix C.2 and $\mathcal{K}^*_{\text{SOS}}$ and $\mathcal{K}^*_{\text{matSOS}}$ in Appendix C.3. In Appendix C.4, we handle $\mathcal{K}_{\text{sPSD}}$ by differentiating a procedure from [4] for computing Hessian products. In Appendix C.5 we also show how to compute the TOO for $\mathcal{K}_{\ell_2}$ and $\mathcal{K}_{\text{sqr}}$. In [18], we derive efficient TOO procedures for a class of spectral function cones on positive domains ($\mathcal{K}_{\text{sepspec}}$, $\mathcal{K}_{\log}$, $\mathcal{K}_{\text{logdet}}$, $\mathcal{K}_{\text{geo}}$, $\mathcal{K}_{\text{rtdet}}$).

7 Computational testing

In Sect. 7.1, we introduce a diverse set of exotic conic benchmark instances generated from a variety of applied examples. In Sect. 7.2, we describe our methodology for comparing the stepping procedures from Sect. 5.5, and in Sect. 7.3 we examine our computational results.

7.1 Exotic conic benchmark set

We generate 379 instances (in our primal general form (6)) 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 Sect. 6. Each cone from this list appears in at least one instance, so we consider our benchmark set to be the most diverse collection of conic instances available.

We generate most instances using JuMP, but for some we use Hypatia's native model interface. Due to the size of some instances and the lack of a standard instance storage format recognizing our cone types, we generate all instances on the fly in Julia. For instances that use random data, we set random seeds to ensure reproducibility. Figure 1 shows the distributions of instance dimensions and exotic cone counts. All instances have at least one cone (note any $\mathcal{K}_{\geq}$ cones are concatenated together, so $\mathcal{K}_{\geq}$ is counted at most once) and take at least one iteration to solve with Hypatia.

Below we briefly introduce each example. In Table 2, we summarize for each example the number of corresponding instances and the cone types represented in at least one of the instances. We do not distinguish dual cones and primal cones in this summary (for example, instances that use $\mathcal{K}^*_{\ell_\infty}$ are only listed as using $\mathcal{K}_{\ell_\infty}$). For some examples, we describe a subset of formulations in [18, 20, 35]. Our benchmark set

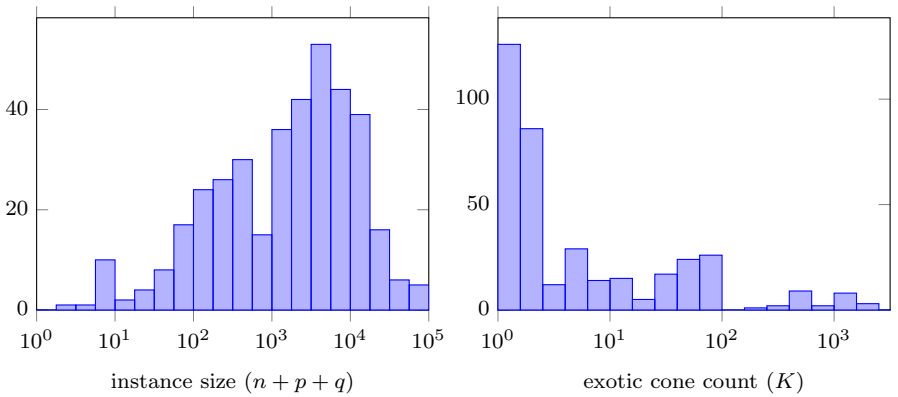

Fig. 1 Histograms summarizing the benchmark instances in the primal conic form (6). Instance size (log scale) is the sum of the primal variable, equality, and conic constraint dimensions. Exotic cone count (log scale) is the number of exotic cones comprising the Cartesian product cone

Table 2 For each example, the count of instances and list of exotic cones (defined in Sect. 6) used in at least one instance

Example	#	Cones in at least one instance
CBLIB	10	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{gpow}}$
Central polynomial matrix	24	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{gpow}}$ $\mathcal{K}_{\text{rtdet}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sepspec}}$
Classical-quantum capacity	9	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sepspec}}$
Condition number	6	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{LMI}}$
Contraction analysis	8	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{matSOS}}$
Convexity parameter	7	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{matSOS}}$
Covariance estimation	13	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{gpow}}$ $\mathcal{K}_{\text{rtdet}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sepspec}}$
Density estimation	16	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{geo}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{SOS}}$
Discrete maximum likelihood	7	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{pow}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sepspec}}$
D-optimal design	16	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{geo}}$ $\mathcal{K}_{\text{rtdet}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{logdet}}$
Entanglement-assisted capacity	3	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{sepspec}}$ $\mathcal{K}_{\text{matrelent}}$
Experiment design	13	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{gpow}}$ $\mathcal{K}_{\text{rtdet}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sepspec}}$
Linear program	3	$\mathcal{K}_{\geq}$
Lotka-Volterra	3	$\mathcal{K}_{\geq}$
Lyapunov stability	10	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{matsqr}}$
Matrix completion	11	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\ell \text{ spec}}$ $\mathcal{K}_{\text{gpow}}$ $\mathcal{K}_{\text{geo}}$ $\mathcal{K}_{\log}$
Matrix quadratic	8	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{matsqr}}$
Matrix regression	11	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\ell_{\text{spec}}}$
Maximum volume hypercube	15	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{geo}}$
Nearest correlation matrix	3	$\mathcal{K}_{\text{matrelent}}$
Nearest polynomial matrix	8	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{SOS}}$ $\mathcal{K}_{\text{matSOS}}$
Nearest PSD matrix	28	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{sPSD}}$
Nonparametric distribution	10	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{sqr}}$ $\mathcal{K}_{\text{geo}}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{sepspec}}$

Table 2 continued

Example	#	Cones in at least one instance
Norm cone polynomial	10	$\mathcal{K}_{\ell_1\text{SOS}}$ $\mathcal{K}_{\ell_2\text{SOS}}$
Polynomial envelope	7	$\mathcal{K}_{\text{SOS}}$
Polynomial minimization	15	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{SOS}}$
Polynomial norm	10	$\mathcal{K}_{\text{SOS}}$ $\mathcal{K}_{\text{matSOS}}$ $\mathcal{K}_{\ell_1\text{SOS}}$ $\mathcal{K}_{\ell_2\text{SOS}}$
Portfolio	9	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\ell_2}$
Region of attraction	6	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{SOS}}$
Relative entropy of entanglement	6	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{matrelent}}$
Robust geometric programming	6	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{relent}}$
Semidefinite polynomial matrix	18	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{matSOS}}$
Shape constrained regression	11	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_\infty}$ $\mathcal{K}_{\ell_2}$ $\mathcal{K}_{\text{SOS}}$ $\mathcal{K}_{\text{matSOS}}$
Signomial minimization	13	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\log}$ $\mathcal{K}_{\text{relent}}$
Sparse LMI	15	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{sPSD}}$ $\mathcal{K}_{\text{LMI}}$
Sparse principal components	6	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\ell_\infty}$
Stability number	6	$\mathcal{K}_{\geq}$ $\mathcal{K}_{\geq}$ $\mathcal{K}_{\text{DNN}}$

includes ten instances from CBLIB (a conic benchmark instance library, see [30]). 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 [26, Section 3.1]).
Condition number. Minimize the condition number of a matrix pencil subject to a linear matrix inequality (adapted from [10, Section 3.2]).
Contraction analysis. Find a contraction metric that guarantees global stability of a dynamical system (adapted from [6, 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 [55, Section 4.3]; see [20, 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 [11, Section 7.5]; see [20, Section 5.4]).

Entanglement-assisted capacity. Compute the entanglement-assisted classical capacity of a quantum channel (adapted from [26, 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 [11, 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 [38, Section 7.2]).
Lyapunov stability. Minimize an upper bound on the root mean square gain of a dynamical system (adapted from [10, Section 6.3.2] and [9, Page 6]).
Matrix completion. Complete a rectangular matrix by minimizing the nuclear norm and constraining the missing entries (compare to [1, Equation 8]; see [20, 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 [20, 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 [42, Section 4.3.2]).
Nearest correlation matrix. Compute the nearest correlation matrix in the quantum relative entropy sense (adapted from [28]).
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 [65]).
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 $\mathcal{K}^*_{\ell_\infty}$. Four instances are primal infeasible.
Polynomial envelope. Find a polynomial that closely approximates, over the unit box, the lower envelope of a given list of polynomials (see [57, Section 7.2.1]).
Polynomial minimization. Compute a lower bound for a given polynomial over a given semialgebraic set (see [57, Section 7.3.1] and [20, Section 5.5]). Some instances use polynomials with known optimal values from [13].
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 [35]).
Portfolio. Maximize the expected returns of a stock portfolio and satisfy various risk constraints (see [20, Section 5.1]).
Region of attraction. Find the region of attraction of a polynomial control system (see [34, Section 9.1]).

Relative entropy of entanglement. Compute a lower bound on relative entropy of entanglement with a positive partial transpose relaxation (adapted from [26, Section 4]).
Robust geometric programming. Bound the worst-case optimal value of an uncertain signomial function with a given coefficient uncertainty set (adapted from [15, Equation 39]).
Semidefinite polynomial matrix. Check a sufficient condition for global convexity of a given polynomial. Two instances are primal infeasible and one is dual infeasible.
Shape constrained regression. Given a dataset, fit a polynomial function that satisfies shape constraints such as monotonicity or convexity over a domain (see [20, Section 5.7]). Several instances use real datasets from [40].
Signomial minimization. Compute a global lower bound for a given signomial function (see [45]). Several instances use signomials with known optimal values from [14, 45].
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 [22, Section 2]).
Stability number. Given a graph, solve for a particular strengthening of the theta function towards the stability number (adapted from [39, Equation 2.4]).

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 Sect. 5.5.

basic. The basic prediction or centering stepping procedure without any enhancements; described in Sect. 5.5.1, this is similar to the method in Alfonso solver [59], which is a practical implementation of the algorithm in [56, 64].
prox. The *basic* procedure modified to use a less restrictive central path proximity condition; described in Sect. 5.5.2.
TOA. The *prox* procedure with the TOA enhancement to incorporate third order LHSCB information; described in Sect. 5.5.3.
curve. The *TOA* procedure adapted for a single backtracking search on a curve instead of two backtracking line searches; described in Sect. 5.5.4.
comb. The *curve* procedure modified to search along a curve of combinations of both the prediction and centering directions and their corresponding adjustment directions; described in Sect. 5.5.5.

We perform all instance generation, computational experiments, and results analysis using double precision floating point format, with Ubuntu 21.04, Julia 1.7, and Hypatia 0.5.2-patch (with default options), on dedicated hardware with an AMD Ryzen 9 3950X 16-core processor (32 threads) and 128GB of RAM. In Appendix A, we outline

the default procedures Hypatia uses for preprocessing, initial point finding, and linear system solving for search directions. Simple scripts and instructions for reproducing all results are available in Hypatia's benchmarks/stepper folder. The benchmark script runs all solves twice and uses results from the second run, to exclude Julia compilation overhead. A CSV file containing raw results is available at the Hypatia wiki page.

When Hypatia converges for an instance, i.e. claims it has found a certificate of optimality, primal infeasibility, or dual infeasibility, our scripts verify that this is the correct type of certificate for that instance. For some instances, our scripts also check additional conditions, for example that the objective value of an optimality certificate approximately equals the known true optimal value. We do not set restrictive time or iteration limits. All failures to converge are caused by Hypatia 'stalling' during the stepping iterations: either the backtracking search cannot step a distance of at least the minimal value in the α schedule, or across several prediction steps or combined directions steps, Hypatia fails to make sufficient progress towards meeting the convergence conditions in Sect. 5.3.

Since some instances are more numerically challenging than others, we set the termination tolerances (described in Sect. 5.3) separately for each instance. Let $\epsilon \approx 2.22 \times 10^{-16}$ be the machine epsilon. For most instances, we use $\varepsilon_f = \varepsilon_r = 10\epsilon^{1/2} \approx 1.49 \times 10^{-7}$ for the feasibility and relative gap tolerances, $\varepsilon_i = \varepsilon_a = 10\epsilon^{3/4} \approx 1.82 \times 10^{-11}$ for the infeasibility and absolute gap tolerances, and $\varepsilon_p = 0.1\epsilon^{3/4} \approx 1.82 \times 10^{-13}$ for the ill-posedness tolerance. For 50 instances that are particularly numerically challenging, we loosen all of these tolerances by a factor of either 10 or 100, and for two challenging primal infeasible instances of the *contraction analysis* example, we set $\varepsilon_i = 10^{-9}$. This ensures that for every benchmark instance, at least one of the five stepping procedures converges.

Following [29], we define the *shifted geometric mean* with shift $s \geq 0$, for d values $v \in \mathbb{R}^d_>$, as:

$$M(v, s) := \prod_{i \in [\![d]\!]} (v_i + s)^{1/d} - s. \tag{42}$$

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 (42)) 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 Sect. 7.3.

Table 3 For each stepping procedure, the number of converged instances and shifted geometric means of iterations and solve times (in milliseconds)

Step	Conv	Iterations			Solve time		
		every	this	all	every	this	all
basic	371	101.34	100.93	102.39	2130.98	2207.10	2282.19
prox	369	64.73	65.28	67.23	1316.50	1390.14	1451.24
TOA	374	34.98	35.31	36.08	1014.26	1062.66	1103.01
curve	372	29.67	30.01	30.99	742.49	780.95	820.16
comb	367	18.31	18.55	20.02	623.82	655.71	706.49

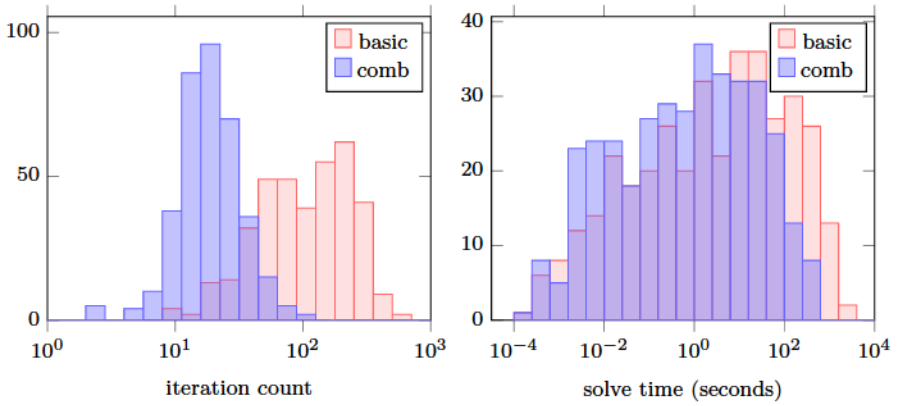

Fig. 2 Overlayed histograms of iteration count (left, log scale) and solve time (right, log scale, in seconds) for the *basic* and *comb* stepping procedures, excluding instances that fail to converge

Table 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 3, as some instances solve very quickly (see Fig. 2).

Table 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 Appendix A).

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

RHS. Performed between one and four times per iteration, depending on the stepping procedure. Includes updating an RHS vector (see (26)) for the linear system for search directions. Note that the TOO is only evaluated while computing the centering TOA RHS (30b) and the prediction TOA RHS (40b).

Table 4 For each stepping procedure, the shifted geometric means of subtimings (in milliseconds) for the key algorithmic components

Set	Step	init	Total				Per iteration			
			LHS	RHS	direc	search	LHS	RHS	direc	search
every	basic	29.49	741.38	1.45	75.59	125.64	7.73	0.02	0.81	1.29
	prox	29.45	486.34	1.11	50.33	67.74	7.88	0.02	0.84	1.10
	TOA	29.42	284.68	10.96	52.21	73.33	8.28	0.32	1.53	2.14
	curve	29.55	244.38	9.24	44.60	33.43	8.33	0.32	1.53	1.15
	comb	29.31	159.94	10.51	57.57	35.23	8.74	0.58	3.14	1.94
this	basic	30.33	784.33	1.48	78.82	131.78	8.20	0.02	0.85	1.36
	prox	30.07	519.48	1.12	53.43	72.60	8.35	0.02	0.88	1.16
	TOA	30.16	301.54	11.97	55.44	78.34	8.70	0.35	1.61	2.26
	curve	30.52	260.67	9.99	47.30	35.10	8.80	0.34	1.60	1.20
	comb	30.46	171.04	11.03	60.72	36.40	9.23	0.60	3.27	1.98
all	basic	31.13	814.44	1.62	82.65	134.65	8.52	0.02	0.91	1.40
	prox	31.29	549.29	1.25	56.22	75.09	8.74	0.02	0.94	1.20
	TOA	31.16	316.91	12.23	57.52	79.71	9.04	0.36	1.66	2.28
	curve	31.38	275.95	10.40	49.63	37.34	9.17	0.36	1.68	1.26
	comb	31.36	188.07	11.88	64.18	40.02	9.66	0.63	3.35	2.10

direc. Performed for each RHS vector. Includes solving the linear system for a search direction (see (26)) using the data computed during *LHS* and a single RHS vector computed during *RHS*, and performing iterative refinement on the direction (see Appendix A).

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

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 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 Figs. 4 and 7 we use *performance profiles* [23, 31] to compare iteration counts and solve times between pairs of stepping procedures. These should be interpreted as follows. The *performance ratio* for procedure i and instance j is the value (iterations or solve time) attained by procedure i on instance j divided by the better/smaller value attained by the two procedures on instance j. Hence a performance ratio is at least one, and smaller values indicate better relative performance. For a point (x, y) on a performance profile curve for a particular procedure, x is the logarithm (base 2) of performance ratio and y is the proportion of instances for which

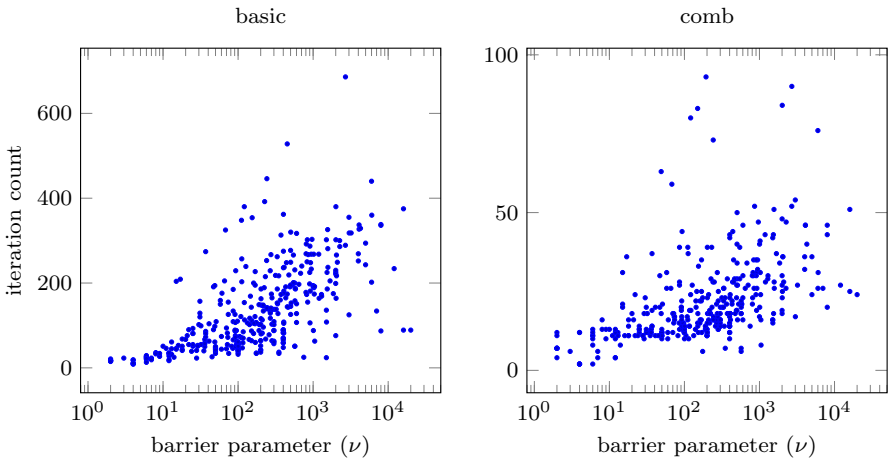

Fig. 3 Iteration count against instance barrier parameter for the *basic* (left) and *comb* (right) stepping procedures, excluding instances that fail to converge

the procedure attains that performance ratio or better/smaller. For example, a curve crosses the vertical axis at the proportion of instances on which the corresponding procedure performed at least as well as the alternative procedure. We use the Julia package BenchmarkProfiles.jl [53] to compute coordinates for the performance profile curves.

7.3 Results

Table 3 and Fig. 7 demonstrate that each of the four cumulative stepping enhancements tends to improve Hypatia's iteration count and solve time. The enhancements do not have a significant impact on the number of instances Hypatia converges on. However, if we had enforced time or iteration limits, the enhancements would have also improved the number of instances solved. This is clear from Fig. 2, which shows the distributions of iteration counts and solve times for the *basic* and *comb* stepping procedures.

We note that Fig. 5 (left) supports the intuition that formulation size is strongly positively correlated with solve time for *comb*. Furthermore, Fig. 3 shows a positive correlation between iteration count and instance barrier parameter ν, for both the *basic* and *comb* steppers. This is expected for the *basic* stepper, as the theoretical worst-case iteration complexity for the SY algorithm is proportional to $\sqrt{\nu}$ [56, 64]. However we note that on our benchmark set, ν is also correlated with the instance size (particularly the cone dimension q) and exotic cone count K, which may also affect iteration counts in practice.

Overall, Table 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 4 shows that the iteration count and solve time improve on nearly every instance solved by both *basic* and *comb*, and the horizontal axis scale shows that the magnitude of these improvements is large on most instances. Figure 6 shows

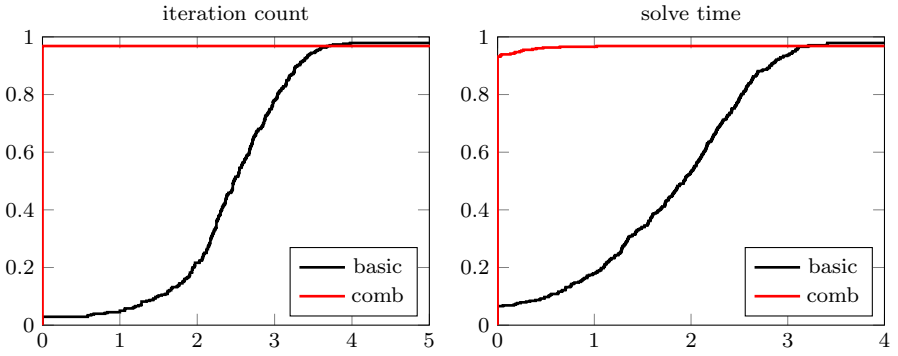

Fig. 4 Performance profiles (see Sect. 7.2) of iteration count (left) and solve time (right) for the four stepping enhancements overall

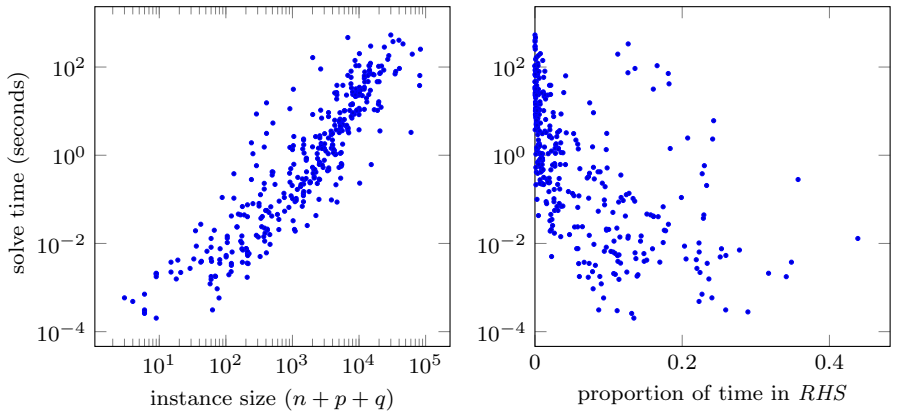

Fig. 5 Solve time (log scale, in seconds) for the *comb* stepping procedure against (left) instance size (log scale) and (right) the proportion of solve time spent in *RHS*, excluding instances that fail to converge

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.

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.

7.3.1 Less restrictive proximity

We compare *basic* and *prox* to evaluate the central path proximity enhancement introduced in Sect. 5.5.2. Figure 7 (first row) shows that the iteration count and solve time improve for nearly all instances. From Table 3, the shifted geometric means of iteration count and solve time improve by over 35%.

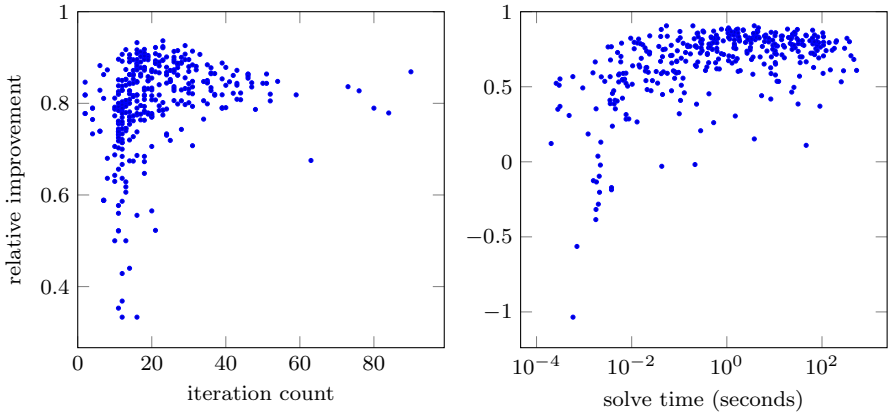

Fig. 6 Relative improvement, from *basic* to *comb*, in iteration count (left) or solve time (right) against iteration count or solve time (in seconds) respectively for *comb*, over the 356 instances on which both *basic* and *comb* converge

The similarity between the iteration count and solve time performance profiles in Fig. 7 and also between the per iteration subtimings in Table 4 suggests that the solve time improvement is driven mainly by the reduction in iteration count. The per iteration *search* time decreases slightly, since on average fewer backtracking search steps are needed per iteration for *prox* (because it tends to step further in the prediction directions, as evidenced by the smaller iteration counts). These results suggest that the central path proximity restrictions in the algorithms in [59, 64] are too conservative from the perspective of practical performance, and that we need not restrict iterates to a very small neighborhood of the central path in order to obtain high quality prediction directions in practice.

7.3.2 Third order adjustments

We compare *prox* and *TOA* to evaluate the TOA enhancement introduced in Sect. 5.5.3. Figure 7 (second row) shows that the iteration count improves for all instances and by a fairly consistent magnitude, and the solve time improves for nearly 80% of instances. From Table 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 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.

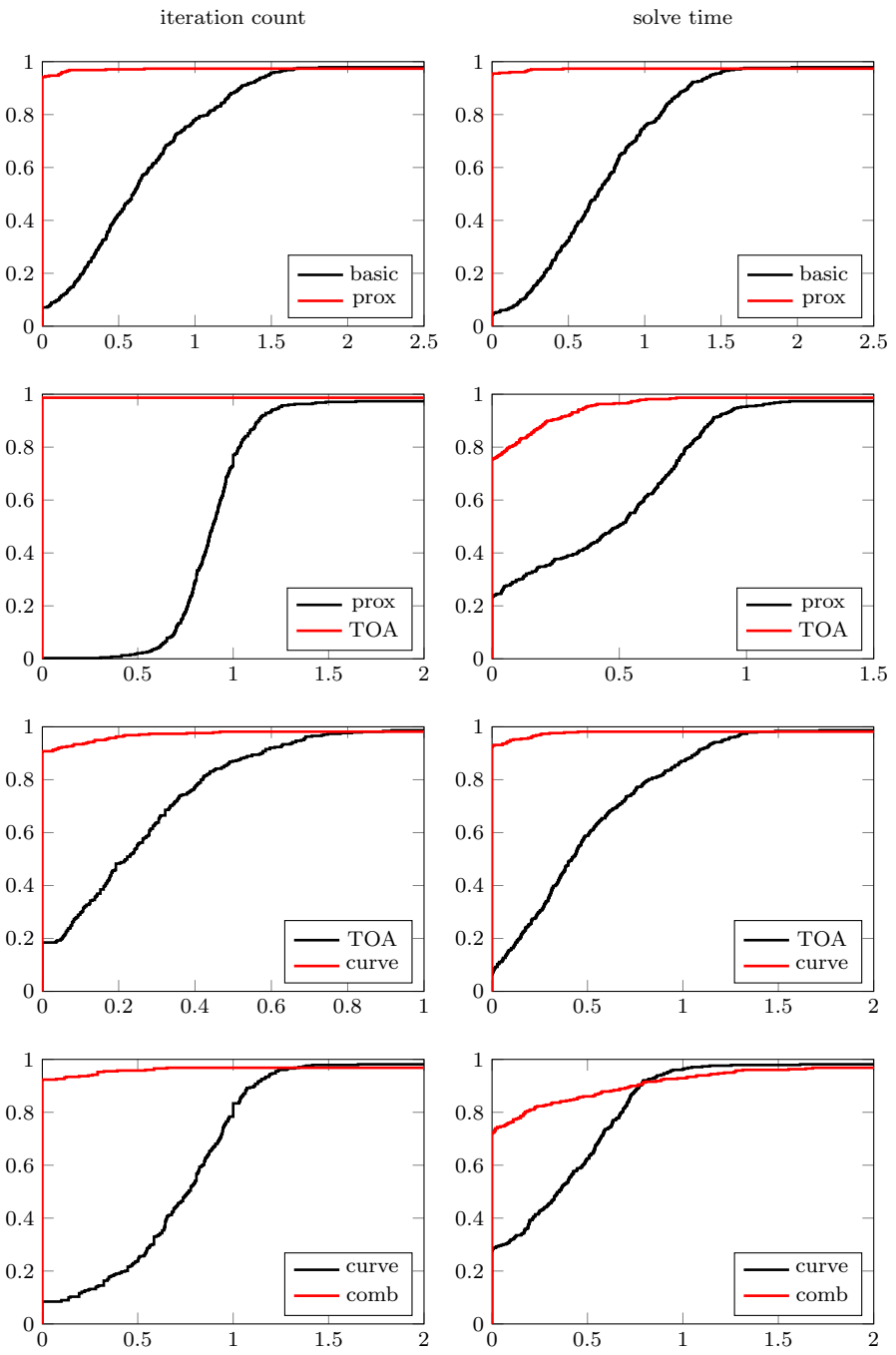

Fig. 7 Performance profiles (see Sect. 7.2) of iteration count (left column) and solve time (right column) for the four stepping enhancements (rows)

7.3.3 Curve search

We compare *TOA* and *curve* to evaluate the curve search enhancement introduced in Sect. 5.5.4. Figure 7 (third row) shows that the iteration count and solve time improve for most instances, with larger and more consistent improvements for the solve time. From Table 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 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.

7.3.4 Combined directions

Finally, we compare *curve* and *comb* to evaluate the combined directions enhancement introduced in Sect. 5.5.5. Figure 7 (fourth row) shows that the iteration count and solve time improve on around 90% and 70% of instances respectively. From Table 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 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 Appendix A). Furthermore, Figure 5 (right) shows that for most instances, *RHS* accounts for a small proportion of the overall solve time for *comb*, especially for instances that take longer to solve. This suggests that the TOO is rarely a bottleneck for our *comb* stepping procedure.

Acknowledgements The authors thank the anonymous referees for their comments and suggestions.

Funding Open Access funding provided by the MIT Libraries. This work has been partially funded by the National Science Foundation under grant OAC-1835443 and the Office of Naval Research under grant N00014-18-1-2079.

Data Availability Statement This manuscript has associated data in a data respiratory. All data analyzed are publicly available. URLs are included in this published article.

Code Availability The full code was made available for review. Specific references are included in this published article.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

A 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 Sect. 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 (6), we first rescale the primal and dual equality constraints (6b) and (7b) to improve the conditioning of the affine data. Next, we perform a QR factorization of A' and check whether any primal equalities are inconsistent (terminating if so). We use this factorization to modify c, G, h and eliminate all p primal equalities (removing dual variable y), reducing the dimension of the primal variable x from n to $n - p$. Next, we perform a QR factorization of the modified G. We use this factorization to check whether any dual equalities are inconsistent (terminating if so) and to remove any redundant dual equalities, further reducing the dimension of x. This factorization also allows us to cheaply compute an initial x^0 satisfying (15c). Since y is eliminated, we do not need to solve (16b) for y^0.

Starting from the initial interior point ω^0 defined in Sect. 5.1, we perform PDIPM iterations until the convergence conditions in Sect. 5.3 (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 (26) 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 Sect. 5.5. Instead of factorizing the large square nonsymmetric block-sparse LHS matrix, we utilize its structure to reduce the size of the factorization needed. Some of these techniques are adapted from methods in CVXOPT (see [66, 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 perfor-

mance 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 (26), 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 Appendix C and [18], these optional oracles can be more efficient and numerically stable to compute for many exotic cones. For cones without these oracles, Hypatia calls the explicit Hessian matrix oracle, performing a Cholesky factorization of the Hessian if necessary. A deeper discussion of Hypatia's linear system solving techniques and optional cone oracles is outside the scope of this paper.

B Efficient proximity checks

Recall that each stepping procedure in Sect. 5.5 uses at least one backtracking search (on a line or a curve) to find a point ω satisfying an aggregate proximity condition: $\pi_{\ell_2}(\omega) \leq \beta_1$ for the *basic* procedure in Sect. 5.5.1 or $\pi_{\ell_\infty}(\omega) \leq \beta_2$ for the procedures in Sects. 5.5.2 to 5.5.5. In Sect. 5.2, we define π_{ℓ_2} and π_{ℓ_∞} in (20) and (21). For each primitive cone $k \in [\![\bar{K}]\!]$, $0 \leq \pi_k(\omega) \leq \pi_{\ell_\infty}(\omega) \leq \pi_{\ell_2}(\omega)$, and by Lemma 1, $\pi_k(\omega) < 1$ implies $\bar{s}_k \in \operatorname{int}(\bar{\mathcal{K}}_k)$ and $\bar{z}_k \in \operatorname{int}(\bar{\mathcal{K}}_k^*)$. We use a schedule of decreasing trial values for the step parameter α and accept the first value that yields a candidate point satisfying the aggregate proximity condition.

Suppose at a particular iteration of the backtracking search, we have the candidate point ω. We check a sequence of increasingly expensive conditions that are necessary for the proximity condition to hold for ω. First, we verify that $\bar{s}_k' \bar{z}_k > 0, \forall k \in [\![\bar{K}]\!]$, which is necessary for interiority (by a strict version of the dual cone inequality (1)). Note that this condition implies $\mu(\omega) > 0$. Next, we verify that $\rho_k(\omega) < \beta, \forall k \in [\![\bar{K}]\!]$, where $\rho_k(\omega)$ is:

$$\rho_k(\omega) := \nu_k^{-1/2} |\bar{s}_k' \bar{z}_k / \mu - \nu_k| \geq 0. \tag{43}$$

In Lemma 2 below, we show that $\rho_k(\omega)$ is a lower bound on $\pi_k(\omega)$, so if $\rho_k(\omega) > \beta$ then then $\pi_k(\omega) > \beta$. Computing ρ_k is much cheaper than computing $\pi_k(\omega)$ as it does not require evaluating any cone oracles.

Next, we iterate over $k \in [\![\bar{K}]\!]$ to check first the primal feasibility oracle, then the optional dual feasibility oracle if implemented, and finally the proximity condition $\pi_k(\omega) < \beta$. Before computing $\pi_k(\omega)$, we check that the gradient and Hessian oracle evaluations approximately satisfy two logarithmic homogeneity conditions [49, Proposition 2.3.4]:

$$(g_k(\bar{s}_k))'(H_k(\bar{s}_k))^{-1} g_k(\bar{s}_k) = -\bar{s}_k' g_k(\bar{s}_k) = \nu_k. \tag{44}$$

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

Lemma 2 *Given a point ω for which $\mu(\omega) > 0$, for each $k \in [\![\bar{K}]\!]$, $0 \le \rho_k(\omega) \le \pi_k(\omega)$.*

Proof We fix $\mu = \mu(\omega) > 0$ for convenience. Let f_k be the ν_k-LHSCB for $\bar{\mathcal{K}}_k$, and let the conjugate of f_k be f_k^* (see (3)), which is a ν_k-LHSCB for $\bar{\mathcal{K}}_k^*$. Let $g_k^* \coloneqq \nabla f_k^*$ and $H_k^* \coloneqq \nabla^2 f_k^*$ denote the gradient and Hessian operators for f_k^*. Using the logarithmic homogeneity properties from [49, Proposition 2.3.4], and from the definition of $\pi_k(\omega)$ in (18), we have:

$$
\begin{aligned}
(\pi_k(\omega))^2 &= (\bar{z}_k/\mu + g_k(\bar{s}_k))'(H_k(\bar{s}_k))^{-1}(\bar{z}_k/\mu + g_k(\bar{s}_k)) && \text{(45a)}\\
&= \mu^{-2}\bar{z}_k'(H_k(\bar{s}_k))^{-1}\bar{z}_k + 2\mu^{-1}\bar{z}_k'(H_k(\bar{s}_k))^{-1}g_k(\bar{s}_k)\\
&\qquad + (g_k(\bar{s}_k))'(H_k(\bar{s}_k))^{-1}g_k(\bar{s}_k) && \text{(45b)}\\
&= \mu^{-2}\bar{z}_k'(H_k(\bar{s}_k))^{-1}\bar{z}_k - 2\mu^{-1}\bar{z}_k'\bar{s}_k + \nu_k. && \text{(45c)}
\end{aligned}
$$

By [56, Equation 13], $(H_k(\bar{s}_k))^{-1} = H_k^*(-g_k(\bar{s}_k))$. Since f_k^* is a self-concordant barrier with parameter ν_k, by [48, Equation 5.3.6] we have: $(\bar{z}_k' g_k^*(-g_k(\bar{s}_k)))^2 \le \nu_k \bar{z}_k' H_k^*(-g_k(\bar{s}_k))\bar{z}_k$. Furthermore, $g_k^*(-g_k(\bar{s}_k)) = \bar{s}_k$. Using these facts, from (45) we have $\rho_k(\omega) \ge 0$ and:

$$
\begin{aligned}
(\pi_k(\omega))^2 &= \mu^{-2}\bar{z}_k' H_k^*(-g_k(\bar{s}_k))\bar{z}_k - 2\mu^{-1}\bar{z}_k'\bar{s}_k + \nu_k && \text{(46a)}\\
&\ge \nu_k^{-1}\mu^{-2}(\bar{z}_k'\bar{s}_k)^2 - 2\mu^{-1}\bar{z}_k'\bar{s}_k + \nu_k && \text{(46b)}\\
&= \nu_k^{-1}(\bar{s}_k'\bar{z}_k/\mu - \nu_k) && \text{(46c)}\\
&= (\rho_k(\omega))^2. && \text{(46d)}
\end{aligned}
$$

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

As an aside, we can use similar arguments to Lemma 2 to show that $\rho_k(\omega)$ also symmetrically bounds a conjugate proximity measure $\pi_k^*(\omega)$, which we define as:

$$
\pi_k^*(\omega) \coloneqq \left\| (H_k^*(\bar{z}_k))^{-1/2}(\bar{s}_k/\mu + g_k^*(\bar{z}_k)) \right\| \ge \nu_k^{-1/2}|\bar{z}_k'(\bar{s}_k/\mu + g_k^*(\bar{z}_k))| = \rho_k(\omega). \tag{47}
$$

In general, we cannot check whether $\pi_k^*(\omega) < \beta$ because as we discuss in Sect. 1.3 we do not have access to fast and numerically stable conjugate barrier oracles (g_k^* and H_k^*).

C Computing the TOO for some exotic cones

C.1 Intersections of slices of the PSD cone

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

$$\mathcal{K} := \{s \in \mathbb{R}^q : \Lambda(s) \succeq 0\}, \tag{48}$$

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

$$\mathcal{K}^* := \{s \in \mathbb{R}^q : \exists S \succeq 0, s = \Lambda^*(S)\}. \tag{49}$$

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

For $\mathcal{K}$ we have the LHSCB $f(s) = -\operatorname{logdet}(\Lambda(s))$ with parameter $\nu = J$. Given a point $s \in \operatorname{int}(\mathcal{K})$, we have $\Lambda(s) \in \mathbb{S}^J_\succ$ and its inverse $\Lambda^{-1}(s) \in \mathbb{S}^J_\succ$. 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 [57, Section 3]):

$$g(s) = -\Lambda^*(\Lambda^{-1}(s)), \tag{50a}$$

$$H(s)\delta = \Lambda^*(\Lambda^{-1}(s)\Lambda(\delta)\Lambda^{-1}(s)), \tag{50b}$$

$$\mathrm{T}(s, \delta) = \Lambda^*(\Lambda^{-1}(s)\Lambda(\delta)\Lambda^{-1}(s)\Lambda(\delta)\Lambda^{-1}(s)). \tag{50c}$$

If we have, for example, a Cholesky factorization $\Lambda(s) = LL'$ (computed during the feasibility check), then the oracles in (50) are easy to compute if Λ and Λ^* are easy to apply. We can compute the TOO (50c) using the following steps:

$$Y := L^{-1}\Lambda(\delta)\Lambda^{-1}(s), \tag{51a}$$

$$Z := Y'Y = \Lambda^{-1}(s)\Lambda(\delta)\Lambda^{-1}(s)\Lambda(\delta)\Lambda^{-1}(s), \tag{51b}$$

$$\mathrm{T}(s, \delta) = \Lambda^*(Z). \tag{51c}$$

We note (51a) can be computed using back-substitutions with L, and (51b) is a simple symmetric outer product. We use this approach to derive simple TOO procedures for $\mathcal{K}_{\mathrm{LMI}}$ in Appendix C.2 and for $\mathcal{K}^*_{\mathrm{SOS}}$ and $\mathcal{K}^*_{\mathrm{matSOS}}$ in Appendix C.3 when $r = 1$.

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

$$\mathcal{K} := \{s \in \mathbb{R}^q : \Lambda_l(s) \succeq 0, \forall l \in [\![r]\!]\}, \tag{52}$$

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

$$\mathcal{K}^* := \left\{ s \in \mathbb{R}^q : \exists S_1, \ldots, S_r \succeq 0, s = \textstyle\sum_{l \in [\![r]\!]} \Lambda_l^*(S_l) \right\}. \tag{53}$$

Feasibility for $\mathcal{K}$ can be checked by performing r Cholesky factorizations. If we let $f_l(s) = -\operatorname{logdet}(\Lambda_l(s)), \forall l \in [\![r]\!]$, then $f(s) = \sum_{l \in [\![r]\!]} f_l(s)$ is an LHSCB for $\mathcal{K}$ with parameter $\nu = \sum_{l \in [\![r]\!]} J_l$. Clearly, $g(s)$, $H(s)\delta$ (and the explicit Hessian matrix), and $\mathrm{T}(s, \delta)$ can all be computed as sums over $l \in [\![r]\!]$ of the terms in (50c).

C.2 LMI cone

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

$$\Lambda(w) := \textstyle\sum_{i \in [\![d]\!]} w_i P_i \in \mathbb{S}^s, \tag{54a}$$

$$\Lambda^*(W) := (\langle P_i, W \rangle)_{i \in [\![d]\!]} \in \mathbb{R}^d. \tag{54b}$$

Our implementation uses specializations of (50) and (51) for $\mathcal{K}_{\mathrm{LMI}}$. For $w \in \operatorname{int}(\mathcal{K}_{\mathrm{LMI}})$ and direction $\delta \in \mathbb{R}^d$, using the Cholesky factorization $\Lambda(w) = LL'$, we compute:

$$Q_i := L^{-1} P_i (L^{-1})' \in \mathbb{S}^s \quad \forall i \in [\![d]\!], \tag{55a}$$

$$g(w) = (-\operatorname{tr}(Q_i))_{i \in [\![d]\!]}, \tag{55b}$$

$$R := \textstyle\sum_{j \in [\![d]\!]} \delta_j Q_j \in \mathbb{S}^s, \tag{55c}$$

$$H(w)\delta = (\langle Q_i, R \rangle)_{i \in [\![d]\!]}, \tag{55d}$$

$$\mathrm{T}(w, \delta) = (\langle Q_i, R'R \rangle)_{i \in [\![d]\!]}, \tag{55e}$$

and we compute the explicit Hessian oracle as:

$$(H(w))_{i,j} = \langle Q_i, Q_j \rangle \quad \forall i, j \in [\![d]\!]. \tag{56}$$

The symmetric form of Q_i and the use of a symmetric outer product $R'R$ in (55e) are beneficial for efficiency and numerical performance.

C.3 Matrix and scalar WSOS dual cones

Recall that Hypatia uses LHSCBs for $\mathcal{K}^*_{\mathrm{SOS}}, \mathcal{K}^*_{\mathrm{matSOS}}$, because LHSCBs for $\mathcal{K}_{\mathrm{SOS}}$, $\mathcal{K}_{\mathrm{matSOS}}$ with tractable oracles are not known (see [35]). Since the scalar WSOS dual cone $\mathcal{K}^*_{\mathrm{SOS}}$ is a special case of the matrix WSOS dual cone $\mathcal{K}^*_{\mathrm{matSOS}}$ with $t = 1$, we only consider $\mathcal{K}^*_{\mathrm{matSOS}}$ here. In general, $\mathcal{K}^*_{\mathrm{matSOS}}$ is an intersection of r slices of $\mathcal{K}_{\succeq}$ (see (52)), so the gradient, Hessian, and TOO oracles are all additive; for simplicity, we only consider $r = 1$ (and $s_1 = s$, $P_1 = P$) below.

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

$$\rho_{i,j} \coloneqq \begin{cases} 1 & \text{if } i = j, \\ \sqrt{2} & \text{otherwise.} \end{cases} \tag{57}$$

For $\mathcal{K}^*_{\text{matSOS}}$ 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 $W \in \mathbb{S}^{st}$:

$$\Lambda(w) \coloneqq \left[P' \operatorname{Diag}\left(\rho_{i,j}^{-1} w_{\max(i,j),\min(i,j),:}\right) P\right]_{i,j \in [\![t]\!]} \in \mathbb{S}^{st}, \tag{58a}$$

$$\Lambda^*(W) \coloneqq (\rho_{i,j} \operatorname{diag}(P(W)_{i,j} P'))_{i \in [\![t]\!], j \in [\![t]\!]} \in \mathbb{R}^{\text{sd}(t)d}, \tag{58b}$$

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

We implement efficient and numerically stable specializations of the oracles in (50) and (51). Suppose we have $w \in \operatorname{int}(\mathcal{K}^*_{\text{matSOS}})$ and direction $\delta \in \mathbb{R}^{\text{sd}(t)d}$, and a Cholesky factorization $\Lambda(w) = LL'$. For each $i, j \in [\![t]\!] : i \geq j$ and $p \in [\![d]\!]$, we implicitly compute oracles according to:

$$(Q)_{i,j,p} \coloneqq ((L^{-1})_{i,j} P') e_p \in \mathbb{R}^s, \tag{59a}$$

$$(g(w))_{i,j,p} = -\rho_{i,j} Q'_{i,:,p} Q_{:,j,p}, \tag{59b}$$

$$(R)_{i,j,p} \coloneqq (L^{-1} \Lambda(\delta) (L^{-1})' Q)_{i,j} e_p \in \mathbb{R}^s, \tag{59c}$$

$$(H(w)\delta)_{i,j,p} = \rho_{i,j} Q'_{i,:,p} R_{:,j,p}, \tag{59d}$$

$$(\mathrm{T}(w, \delta))_{i,j,p} = \rho_{i,j} R'_{i,:,p} R_{:,j,p}. \tag{59e}$$

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

$$(H(w))_{(i,j,:),(k,l,:)} = \tfrac{1}{2} \rho_{i,j} \rho_{k,l} \left(Q^2_{i,k} \circ Q^2_{j,l} + Q^2_{i,l} \circ Q^2_{j,k}\right) \in \mathbb{S}^d \quad \forall i, j, k, l \in [\![t]\!], \tag{60}$$

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

C.4 Sparse PSD cone

Let $\mathcal{S} = ((i_l, j_l))_{l \in [\![d]\!]}$ be a collection of row-column index pairs defining the sparsity pattern of the lower triangle of a symmetric matrix of side dimension s (including all diagonal elements). We do not require $\mathcal{S}$ to be a chordal sparsity pattern (unlike [4, 12]), as this restriction is not necessary for the oracles Hypatia uses. Note $s \leq d \leq \text{sd}(s)$. For $\mathcal{K}_{\text{sPSD}}$ parametrized by $\mathcal{S}$, we define $\Lambda : \mathbb{R}^d \to \mathbb{S}^s$ as the linear operator satisfying,

for all $i, j \in [\![s]\!] : i \geq j$:

$$(\Lambda(w))_{i,j} := \begin{cases} \rho_{i,j}^{-1} w_l & \text{if } i = i_l = j = j_l, \\ 0 & \text{otherwise,} \end{cases} \tag{61}$$

where $\rho_{i,j}$ is given by (57). Then Λ^* is the vectorized projection onto $\mathcal{S}$, i.e. for $W \in \mathbb{S}^s$:

$$\Lambda^*(W) := (\rho_{i,j} W_{i,j})_{(i,j)\in\mathcal{S}} \in \mathbb{R}^d. \tag{62}$$

Consider $w \in \text{int}(\mathcal{K}_{\text{sPSD}})$ and direction $\delta \in \mathbb{R}^d$. The gradient (50a) and Hessian product (50b) for $\mathcal{K}_{\text{sPSD}}$ can be computed using [4, Algorithms 4.1 and 5.1]. To derive the TOO, we use the fact that:

$$-2\text{T}(w, \delta) = \nabla^3 f(w)[\delta, \delta] = \tfrac{d^2}{dt^2} \nabla f(w + t\delta)\big|_{t=0}. \tag{63}$$

In order to succinctly describe our TOO approach as an extension of the procedures in [4], we describe an approach based on a sparse LDL factorization of $\Lambda(w)$. However, our current implementation in Hypatia uses a sparse Cholesky (LL') factorization, which is very similar to the LDL-based approach here. We compute the sparse Cholesky factors using Julia's SuiteSparse wrapper of CHOLMOD [17]. We note that Hypatia implements a *supernodal* generalization (see [4, Section 7]) of the TOO procedure we describe below. Before we describe the TOO procedure, we repeat useful definitions from [4], define higher order derivative terms, and differentiate several equations that are used for the gradient and Hessian oracles. As discussed in Sect. 6, Hypatia computes the feasibility check and gradient oracles before the TOO, and our TOO procedure reuses cached values computed for these oracles.

We define:

$$R := \Lambda(\nabla f(w + t\delta)). \tag{64}$$

Let $LDL' = \Lambda(w)$ be a sparse LDL factorization, i.e. L is a sparse unit lower triangular matrix and D is a positive definite diagonal matrix. The sparsity pattern of L is associated with an *elimination tree* [4, Section 2], and each column of L corresponds to a node of this tree. Let $\mathcal{I}_k$ be the ordered row indices of nonzeros below the diagonal in column k of L, and let $\mathcal{J}_k = \mathcal{I}_k \cup \{k\}$. Let $\text{ch}(i)$ denote the children of node i in the tree. For an index set $\mathcal{I}$ let $\mathcal{I}(i)$ denote the ith element. For index sets $\mathcal{J} \subset \mathcal{I}$, we define $E_{\mathcal{I},\mathcal{J}} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{J}|}$ satisfying, $i \in [\![|\mathcal{I}|]\!]$, $j \in [\![|\mathcal{J}|]\!]$:

$$(E_{\mathcal{I},\mathcal{J}})_{i,j} := \begin{cases} 1 & \text{if } \mathcal{I}(i) = \mathcal{J}(j), \\ 0 & \text{otherwise.} \end{cases} \tag{65}$$

Let U_i be the update matrix for node i (see [4, Equation 14]):

$$U_i := -\textstyle\sum_{k\in\text{ch}(i)\cup\{i\}} D_{k,k} L_{\mathcal{I}_i,k} L'_{\mathcal{I}_i,k}. \tag{66}$$

Let $\dot{D}, \dot{L}, \dot{U}, \dot{R}$ and $\ddot{D}, \ddot{L}, \ddot{U}, \ddot{R}$ denote the first and second derivatives of D, L, U, R with respect to the linearization variable t in (63). For convenience, we let:

$$\bar{L}_j \coloneqq \begin{bmatrix} 1 & 0 \\ -L_{\mathcal{I}_j,j} & I(d) \end{bmatrix}. \tag{67}$$

Suppose we have computed $\dot{D}, \dot{L}, \dot{U}$ according to [4, Equation 30]. Differentiating [4, Equation 30] once with respect to t gives:

$$\begin{bmatrix} \ddot{D}_{j,j} & P'_j \\ P_j & 2D_{j,j}\dot{L}_{\mathcal{I}_j,j}\dot{L}'_{\mathcal{I}_j,j} + \ddot{U}_j \end{bmatrix} = \bar{L}_j\big(\textstyle\sum_{i\in\mathrm{ch}(j)} E_{\mathcal{J}_j,\mathcal{I}_i}\ddot{U}_i E'_{\mathcal{J}_j,\mathcal{I}_i}\big)\bar{L}'_j, \tag{68}$$

where $P_j \coloneqq 2\dot{D}_{j,j}\dot{L}_{\mathcal{I}_j,j} + D_{j,j}\ddot{L}_{\mathcal{I}_j,j}$ for convenience. This allows us to compute $\ddot{D}$, $\ddot{L}$, $\ddot{U}$. [4, Equations 21 and 22] show that:

$$R_{\mathcal{I}_j,j} = -R_{\mathcal{I}_j,\mathcal{I}_j}L_{\mathcal{I}_j,j}, \tag{69a}$$

$$\begin{bmatrix} R_{j,j} & R'_{\mathcal{I}_j,j} \\ R_{\mathcal{I}_j,j} & R_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix}\begin{bmatrix} 1 \\ L_{\mathcal{I}_j,j} \end{bmatrix} = \begin{bmatrix} D_{j,j}^{-1} \\ 0 \end{bmatrix}, \tag{69b}$$

for each node j. Differentiating (69a) once with respect to t gives:

$$\dot{R}_{\mathcal{I}_j,j} = -R_{\mathcal{I}_j,\mathcal{I}_i}\dot{L}_{\mathcal{I}_j,j} - \dot{R}_{\mathcal{I}_j,\mathcal{I}_j}L_{\mathcal{I}_j,j}. \tag{70}$$

Differentiating (69b) twice and substituting (69a) and (70), we have:

$$\begin{bmatrix} \ddot{R}_{j,j} & \ddot{R}'_{\mathcal{I}_j,j} \\ \ddot{R}_{\mathcal{I}_j,j} & \ddot{R}_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix} = \bar{L}'_j\begin{bmatrix} 2\dot{D}_{j,j}^2 D_{j,j}^{-3} - \ddot{D}_{j,j}D_{j,j}^{-2} + 2\dot{L}'_{\mathcal{I}_j,j}R_{\mathcal{I}_j,\mathcal{I}_j}\dot{L}_{\mathcal{I}_j,j} & Q'_j \\ Q_j & \ddot{R}_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix}\bar{L}_j, \tag{71}$$

where $Q_j \coloneqq -R_{\mathcal{I}_j,\mathcal{I}_j}\ddot{L}_{\mathcal{I}_j,j} - 2\dot{R}_{\mathcal{I}_j,\mathcal{I}_j}\dot{L}_{\mathcal{I}_j,j}$ for convenience. This allows us to compute $\ddot{R}$. Finally, by (63) and (64), we can compute the TOO as:

$$-2\mathrm{T}(w,\delta) = \Lambda^*\big(\ddot{R}\big). \tag{72}$$

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

$$\Delta = \Lambda(\delta) \in \mathbb{S}^s. \tag{73}$$

Following [2], we define K and M as sparse matrices with the same structure as L, satisfying for all $j \in [\![s]\!]$:

$$K_{j,j} = \dot{D}_{j,j}, \tag{74a}$$

$$K_{\mathcal{I}_j,j} = D_{j,j}\dot{L}_{\mathcal{I}_j,j}, \tag{74b}$$

$$M_{j,j} = D_{j,j}^{-2}K_{j,j}, \tag{74c}$$

$$M_{\mathcal{I}_j,j} = D_{j,j}^{-1} R_{\mathcal{I}_j,\mathcal{I}_j} K_{\mathcal{I}_j,j}. \tag{74d}$$

The first three steps in the TOO procedure below compute $\dot{D}$, $\dot{L}$, $\dot{U}$, and $\dot{R}$ and are identical to steps in [4, Algorithm 5.1].

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

$$\begin{bmatrix} K_{j,j} & K'_{\mathcal{I}_j,j} \\ K_{\mathcal{I}_j,j} & U'_j \end{bmatrix} = \bar{L}_j \left(\begin{bmatrix} \Delta_{j,j} & \Delta'_{\mathcal{I}_j,j} \\ \Delta_{\mathcal{I}_j,j} & 0 \end{bmatrix} + \textstyle\sum_{i \in \mathrm{ch}(j)} E_{\mathcal{J}_j,\mathcal{I}_i} U'_i E'_{\mathcal{J}_j,\mathcal{I}_i} \right) \bar{L}'_j. \tag{75}$$

2. For $j \in [\![s]\!]$, store $\dot{D}_{j,j}$ and $\dot{L}_{\mathcal{I}_j,j}$ from (74a) and (74b), and compute $M_{\mathcal{J}_j,j}$ from (74c) and (74d).
3. Iterate over $j \in [\![s]\!]$ in reverse topological order, computing $\dot{R}_{\mathcal{J}_j,j}$ according to:

$$\begin{bmatrix} \dot{R}_{j,j} & \dot{R}'_{\mathcal{I}_j,j} \\ \dot{R}_{\mathcal{I}_j,j} & \dot{R}_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix} = \bar{L}'_j \begin{bmatrix} M_{j,j} & M'_{\mathcal{I}_j,j} \\ M_{\mathcal{I}_j,j} & \dot{R}_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix} \bar{L}_j, \tag{76}$$

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

$$\dot{R}_{\mathcal{I}_i,\mathcal{I}_i} = E'_{\mathcal{J}_j,\mathcal{I}_i} \begin{bmatrix} \dot{R}_{j,j} & \dot{R}'_{\mathcal{I}_j,j} \\ \dot{R}_{\mathcal{I}_j,j} & \dot{R}_{\mathcal{I}_j,\mathcal{I}_j} \end{bmatrix} E_{\mathcal{J}_j,\mathcal{I}_i}. \tag{77}$$

4. Iterate over $j \in [\![s]\!]$ in topological order, computing $\ddot{D}_{j,j}$, $\ddot{L}_{\mathcal{I}_j,j}$, $\ddot{U}_j$ from (68).
5. Iterate over $j \in [\![s]\!]$ in reverse topological order, computing $\ddot{R}_{j,j}$, $\ddot{R}_{\mathcal{I}_j,j}$, $\ddot{R}_{\mathcal{I}_j,\mathcal{I}_j}$ from (71).
6. Compute $\mathrm{T}(w, \delta)$ using $\ddot{R}$ and (72).

C.5 Euclidean norm cone and Euclidean norm square cone

Although $\mathcal{K}_{\ell_2}, \mathcal{K}_{\mathrm{sqr}} \subset \mathbb{R}^q$ are inverse linear images of $\mathbb{S}^q_{\succeq}$ and hence admit LHSCBs with parameter $\nu = q$, we use the standard LHSCBs with parameter $\nu = 2$, which have a different form (see [66, Section 2.2]). For $\mathcal{K}_{\ell_2}, \mathcal{K}_{\mathrm{sqr}}$, the LHSCB is $f(s) = -\log(s'Js)$, where $J \in \mathbb{S}^q$ is defined according to, for $i, j \in [\![q]\!] : i \geq j$:

$$J_{i,j} \coloneqq \begin{cases} 1 & \text{if } j = 1 \text{ and } \big(i = 1 \text{ for } \mathcal{K}_{\ell_2} \text{ or } i = 2 \text{ for } \mathcal{K}_{\mathrm{sqr}}\big), \\ -1 & \text{if } i = j \text{ and } \big(i > 1 \text{ for } \mathcal{K}_{\ell_2} \text{ or } i > 2 \text{ for } \mathcal{K}_{\mathrm{sqr}}\big), \\ 0 & \text{otherwise.} \end{cases} \tag{78}$$

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

$$g(s) = -2\bar{J}Js, \tag{79a}$$

$$H(s)\delta = 2\bar{J}(2\bar{J}Jss'J\delta - J\delta), \tag{79b}$$

$$\mathrm{T}(s,\delta) = \bar{J}(Js\delta'H\delta + H\delta s'J\delta - s'H\delta J\delta). \tag{79c}$$

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

$$H(s) = 2\bar{J}(2\bar{J}Jss'J - J). \tag{80}$$

References

1. Agrawal, A., Diamond, S., Boyd, S.: Disciplined geometric programming. Optimization Letters **13**(5), 961–976 (2019)
2. Andersen, E.D., Roos, C., Terlaky, T.: On implementing a primal-dual interior-point method for conic quadratic optimization. Math. Program. **95**(2), 249–277 (2003)
3. Andersen, M., Dahl, J., Liu, Z., Vandenberghe, L., Sra, S., Nowozin, S., Wright, S.: Interior-point methods for large-scale cone programming. In: Sra, S., Wright, S.J., Nowozin, S. (eds.) Optimization for Machine Learning, vol. 5583. MIT Press Cambridge, MA (2011)
4. Andersen, M.S., Dahl, J., Vandenberghe, L.: Logarithmic barriers for sparse matrix cones. Optimization Methods and Software **28**(3), 396–423 (2013)
5. Anh Truong, V., Tunçel, L.: Geometry of homogeneous convex cones, duality mapping, and optimal self-concordant barriers. Math. Program. **100**(2), 295–316 (2004)
6. Aylward, E.M., Parrilo, P.A., Slotine, J.J.E.: Stability and robustness analysis of nonlinear systems via contraction metrics and SOS programming. Automatica **44**(8), 2163–2170 (2008)
7. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017)
8. Borchers, B.: CSDP, a C library for semidefinite programming. Optimization Methods and Software **11**(1–4), 613–623 (1999)
9. Boyd, S.: EE363 review session 4: Linear matrix inequalities. University Lecture (2009). https://stanford.edu/class/ee363/sessions/s4notes.pdf. Online, accessed 30-July-2022
10. Boyd, S., El Ghaoui, L., Feron, E., Balakrishnan, V.: Linear Matrix Inequalities in System and Control Theory. Studies in Applied and Numerical Mathematics, vol. 15. SIAM (1994)
11. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge University Press (2004)
12. Burer, S.: Semidefinite programming in the space of partial positive semidefinite matrices. SIAM J. Optim. **14**(1), 139–172 (2003)
13. Burkardt, J.: Polynomials for global optimization tests (2016). https://people.sc.fsu.edu/~jburkardt/py_src/polynomials/polynomials.html. Online, accessed 30-July-2022
14. Chandrasekaran, V., Shah, P.: Relative entropy relaxations for signomial optimization. SIAM J. Optim. **26**(2), 1147–1173 (2016)
15. Chandrasekaran, V., Shah, P.: Relative entropy optimization and its applications. Math. Program. **161**(1–2), 1–32 (2017)
16. Chares, R.: Cones and interior-point algorithms for structured convex optimization involving powers and exponentials. Ph.D. thesis, Université Catholique de Louvain (2009)
17. Chen, Y., Davis, T.A., Hager, W.W., Rajamanickam, S.: Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. ACM Transactions on Mathematical Software (TOMS) **35**(3), 1–14 (2008)
18. Coey, C., Kapelevich, L., Vielma, J.P.: Conic optimization with spectral functions on Euclidean Jordan algebras. arXiv:2103.04104 (2021)
19. Coey, C., Kapelevich, L., Vielma, J.P.: Hypatia documentation (2022). https://chriscoey.github.io/Hypatia.jl/dev/. Online, accessed 30-July-2022
20. Coey, C., Kapelevich, L., Vielma, J.P.: Solving natural conic formulations with Hypatia. jl. INFORMS Journal on Computing (2022). https://doi.org/10.1287/ijoc.2022.1202
21. Dahl, J., Andersen, E.D.: A primal-dual interior-point algorithm for nonsymmetric exponential-cone optimization. Math. Program. **194**(1–2), 341–370 (2022)

22. d'Aspremont, A., El Ghaoui, L., Jordan, M.I., Lanckriet, G.R.: A direct formulation for sparse PCA using semidefinite programming. SIAM Rev. **49**(3), 434–448 (2007)
23. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002)
24. Domahidi, A., Chu, E., Boyd, S.: ECOS: an SOCP solver for embedded systems. In: 2013 European Control Conference (ECC), pp. 3071–3076. IEEE (2013)
25. Dunning, I., Huchette, J., Lubin, M.: JuMP: a modeling language for mathematical optimization. SIAM Rev. **59**(2), 295–320 (2017)
26. Fawzi, H., Fawzi, O.: Efficient optimization of the quantum relative entropy. J. Phys. A: Math. Theor. **51**(15), 154003 (2018)
27. Fawzi, H., Saunderson, J.: Optimal self-concordant barriers for quantum relative entropies. arXiv:2205.04581 (2022)
28. Fawzi, H., Saunderson, J., Parrilo, P.A.: Semidefinite approximations of the matrix logarithm. Found. Comput. Math. **19**(2), 259–296 (2019)
29. Fleming, P.J., Wallace, J.J.: How not to lie with statistics: the correct way to summarize benchmark results. Commun. ACM **29**(3), 218–221 (1986)
30. Friberg, H.A.: CBLIB 2014: A benchmark library for conic mixed-integer and continuous optimization. Math. Program. Comput. **8**(2), 191–214 (2016)
31. Gould, N., Scott, J.: A note on performance profiles for benchmarking software. ACM Transactions on Mathematical Software (TOMS) **43**(2), 1–5 (2016)
32. Güler, O.: Barrier functions in interior point methods. Math. Oper. Res. **21**(4), 860–885 (1996)
33. Güler, O., Tunçel, L.: Characterization of the barrier parameter of homogeneous convex cones. Math. Program. **81**(1), 55–76 (1998)
34. Henrion, D., Korda, M.: Convex computation of the region of attraction of polynomial control systems. IEEE Trans. Autom. Control **59**(2), 297–312 (2013)
35. Kapelevich, L., Coey, C., Vielma, J.P.: Sum of squares generalizations for conic sets. Math. Program. (2022). https://doi.org/10.1007/s10107-022-01831-6
36. Karimi, M., Tunçel, L.: Domain-Driven Solver (DDS) version 2.0: a MATLAB-based software package for convex optimization problems in domain-driven form. arXiv:1908.03075 (2020)
37. Karimi, M., Tunçel, L.: Primal-dual interior-point methods for domain-driven formulations. Math. Oper. Res. **45**(2), 591–621 (2020)
38. Korda, M., Henrion, D., Jones, C.N.: Controller design and value function approximation for nonlinear dynamical systems. Automatica **67**, 54–66 (2016)
39. Laurent, M., Piovesan, T.: Conic approach to quantum graph parameters using linear optimization over the completely positive semidefinite cone. SIAM J. Optim. **25**(4), 2461–2493 (2015)
40. Mazumder, R., Choudhury, A., Iyengar, G., Sen, B.: A computational framework for multivariate convex regression and its variants. J. Am. Stat. Assoc. **114**(525), 318–331 (2019)
41. Mehrotra, S.: On the implementation of a primal-dual interior point method. SIAM J. Optim. **2**(4), 575–601 (1992)
42. MOSEK ApS: Modeling Cookbook Release 3.3.0 (2022). https://docs.mosek.com/modeling-cookbook/index.html. Online, accessed 30-July-2022
43. MOSEK ApS: MOSEK Optimizer API for Java 9.3.20 (2022). https://docs.mosek.com/latest/javaapi/index.html. Online, accessed 30-July-2022
44. M.S. Andersen and J. Dahl and L. Vandenberghe: CVXOPT User's Guide - Cone Programming - Algorithm Parameters (2021). https://cvxopt.org/userguide/coneprog.html#algorithm-parameters. Online, accessed 30-July-2022
45. Murray, R., Chandrasekaran, V., Wierman, A.: Signomial and polynomial optimization via relative entropy and partial dualization. Math. Program. Comput. **13**, 257–295 (2021)
46. Myklebust, T., Tunçel, L.: Interior-point algorithms for convex optimization based on primal-dual metrics. arXiv:1411.2129 (2014)
47. Nesterov, Y.: Towards non-symmetric conic optimization. Optimization Methods and Software **27**(4–5), 893–917 (2012)
48. Nesterov, Y.: Lectures on Convex Optimization. Springer Optimization and Its Applications, vol. 137. Springer Cham (2018). https://doi.org/10.1007/978-3-319-91578-4
49. Nesterov, Y., Nemirovskii, A.: Interior-point polynomial algorithms in convex programming. Studies in Applied Mathematics. Society for Industrial and Applied Mathematics (1994)

50. Nesterov, Y., Todd, M.J., Ye, Y.: Infeasible-start primal-dual methods and infeasibility detectors for nonlinear programming problems. Math. Program. **84**(2), 227–267 (1999)
51. Nesterov, Y.E., Todd, M.J.: Self-scaled barriers and interior-point methods for convex programming. Math. Oper. Res. **22**(1), 1–42 (1997)
52. Nesterov, Y.E., Todd, M.J.: Primal-dual interior-point methods for self-scaled cones. SIAM J. Optim. **8**(2), 324–364 (1998)
53. Orban, D.: BenchmarkProfiles.jl (2019). https://doi.org/10.5281/zenodo.4630955
54. O'Donoghue, B., Chu, E., Parikh, N., Boyd, S.: Conic optimization via operator splitting and homogeneous self-dual embedding. J. Optim. Theory Appl. **169**(3), 1042–1068 (2016)
55. Papp, D., Alizadeh, F.: Shape-constrained estimation using nonnegative splines. J. Comput. Graph. Stat. **23**(1), 211–231 (2014)
56. Papp, D., Yıldız, S.: On "a homogeneous interior-point algorithm for non-symmetric convex conic optimization". arxiv:1712.00492 (2018)
57. Papp, D., Yıldız, S.: Sum-of-squares optimization without semidefinite programming. SIAM J. Optim. **29**(1), 822–851 (2019)
58. Papp, D., Yıldız, S.: alfonso: ALgorithm FOr Non-Symmetric Optimization (2020). https://github.com/dpapp-github/alfonso
59. Papp, D., Yıldız, S.: Alfonso: Matlab package for nonsymmetric conic optimization. INFORMS J. Comput. **34**(1), 11–19 (2021)
60. Permenter, F., Friberg, H.A., Andersen, E.D.: Solving conic optimization problems via self-dual embedding and facial reduction: a unified approach. SIAM J. Optim. **27**(3), 1257–1282 (2017)
61. Renegar, J.: A mathematical view of interior-point methods in convex optimization. MOS-SIAM Series on Optimization. SIAM (2001)
62. Roy, S., Xiao, L.: On self-concordant barriers for generalized power cones. Optimization Letters **16**(2), 681–694 (2022)
63. Serrano, S.A.: Algorithms for unsymmetric cone optimization and an implementation for problems with the exponential cone. Ph.D. thesis, Stanford University (2015)
64. Skajaa, A., Ye, Y.: A homogeneous interior-point algorithm for nonsymmetric convex conic optimization. Math. Program. **150**(2), 391–422 (2015)
65. Sun, Y., Vandenberghe, L.: Decomposition methods for sparse matrix nearness problems. SIAM J. Matrix Anal. Appl. **36**(4), 1691–1717 (2015)
66. Vandenberghe, L.: The CVXOPT linear and quadratic cone program solvers (2010). https://www.seas.ucla.edu/~vandenbe/publications/coneprog.pdf. Online, accessed 30-July-2022
67. Xu, X., Hung, P.F., Ye, Y.: A simplified homogeneous and self-dual linear programming algorithm and its implementation. Ann. Oper. Res. **62**(1), 151–171 (1996)
68. Yamashita, M., Fujisawa, K., Kojima, M.: Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0). Optimization Methods and Software **18**(4), 491–505 (2003)