On primal-dual interior-point algorithms for convex optimisation

by

Tor Gunnar Josefsson Myklebust

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requiements for the degree of
Doctor of Philosophy
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2015

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of contributions

The bulk of this thesis was authored by me alone. Parts of Sections 2.5, 2.6, 2.7, and 2.9 and Appendix A appeared first in [56], which was written jointly by Levent Tunçel and I.

Abstract

This thesis studies the theory and implementation of interior-point methods for convex optimisation. A number of important problems from mathematics and engineering can be cast naturally as convex optimisation problems, and a great many others have useful convex relaxations. Interior-point methods are among the successful algorithms for solving convex optimisation problems. One class of interior-point methods, called primal-dual interior-point methods, have been particularly successful at solving optimisation problems defined over symmetric cones, which are self-dual cones whose linear automorphisms act transitively on their interiors.

The main theoretical contribution is the design and analysis of a primal-dual interior-point method for general convex optimisation that is "primal-dual symmetric"—if arithmetic is done exactly, the sequence of iterates generated is invariant under interchange of primal and dual problems.

The proof of this algorithm's correctness and asymptotic worst-case iteration complexity hinges on a new analysis of a certain rank-four update formula akin to the Hessian estimate updates performed by quasi-Newton methods.

This thesis also gives simple, explicit constructions of primal-dual scalings—linear maps from the dual space to the primal space that map the dual iterate to the primal iterate and the barrier gradient at the primal iterate to the barrier gradient at the dual iterate—by averaging the primal or dual Hessian over a line segment. These scalings are called the primal and dual integral scalings in this thesis. The primal and dual integral scalings can inherit certain kinds of good behaviour from the barrier whose Hessian is averaged. For instance, if the primal barrier Hessian at every point maps the primal cone into the dual cone, then the primal integral scaling also maps the primal cone into the dual cone.

This gives the idea that primal-dual interior-point methods based on the primal integral scaling might be effective on problems in which the primal barrier is somehow well-behaved, but the dual barrier is not.

One such class of problems is *hyperbolicity cone optimisation*—minimising a linear function over the intersection of an affine space with a so-called hyperbolicity cone. Hyperbolicity cones arise from hyperbolic polynomials, which can be seen as a generalisation of the determinant polynomial on symmetric matrices. Hyperbolic polynomials themselves have been of considerable recent interest in mathematics, their theory playing a role in the resolution of the Kadison-Singer problem.

In the setting of hyperbolicity cone optimisation, the primal barrier's Hessian satisfies "the long-step Hessian estimation property" with which the primal barrier may be easily estimated everywhere in the interior of the cone in terms of the primal barrier anywhere else in the interior of the cone, and the primal barrier Hessian at every point in the interior of the

cone maps the primal cone into the dual cone. In general, however, the dual barrier satisfies neither of these properties.

This thesis also describes an adaptation of the Mizuno-Todd-Ye method for linear optimisation to hyperbolicity cone optimisation and its implementation. This implementation is meant as a window into the algorithm's convergence behaviour on hyperbolicity cone optimisation problems rather than as a useful software package for solving hyperbolicity cone optimisation problems that might arise in practice.

In the final chapter of this thesis is a description of an implementation of an interior-point method for linear optimisation. This implementation can efficiently use primal-dual scalings based on rank-four updates to an old scaling matrix and was meant as a platform to evaluate that technique. This implementation is modestly slower than CPLEX's barrier optimiser on problems with no free or double-bounded variables. A computational comparison between the "standard" interior-point algorithm for solving LPs with one instance of the rank-four update technique is given.

The rank-four update formula mentioned above has an interesting specialisation to linear optimisation that is also described in this thesis. A serious effort was made to improve the running time of an interior-point method for linear optimisation using this technique, but it ultimately failed.

This thesis revisits work from the early 1990s by Rothberg and Gupta on cache-efficient data structures for Cholesky factorisation. This thesis proposes a variant of their data structure, showing that, in this variant, the time needed to perform triangular solves can be reduced substantially from the time needed by either the usual supernodal or simplicial data structures. The linear optimisation problem solver described in this thesis is also used to study the impact of these different data structures on the overall time required to solve a linear optimisation problem.

Acknowledgements

I thank my family and friends for their support. I thank my advisor, Levent Tunçel, whose guidance and support made this work possible. I also thank my committee members, namely Kenneth Davidson, James Renegar, Stephen Vavasis, and David Wagner, for their helpful feedback.

Contents

List of Figures

List of Tables

Chapter 1

Convex sets and convex optimisation

A number of important problems in mathematics and its applications can be formulated as convex optimisation problems—problems of minimising a convex objective function of finitely many variables subject to the constraint that the variables, as a vector, lie in some convex set. These are called convex optimisation problems. A number of other problems admit convex relaxations whose solutions can yield substantial insight.

Every convex optimisation problem has a dual, whose feasible solutions correspond to valid proofs of lower bounds on the optimal objective value of the primal. The objective in the dual is to prove the strongest lower bound possible using the constraints in the primal. This dual can always be expressed as a concave maximisation problem—maximise a concave function subject to convex constraints. This is a convex optimisation problem after negating the objective function.

Many convex optimisation algorithms treat the primal and dual as "equal partners" in specifying the problem and its space of solutions, generating sequences of primal solutions and dual solutions at the same time. One class of such algorithms are primal-dual interior-point methods, which are the focus of this thesis.

The purpose of this chapter is to introduce some of the theory of convex sets and of convex functions.

1.1 Convex sets

Proofs for the theorems in this section may be found in, for example, the first chapter of Schneider's book [80].

Definition 1.1.1. Let $x \in \mathbb{R}^d$ and $r > 0$. The open *ball* of radius r centred at x, written

$B(x, r)$, is the set of points y such that $||x - y|| < r$. The norm here is the usual Euclidean 2-norm.

Let $S \subseteq \mathbb{R}^d$. The *interior* of S, written $\operatorname{int} S$, is the set of points $x \in \mathbb{R}^d$ such that there exists an $\epsilon > 0$ for which $B(x, \epsilon) \subseteq S$. The *closure* of S, written $\operatorname{cl} S$, is the set of $x \in \mathbb{R}^d$ such that, for all $\epsilon > 0$, $B(x, \epsilon) \cap S$ is nonempty. The *boundary* of S, written $\operatorname{bd} S$, is the set of $x \in \mathbb{R}^d$ such that, for all $\epsilon > 0$, both $S \cap B(x, \epsilon)$ and $S \setminus B(x, \epsilon)$ are nonempty.

Definition 1.1.2. Let $S \subseteq \mathbb{R}^d$. S is said to be *convex* if, for every $x \in S$ and $y \in S$ and $t \in [0, 1]$, the point $x + t(y - x)$ lies in S.

Definition 1.1.3. Let $S \subseteq \mathbb{R}^d$. S is said to be a *cone* if, for every $x \in S$ and $t > 0$, the point tx lies in S.

Definition 1.1.4. Let $S \subseteq \mathbb{R}^d$. S is said to be a *convex cone* if, for every $x \in S$, $y \in S$, $s \in [0, 1]$, and $t \in [0, 1]$, the point $sx + ty$ lies in S.

Theorem 1.1.5. *Let $S \subseteq \mathbb{R}^d$. S is a convex cone if and only if (S is convex and S is a cone).*

Definition 1.1.6. Let $K \subseteq \mathbb{R}^d$ be a convex cone. K is said to be pointed if there is no $v \in \mathbb{R}^d \setminus \{0\}$ such that $v \in K$ and $(-v) \in K$.

Definition 1.1.7. Let $S \subseteq \mathbb{R}^d$. The *convex hull* of S, written $\operatorname{conv} S$, is defined by

$$\operatorname{conv} S := \left\{ \lambda_1 x_1 + \cdots + \lambda_k x_k : \begin{array}{l} k > 0, \\ \sum_{i=1}^{k} \lambda_i = 1, \\ \lambda_1 \geq 0 \ldots \lambda_k \geq 0, \\ x_1 \in S, \ldots, x_k \in S \end{array} \right\}.$$

A theorem of Carathéodory states that one can limit $k \leq d + 1$ in the above definition.

Theorem 1.1.8. *Let $S \subseteq \mathbb{R}^d$. Then* $\operatorname{conv} S$ *is convex and*

$$\operatorname{conv} S = \bigcap_{T \supseteq S, T \, convex} T.$$

Definition 1.1.9. Let S be a convex subset of $\mathbb{R}^d$. The *polar* of S is written S° and is defined by

$$S^\circ := \{h \in (\mathbb{R}^d)^* : \forall x \in S, hx \leq 1\}.$$

If S is a convex cone, its *dual cone* is written S^* and is defined by

$$S^* := \{h \in (\mathbb{R}^d)^* : \forall x \in S, hx \geq 0\}.$$

The choice of $\geq$ or $\leq$ in the definition of the dual cone is a matter of personal taste. Using $\leq$ appears to be more common in the literature. However, this thesis contains a chapter devoted to linear optimisation, where the cone in question is the nonnegative orthant and its dual is pervasive. That chapter would contain significantly more errors had I followed the more common $\leq$ convention.

Theorem 1.1.10. *Let S be a convex subset of $\mathbb{R}^d$. Then S° is convex. Further,*

- *If $0 \in \operatorname{int} S$, then S° is compact.*
- *If S is a cone, then S° is a cone and $S^* = -S^\circ$.*

Theorem 1.1.11. *If $A \subseteq B \subseteq \mathbb{R}^d$, then $A^\circ \supseteq B^\circ$.*

Theorem 1.1.12. *Let S be a closed convex subset of $\mathbb{R}^d$. Then $S = (S^\circ)^\circ$. If S is a closed convex cone, then $(S^*)^* = S$.*

Theorem 1.1.13. *Let A and B be compact convex subsets of $\mathbb{R}^d$, each containing the origin in its interior. Then*

$$A \cap B = \operatorname{conv}(A^\circ \cup B^\circ)^\circ.$$

Proof. Note that $\operatorname{conv}(A^\circ \cup B^\circ) \supseteq A^\circ$, so $\operatorname{conv}(A^\circ \cup B^\circ)^\circ \subseteq (A^\circ)^\circ = A$. Similarly, $\operatorname{conv}(A^\circ \cup B^\circ)^\circ \subseteq B$. This establishes the $\supseteq$ containment.

Now suppose $x \in A \cap B$. Let $v^T \in \operatorname{conv}(A^\circ \cup B^\circ)$. Write $v^T = \lambda a^T + (1-\lambda)b^T$ for $\lambda \in [0,1]$, $a^T \in A^\circ$, and $b^T \in B^\circ$. Then $v^T x = \lambda a^T x + (1-\lambda) b^T x \leq 1$, so $x \in \operatorname{conv}(A^\circ \cup B^\circ)^\circ$. This establishes the $\subseteq$ containment; together with the last paragraph, the desired equality has been established. □

One can take the polar of both sides of the equality in the above theorem. One can interpret the polar A° as a set of certificates of nonmembership in A, and similarly B°. The theorem then states that one can always certify that a point is outside $A \cap B$ by taking a convex combination of a certificate that some possibly different point is outside A with a certificate that yet another possibly different point is outside B.

The hypothesis that A and B's relative interiors intersect above is called "Slater's condition" in many contexts.

One particularly important example of a convex cone is the cone of $n \times n$ symmetric, positive definite real matrices. This shall be denoted $\mathbb{S}^n_{++}$.

1.2 Optimisation over convex sets

It is sometimes interesting to compute or estimate the minimum value some function takes on some set. Occasionally, problems of this sort can be naturally formulated, or at least approximated well, by problems of minimising some linear functional over some finite-dimensional convex set.

Given a linear functional $c^T \neq 0$ on $\mathbb{R}^d$ and a convex set S that is for now nonempty, compact, and having the origin in its interior, we can formulate the problem

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & x \in S. \end{aligned} \tag{1.1}$$

There exists an optimal solution x^* to the above problem that lies in $\operatorname{bd} S$. At this solution, we have $c^T x^* \leq c^T x$ for every x in S and $c^T x^* < 0$. Then $c^T / c^T x^*$ lies in the polar S° of S.

Indeed, if $\lambda \geq 0$ is such that $c^T \in \lambda S^\circ$, then $-\lambda$ is a valid lower bound on the optimal objective value of (1.1). One can formulate the *dual problem*

$$\begin{aligned} \max \quad & -\lambda \\ \text{subject to} \quad & c^T \in \lambda S^\circ \\ & \lambda \geq 0 \end{aligned} \tag{1.2}$$

of finding the best such lower bound on the optimal objective value of (1.1).

Note that every feasible solution to (1.2) is a valid lower bound on the optimal objective value of (1.1), and, conversely, every feasible solution to (1.1) is a valid lower bound on the optimal objective value of (1.2). This is called *weak duality*.

Since $\lambda^* = -c^T x^*$ is a feasible solution to this dual problem when x^* is an optimal solution to (1.1) and λ^* has the same objective value in (1.2) as x^* has in (1.1), (1.2) is capable of proving a tight lower bound on the optimal objective value of (1.1) and vice-versa. This is called *strong duality*. (In the literature, strong duality usually refers only to the ability of some dual feasible solution to prove a tight bound on the primal objective. However, there is no occasion in this thesis in which the weaker assumptions of these traditional strong duality theorems lead to more interesting results.)

Remark 1.2.1. Simple geometric considerations might convince somebody that there is nothing to be done beyond the above duality relationship. After all, if S does not have full dimension, one can restrict to its affine hull; if S does not have the origin in its interior, one can translate S until it does; and if S is, say, closed but not compact, either $c^T x$ is not bounded below on S or one can intersect S with a sufficiently large ball that the optimal objective value is not disturbed but S is made compact.

This is all certainly true. However, the job of an optimisation algorithm when presented with (1.1) is to discover one fact about the geometry of S—how far in the direction c^T can

x go without leaving S. It cannot necessarily rely on having deep knowledge, such as an explicit description of S°, about the structure of S a priori.

Indeed, interesting pathologies can arise if one tries to apply Theorem 1.1.13 in situations where its hypotheses fail. If S is the intersection of a disc centred at the origin with a line, then either the line misses the disc, or the line intersects the interior of the disc, or the line is tangent to the disc. In one case, S is empty; in another, S° is simply the Minkowski sum of the polars of the line and the disc; and in the third, S° is strictly larger than said Minkowski sum.

Another objection that one might have to studying these pathologies is that they are simply degenerate cases that occur "with probability zero," and one can always apply a small, random perturbation to erase the pathology "with probability one." Unfortunately, in computation, inputs that are nondegenerate but are close to being degenerate can often cause problems. Further, many applications of convex optimisation do not choose problems randomly at all, and degenerate problems do arise.

This motivates the study of duality theorems that work in terms of an "algebraic" description of S, say as the intersection of several "well-understood" convex sets, that give sufficient conditions for strong duality to hold. This is actually still an active area of research.

In this thesis, the required strong duality theorem, Theorem 1.2.5, covers the minimisation of smooth, convex, $(\mathbb{R} \cup \{\infty\})$-valued functions over an affine subspace of $\mathbb{R}^d$. The two corollaries, Corollary 1.2.6 and Corollary 1.2.7, offer specialisations to the two cases that arise in this thesis.

Definition 1.2.2. Let $F : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$. F is *convex* if, for every x and y in $\mathbb{R}^d$ and $\lambda \in (0, 1)$,

$$F(\lambda x + (1 - \lambda)y) \leq \lambda F(x) + (1 - \lambda)F(y).$$

(Here, $\lambda F(x)$ is taken to be ∞ if $F(x) = \infty$.)

Definition 1.2.3. Let $F : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$ be convex. The *modified Legendre-Fenchel conjugate* of F is

$$F_*(s) := -\inf_{x \in \mathbb{R}^d} \left\{ s^T x + F(x) \right\}.$$

F_* is always a convex function.

Proposition 1.2.4. *Let $\mu > 0$. Let $F : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$ be convex. Then*

$$F_*(s/\mu) = \frac{1}{\mu} \inf_{x \in \mathbb{R}^d} \left\{ s^T x + \mu F(x) \right\}. = \frac{1}{\mu}(\mu F)_*(s).$$

The following result essentially appears as Corollary 3.3.11 in [5].

Theorem 1.2.5. *Let $F : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$ be convex. Let $A : \mathbb{R}^d \to \mathbb{R}^m$ be a linear map, and let $b \in \mathbb{R}^m$. Then*

$$\inf_{x \in \mathbb{R}^d} \{F(x) : Ax = b\} \geq \sup_{y \in \mathbb{R}^m} \left\{b^T y - F_*(-A^T y)\right\}.$$

Furthermore, if $b \in \operatorname{int}\{Ax : F(x) < \infty\}$, then equality holds and the supremum is attained.

Corollary 1.2.6. *Let $F : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$ be convex. Let $A : \mathbb{R}^d \to \mathbb{R}^m$ be a surjective linear map, let $b \in \mathbb{R}^m$, and let $c^T \in (\mathbb{R}^d)^*$. Suppose there are $x^\circ \in \mathbb{R}^d$ and $s^\circ \in (\mathbb{R}^d)^*$ and a positive real ϵ such that F is real-valued on $B(x^\circ, \epsilon)$ and F_* is real-valued on $B(s^\circ, \epsilon)$. Then*

$$\inf_{x \in \mathbb{R}^d} \left\{c^T x + F(x) : Ax = b\right\} = \sup_{y \in \mathbb{R}^m, s \in (\mathbb{R}^d)^*} \left\{b^T y - F_*(s) : A^T y + s = c\right\}.$$

Furthermore, both the infimum and supremum above are attained.

Proof. Define $G(x) = c^T x + F(x)$. Then $G_*(z) = F_*(c + z)$. Writing $s = c - A^T y$, the result, apart from attainment of the infimum, immediately follows from Theorem 1.2.5.

To see that the infimum is also attained, interchange primal and dual. The remainder of this proof does so explicitly. Let B be a linear map such that B has $d - m$ rows and full row rank, but $AB^T = BA^T = 0$. (One example of such a thing is orthogonal projection onto the orthogonal complement of the row space of A.) Then $x \in \ker A$ if and only if $x \in \operatorname{im} B^T$, and $s \in \ker B$ if and only if $s \in \operatorname{im} A^T$. Let d be such that $Ad = b$. Then

$$\begin{aligned} & \sup_{y \in \mathbb{R}^m, s \in (\mathbb{R}^d)^*} \left\{d^T A^T y - F_*(s) : A^T y + s = c\right\} \\ = \; & d^T c - \inf_{s \in (\mathbb{R}^d)^*} \left\{d^T s + F_*(s) : Bs = Bc\right\}. \end{aligned}$$

This is equal to—and attainment occurs by the above considerations so I shall write max—

$$d^T c - \max_{z \in \mathbb{R}^{n-m}, x \in \mathbb{R}^d} \left\{c^T B^T z - F(x) : B^T z + x = d\right\}.$$

By simple algebra,

$$\begin{aligned} & d^T c - \max_{z \in \mathbb{R}^{n-m}, x \in \mathbb{R}^d} \left\{c^T B^T z - F(x) : B^T z + x = d\right\} \\ = \; & d^T c - \max_{x \in \mathbb{R}^d} \left\{c^T d - c^T x - F(x) : B^T z + x = d\right\} \\ = \; & \min_{x \in \mathbb{R}^d} \left\{c^T x + F(x) : Ax = b\right\}. \end{aligned}$$

This shows that the infimum is also attained. □

Corollary 1.2.7. *Let K be a closed convex cone. Let $A : \mathbb{R}^d \to \mathbb{R}^m$ be a surjective linear map and let $b \in \mathbb{R}^m$ be such that there is a point $x^\circ \in \operatorname{int} K$ such that $Ax^\circ = b$. Let $c^T \in (\mathbb{R}^d)^*$ be such that there is a point $(y^\circ, s^\circ) \in \mathbb{R}^m \times \operatorname{int} K^*$ such that $A^T y^\circ + s^\circ = c$. Then*

$$\inf_{x \in \mathbb{R}^d} \left\{ c^T x : Ax = b, x \in K \right\} = \sup_{y \in \mathbb{R}^m, s \in (\mathbb{R}^d)^*} \left\{ b^T y : A^T y + s = c, s \in K^* \right\},$$

and both the infimum and supremum are attained.

Proof. Write $F(x)$ for the function that is 0 on $\operatorname{int} K$ and ∞ outside $\operatorname{int} K$. Then F is convex. Note that

$$F_*(s) = -\inf_{x \in K} s^T x.$$

This is finite—zero—if and only if $s \in K^*$. The result immediately follows from Corollary 1.2.6. □

Remark 1.2.8. An optimisation problem of the form

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \in K \end{aligned}$$

for a convex cone K and surjective A may turn out to be infeasible—there may be no x such that $x \in K$ and $Ax = b$. In this case, AK and $\{b\}$ are disjoint nonempty convex sets; by a separating hyperplane theorem for convex sets, there must be a $y \neq 0$ such that $(AK)^T y \leq 0$ and $b^T y \geq 0$. The first condition can be rephrased as: there exists an $s \in K^*$ such that $A^T y + s = 0$. If $\{b\}$ does not lie in the closure of AK, then such a y exists such that $b^T y > 0$, strictly separating AK from b. Such a y constitutes a proof of infeasibility. (Observe that "failure" here occurs when the closure of AK contains b but AK itself does not.) Thus one can sometimes prove an optimisation problem in conic form infeasible by finding a recession direction of its homogenised dual.

Furthermore, if one is trying to find a ray in $K^* \cap \operatorname{im} A^T$ as above, one can equivalently look for a ray in $K^* \cap (s^\circ + \operatorname{im} A^T)$ for some $s^\circ \in \operatorname{int} K^*$. This shifted problem may be better-behaved since it has an interior feasible solution.

1.3 Properties of self-concordant barriers

The interior-point methods discussed in this thesis reformulate the conic constraint $x \in K$ using a convex "barrier function" that tends to ∞ as one approaches the boundary of K. This allows the algorithm to ignore, in some sense, the constraint $x \in K$, and use ideas from unconstrained optimisation within the affine space $\{x : Ax = b\}$. One such idea is Newton's method; indeed, the steps taken by some of the methods in this thesis can be

interpreted as damped Newton steps. In order to get a good worst-case bound on the rate of convergence of the method, a bound on the change in the Hessian while taking a step is necessary. This is the role of self-concordance. Self-concordance implies, in particular, that, for every $x \in \operatorname{int} K$, the Hessian defines a norm near x whose open unit ball centred at x lies inside $\operatorname{int} K$. Within that ball, useful, nontrivial bounds on the Hessian hold.

Without exception, the results in this section can be found in the book by Nesterov and Nemirovskii [58].

Definition 1.3.1. Let K be a closed convex cone in $\mathbb{R}^d$. Let $F : \operatorname{int} K \to \mathbb{R}$ satisfy the following properties:

- (*θ-logarithmic homogeneity*) There exists a $\theta \geq 1$ such that, for every $t > 0$ and every $x \in \operatorname{int} K$, $F(tx) = -\theta \log t + F(x)$.
- (*Self-concordance*) F is $\mathcal{C}^3$ and convex and, for every $x \in \operatorname{int} K$ and $h \in \mathbb{R}^d$,
$$|F'''(x)[h,h,h]| \leq 2 \left(F''(x)[h,h]\right)^{3/2}.$$
- (*Barrier*) If $\{x_k\}_{k=1}^{\infty}$ is a sequence drawn from $\operatorname{int} K$ such that $\lim_{k\to\infty} x_k$ exists and is a point of $\operatorname{bd} K$, then $\lim_{k\to\infty} F(x_k) = \infty$.

I call F a *θ-logarithmically homogeneous self-concordant barrier* (or *θ-LHSCB*) for K.

Example 1.3.2. The simplest example of a logarithmically homogeneous self-concordant barrier is the function $x \mapsto -\log x$. This is a 1-LHSCB for the closed convex cone $\mathbb{R}_+$.

Another important example is the 2-LHSCB
$$(x,t) \mapsto -\log\left(t^2 - ||x||_2^2\right)$$
for the second-order cone
$$\{(x,t) \in \mathbb{R}^d \times \mathbb{R} : t \geq ||x||_2\}.$$

Note that, if F_1 is a θ_1-LHSCB for K_1 and F_2 is a θ_2-LHSCB for K_2, then the direct sum $F_1 \oplus F_2$ is a $(\theta_1 + \theta_2)$-LHSCB for the direct sum $K_1 \oplus K_2$.

Self-concordance may appear mysterious at first. The purpose of this condition is to permit bounding higher-order derivatives in terms of the second derivative. Whereas Taylor series-type arguments for general $\mathcal{C}^k$ functions "work" only within some region whose size is not necessarily known a priori, self-concordance allows rigorous statements about function behaviour within constant-sized ellipsoids, where the quadratic form defining the ellipsoid is the Hessian F'' of the self-concordant barrier F.

Indeed, it is useful to think of F giving $\operatorname{int} K$ the structure of a Riemannian manifold with the metric at a point x defined by $F''(x)$:

Definition 1.3.3. Let K be a closed convex cone and let F be a θ-LHSCB for K. Let $x \in \operatorname{int} K$ and define the *local metric* at x by

$$||h||_{x,F} := \sqrt{F''(x)[h,h]}.$$

(I will typically write $||h||_x$ when the barrier F is understood from context.)

Self-concordance then gives a sort of local "compatibility" between the local metric and the usual Euclidean metric:

Theorem 1.3.4. *Let K be a closed convex cone and let F be a θ-LHSCB for K. Let $x \in \operatorname{int} K$ and $h_1, h_2, h_3 \in \mathbb{R}^d$. Then*

$$|F'''(x)[h_1,h_2,h_3]| \leq 2\,||h_1||_x\,||h_2||_x\,||h_3||_x\,.$$

Theorem 1.3.5. *Let K be a closed convex cone and let F be a θ-LHSCB for K. Let $x \in \operatorname{int} K$ and let $\sigma \in \mathbb{R}_+$ and $s \in \mathbb{R}^d$ be such that $\sigma^2 = F''(x)[s,s] < 1$. Let $h \in \mathbb{R}^d$. Then*

$$(1-\sigma)^2 F''(x+s)[h,h] \leq F''(x)[h,h] \leq \frac{1}{(1-\sigma)^2} F''(x+s)[h,h].$$

Theorem 1.3.6. *Let K be a convex cone and let $F : \operatorname{int} K \to \mathbb{R}$ be θ-logarithmically homogeneous. Let $x \in \operatorname{int} K$. Then*

- $F^{(k)}(tx) = t^{-k} F^{(k)}(x)$ *whenever* $t > 0$.
- $F'(x)[x] = -\theta$.
- $F''(x)[x] = -F'(x)$.
- $F''(x)[x,x] = \theta$.
- $F'''(x)[x] = -2F''(x)$.

Proof. Compute, using the definition of the derivative,

- $F^{(k)}(tx) = \lim_{h\to 0} \frac{\sum_{j=0}^k (-1)^{k-j}\binom{k}{j} F(tx+jh)}{h^k}$
  $= \lim_{h\to 0} \frac{\sum_{j=0}^k (-1)^{k-j}\binom{k}{j} (F(x+jh/t)+\theta\log t)}{h^k}$
  $= \lim_{h\to 0} \frac{\sum_{j=0}^k (-1)^{k-j}\binom{k}{j} F(x+jh/t)}{t^k (h/t)^k} = \frac{1}{t^k} F^{(k)}(x).$
- $F'(x)[x] = \lim_{h\to 0} \frac{F(x+hx)-F(x)}{h}$
  $= \lim_{h\to 0} \frac{-\theta\log(1+h)}{h} = -\theta.$
- $F''(x)[x] = \lim_{h\to 0} \frac{F'(x+hx)-F'(x)}{h}$
  $= \lim_{h\to 0} \left(\frac{1}{1+h} - 1\right) F'(x)/h = -F'(x).$
- $F''(x)[x,x] = -F'(x)[x] = \theta.$
- $F'''(x)[x] = \lim_{h\to 0} \frac{F''(x+hx)-F''(x)}{h}$
  $= \lim_{h\to 0} \left(\frac{1}{(1+h)^2} - 1\right) F''(x)/h = -2F''(x).$

□

1.4 Stronger regularity properties

Some barriers enjoy stronger Hessian estimation properties than self-concordance. A bound like Theorem 1.3.5 that holds in a larger region than $\{h : ||h||_x < 1\}$ can mean that an algorithm is able to take longer steps and, hopefully, converge faster in practice.

A useful norm to use in this context is the *norm induced by K at x*:

$$||h||_{K,x} = \min\{|1/t| : x + th \in K \text{ and } x - th \in K\}.$$

Nesterov and Nemirovskii [58] defined *α-regularity*; a barrier F is said to be α-regular if, for every $x \in \operatorname{int} K$ and every $h \in \mathbb{R}^d$,

$$\left|F^{(4)}(x)[h,h,h,h]\right| \leq \alpha(\alpha+1)\,||h||_{K,x}^2\,||h||_x^2\,.$$

Some barriers, such as the self-scaled barriers later in this chapter and the hyperbolic barriers of Chapter 3 satisfy a stronger property called the *long-step Hessian estimation property*. F satisfies the long-step Hessian estimation property if, for every $y \in K$, $F'(x)[y]$ is a convex function of its argument $x \in \operatorname{int} K$. This implies that, for every $x \in \operatorname{int} K$, every $z \in \mathbb{R}^d$ such that $x \pm z \in \operatorname{int} K$, and every $h \in \mathbb{R}^d$,

$$(1 - ||z||_{K,x})^2 F''(x+z)[h,h] \leq F''(x)[h,h] \leq \frac{1}{(1-||z||_{K,x})^2} F''(x+z)[h,h].$$

1.5 Barrier duality

Recall Definition 1.2.3, which defines the Legendre-Fenchel conjugate of a convex function. This section examines the connection between convex cones K with θ-LHSCBs F on one hand and the dual cone K^* with the Legendre-Fenchel conjugate F_* on the other. Notably, F_* is a θ-LHSCB and the nonlinear map $x \mapsto F'(x)$ is a bijection from $\operatorname{int} K$ to $\operatorname{int} K^*$.

The results in this section can also be found in Nesterov and Nemirovskii's book [58].

Theorem 1.5.1. *Let K be a pointed closed convex cone and let F be a θ-LHSCB for K. Let $x \in \operatorname{int} K$ and $s \in \operatorname{int} K^*$. Then*

- *$F_*(-F'(x)) = -\theta - F(x)$ and $F(-F_*'(s)) = -\theta - F_*(s)$. (In particular, $-F'(x) \in \operatorname{int} K^*$ and $-F_*'(s) \in \operatorname{int} K$.)*
- *$-F_*'(-F'(x)) = x$.*

- $-F'(-F'_*(s)) = s$.
- $F''_*(-F'(x)) = (F''(x))^{-1}$.
- $F''(-F'_*(s)) = (F''_*(s))^{-1}$.

Proof. Note that

$$F_*(-F'(x)) = -\inf_z -F'(x)[z] + F(z).$$

Writing ∇_z for the derivative with respect to z, the infimum is attained at the point where

$$\nabla_z \left(-F'(x)[z] + F(z)\right) = 0.$$

This occurs where $-F'(x) + F'(z) = 0$; since F is strictly convex, this implies that $z = x$. Thus

$$F_*(-F'(x)) = F'(x)[x] - F(x) = -\theta - F(x),$$

as advertised.

One can then differentiate both sides of $F_*(-F'(x)) = -\theta - F(x)$ using the chain rule on the left-hand side to obtain $-F'_*(-F'(x))F''(x) = -F'(x)$; multiplying both sides by $(F''(x))^{-1}$ yields $-F'_*(-F'(x)) = (F''(x))^{-1}[-F'(x)] = x$.

One can differentiate again to find that

$$F''_*(-F'(x))F''(x) = I$$

from which it follows that $F''_*(-F'(x)) = (F''(x))^{-1}$.

One can arrive at the other three properties using the Legendre-Fenchel conjugate. □

The Legendre-Fenchel conjugate gives a natural barrier for the dual cone:

Theorem 1.5.2. *Let K be a pointed closed convex cone and let F be a θ-LHSCB for K. Then F_* is a θ-LHSCB for K^*.*

The dual barrier defines a local norm everywhere on the dual cone. Its inverse defines a local norm everywhere on the primal cone by

$$||h||_s^* = \sqrt{(F''_*(s))^{-1}[h,h]}.$$

This norm is dual to the s-norm in the sense that

$$||h||_s^* = \sup_{v^T : ||v^T||_s \leq 1} v^T h.$$

Because $(F''_*(s))^{-1} = F''(-F'_*(s))$, one can write $||\cdot||_s^* = ||\cdot||_{-F'_*(s)}$.

1.6 Self-scaled barriers

There are barriers F for which the long-step Hessian estimation property mentioned in Section 1.4 holds but the same property does not hold for F_*. There are also barriers where both F and F_* satisfy the long-step Hessian estimation property:

Definition 1.6.1 (Nesterov and Todd [59])**.** A θ-LHSCB F is called a *self-scaled barrier* if, for every $x \in \operatorname{int} K$ and $w \in \operatorname{int} K$, both $F''(w)x \in \operatorname{int} K^*$ and $F_*(F''(w)x) = F(x) - 2F(w) - \theta$.

Theorem 1.6.2 (Nesterov and Todd [59])**.** *Let K be a pointed closed convex cone and let F be a self-scaled θ-LHSCB for K. Then F_* is a self-scaled θ-LHSCB for K^*.*

Self-scaled barriers only exist for symmetric cones—cones K such that K is linearly isomorphic to K^* and such that the group $\operatorname{Aut}(K)$ of linear automorphisms of K acts transitively on the points of $\operatorname{int} K$. This result was first stated by Güler [28].

Symmetric cones have a rich theory owing to their connection to formally real Euclidean Jordan algebras. Every symmetric cone arises as a direct sum of cones of positive elements of finite-dimensional formally real Euclidean Jordan algebras, a result of Koecher [41] (1957) and Vinberg [91] (1961). The finite-dimensional formally real Euclidean Jordan algebras were completely classified by Jordan, von Neumann, and Wigner in 1933 [38]. Faraut and Korányi [18] give a more recent treatment of symmetric cones and Euclidean Jordan algebras.

The characterisation of symmetric cones is classical, but it does not directly imply that all self-scaled barriers arise in the same way. This was first proven by Schmieta [79]. Independently, Hauser and Güler [33] showed that every self-scaled barrier decomposes as a direct sum of self-scaled barriers for irreducible symmetric cones, and Hauser and Lim [34] classified the self-scaled barriers for irreducible symmetric cones.

A key theorem regarding self-scaled barriers is that, if x and x' lie in $\operatorname{int} K$, there is a unique $w \in \operatorname{int} K$ such that $F''(w)x = x'$. Furthermore, this w can be computed as the midpoint of the geodesic between x and x' if $\operatorname{int} K$ is given the Riemannian metric induced by F''. Still further, in the self-scaled case, $F''(w)$ is the midpoint of the geodesic between $F''(x)$ and $(F''(x'))^{-1}$. See Nesterov and Todd [59, 62, 63].

Some practically successful interior-point methods for second-order cone and semidefinite optimisation such as SDPT3 [87] and SeDuMi [83] rely heavily on the rich theory of self-scaled barriers.

Chapter 2

Primal-dual interior-point methods for convex conic optimisation

For expository purposes, this chapter begins in the concrete setting of linear optimisation. Section 2.1 gives an overview of the structure of interior-point methods. The discussion is informal; many theorems are stated ambiguously in this overview for which clear statements and proofs in a more general setting are given in later sections. Section 2.2 gives an explicit example of an interior-point method for linear optimisation due to Mizuno, Todd, and Ye [52].

Section 2.4 defines classes of linear maps called *primal-dual scalings* that exist for all convex optimisation problems. Section 2.9 gives a generalisation of the Mizuno-Todd-Ye algorithm to convex optimisation based on primal-dual scalings.

This chapter works with algorithms in a model of computation in which real numbers can be stored exactly and real-number arithmetic has unit cost. This would be an unreasonable choice for a general-purpose model of computation, but it allows the presentation to focus on the mathematical structure of the algorithms and what, at a high level, makes them work. Later chapters will focus more on bridging the gap between the theory presented here and an actual implementation of a practical interior-point method.

2.1 Overview of interior-point methods for linear optimisation

Consider solving the linear optimisation problem

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.1}$$

and its dual

$$\begin{aligned} \max \quad & b^T y \\ \text{subject to} \quad & A^T y + s = c \\ & s \geq 0. \end{aligned} \tag{2.2}$$

Here, c^T is a linear functional on $\mathbb{R}^\theta$, b is a vector, x is a vector of θ variables, A is some surjective linear map, and $x \geq 0$ means that every component of x is nonnegative.

The dimension of the space of variables is called θ here because it coincides with the barrier parameter of the barrier $-\sum_i \log x_i$ for the nonnegative orthant $\{x \in \mathbb{R}^\theta : x \geq 0\}$. The barrier parameter does not always coincide with the dimension of the space of variables, however. Indeed, the barrier parameter need not always be an integer.

Assume that solutions x to (2.1) and (y, s) to (2.2) such that $x_i s_i = x_1 s_1$ for every i are known. This turns out not to be very restrictive; Section 5.7 gives constructions that transform an arbitrary linear optimisation problem into one with this property.

An interior-point method begins with some (possibly very bad) *iterate*—a choice of x, y, and s. It proceeds with a sequence of *iterations* aimed at bringing the iterate somehow closer to solving the problem. Each iteration first computes a *search direction* (d_x, d_y, d_s), then it computes a step length α, then it replaces the iterate (x, y, s) with $(x+\alpha d_x, y+\alpha d_y, s+\alpha d_s)$.[1]

The primal-dual interior-point methods discussed in this chapter compute approximate solutions to the linear optimisation problems (2.1) and (2.2). By "approximate solution," I mean that the constraints are satisfied exactly, but $c^T x$ is not quite as small as possible and $b^T y$ is not quite as large as possible. We will be able to bound the distance (in objective value) to optimality.

The *weak duality* relation $b^T y \leq c^T x$ holds for every feasible solution x to the primal and every feasible solution y to the dual. Thus, if the duality gap $c^T x - b^T y$ is small, then x is a good primal solution *and* y is a good dual solution. Note that $b^T y = x^T A^T y = x^T (c - s)$; one can also write the duality gap as $x^T s$.

[1] This is a little bit of a white lie. All of the interior-point methods in this chapter work this way, but practical interior-point methods can take different-length steps in the primal and dual spaces.

The interior-point methods discussed here work by reformulating the primal and dual problems using barrier functions that tend to ∞ when some component of x or s gets small but which otherwise act tame. (These barriers are θ-logarithmically-homogeneous self-concordant barriers as defined in Chapter 1.) Notionally, they approximately solve a sequence of problems of the form

$$\begin{array}{rl} \min & c^T x - \mu \sum_{i=1}^{\theta} \log x_i \\ \text{subject to} & Ax = b \end{array} \tag{2.3}$$

for various positive choices of the scalar parameter μ that tend to zero. The choice of μ at each iteration is not necessarily explicit; some algorithms find a direction that reduces μ and move as far as they choose in that direction.

It happens that the dual to this problem is

$$\begin{array}{rl} \max & -\theta\mu \log \mu + b^T y + \mu \sum_{i=1}^{\theta} \log s_i \\ \text{subject to} & A^T y + s = c. \end{array} \tag{2.4}$$

(Note that $-\theta\mu \log \mu$ is a constant. I will henceforth ignore it.)

For every $\mu > 0$, there exists a unique optimal solution to both (2.3) and (2.4). These optimal solutions vary continuously with μ; the curve of optimal solutions to (2.3) and (2.4) together, $\{(x(\mu), y(\mu), s(\mu)) : \mu > 0\}$, is called the *central path*. The central path can also be characterised using the first-order optimality conditions for (2.3) and (2.4): The point $(x(\mu), y(\mu), s(\mu))$ is the unique (up to a translation of $y(\mu)$ by a vector in the kernel of A^T) point satisfying $Ax(\mu) = b$, $A^T y(\mu) + s(\mu) = c$, and the condition $x(\mu)_i s(\mu)_i = \mu$ for every i.

Given an x and an s, we can speak of "the μ of" x and s; define $\mu(x, s) := \langle x, s \rangle / \theta$. Some algorithms, such as the Mizuno-Todd-Ye method to be described shortly, do not keep track of a μ explicitly; instead, they keep track of x and s and, whenever a barrier coefficient μ arises, they use $\mu(x, s)$.

It is "good" for the iterate to be "near" the central path; this means that x is "close" to an optimal solution to (2.3) and (y, s) is "close" to an optimal solution to (2.4). Broadly, one can show that it is always possible to make "satisfactory" progress reducing μ "near" the central path. Interior-point methods with theoretical performance guarantees thus have a goal and an obligation—reduce μ, but stay near the central path.

There are a lot of different choices of meanings for "near" and "satisfactory" here that lead to relatively appealing worst-case complexity bounds. Some choices yield quite a lot of leeway for tricks and heuristics without disturbing worst-case complexity, while other choices offer essentially none.[2]

[2]Some practical algorithms do away with worst-case complexity bounds entirely, and even global convergence guarantees. C. Cartis proved that a variant of a very popular algorithm of S. Mehrotra [49] fails to converge on some examples [11]. This popular algorithm remains in widespread use since these convergence failures do not seem to occur on instances of practical interest.

A classical short-step path-following method such as Renegar's 1988 method [71] balances these goals in a simple way; one can show that, "close" to the central path, a factor-$(1 - \Omega(1/\sqrt{\theta}))$ reduction in μ can be attained by a single Newton step and that that Newton step leaves you "close" to the central path. The principal disadvantage of this method is that it only rarely can outperform its worst-case bound substantially.

The Mizuno-Todd-Ye algorithm described in the next section alternates between two kinds of steps, called "predictor" and "corrector" steps. Predictor steps start from "very close" to the central path, reduce μ "satisfactorily", and leave the iterate merely "close" to the central path. Corrector steps take an iterate that is "close" to the central path, leave μ unchanged, and find a new iterate that is "very close" to the central path. One can prove that at least a factor-$(1 - 1/(4\sqrt{\theta}))$ reduction in μ occurs at every predictor step—but, in practice, it is often possible to reduce μ much farther without breaking the algorithm's invariant.

Potential-reduction algorithms balance the goals of reducing μ and maintaining centrality using a $\mathcal{C}^3$ "potential function" to measure progress toward optimality and moving in the direction of local fastest improvement in potential. For many potential-reduction methods, one can show that the potential function's value is reduced by at least an additive constant at each step, and therefore *either* the iteration has reduced μ by a satisfactory amount *or* the new iterate is considerably closer to the central path, or possibly both.

2.2 The Mizuno-Todd-Ye algorithm for linear optimisation

If v is a vector, I will write V for the diagonal matrix whose (i, i) entry is v_i. (That is, a vector's name written in uppercase denotes the diagonal matrix whose diagonal is made of the vector's entries.) The vector e is the vector of the appropriate dimension whose entries are all 1; its dimension will be clear from context.

In particular, $V^{-1}e$ is the vector whose ith entry is $1/v_i$. Note that, if F is the function on $\mathbb{R}^{\theta}_{++}$ given by $F(x) = \sum_i -\log x_i$, then $-F'(x) = X^{-1}e$ and $F''(x) = X^{-2}$.

Consider solving a linear optimisation problem of the form

$$\begin{array}{lrcl} \min & c^T x & & \\ & Ax & = & b \\ & x & \geq & 0 \end{array}$$

and its dual

$$\begin{array}{lrcl} \max & b^T y & & \\ & A^T y \; + s & = & c \\ & s & \geq & 0 \end{array}$$

starting from an initial central feasible solution (x^0, y^0, s^0).

The Mizuno-Todd-Ye algorithm alternates steps that reduce μ with steps that restore centrality. The steps reducing μ ("predictor steps") are made by applying Newton's method to the nonlinear system of equations

$$\begin{aligned} Ax &= 0 \\ A^T y + s &= 0 \\ Xs &= 0, \end{aligned}$$

and the steps restoring centrality ("corrector steps") are made by applying Newton's method to the nonlinear system

$$\begin{aligned} Ax &= 0 \\ A^T y + s &= 0 \\ Xs &= \mu(x,s)e. \end{aligned}$$

If the current iterate is (x, y, s) and the current iterate already satisfies the first two equations, the step (d_x, d_y, d_s) will satisfy $Ad_x = 0$ and $A^T d_y + d_s = 0$.

The third equation in the case of the predictor step reads

$$(X + D_x)(s + d_s) = 0,$$

which can be rewritten as

$$Xs + Xd_s + Sd_x + D_x d_s = 0.$$

Dropping the quadratic term, this becomes

$$Xd_s + Sd_x = -Xs.$$

Doing the same with the third equation defining the corrector step gives the linearisation [3]

$$Xd_s + Sd_x = -Xs + \mu(x,s)e.$$

The following lemma is key:

Lemma 2.2.1. *If d_x, d_y, and d_s satisfy the linear system*

$$\begin{aligned} Ad_x &= 0 \\ A^T d_y + d_s &= 0 \\ d_x + XS^{-1}d_s &= z \end{aligned}$$

then

$$T^{-1}d_x + Td_s = T^{-1}z,$$

where $T = X^{1/2}S^{-1/2}$.

Furthermore, $T^{-1}d_x$ is orthogonal to Td_s.

[3] In both cases, there is more than one way to write the third equation, and a different choice will lead to a different linearisation. This linearisation remains the same under interchange of primal and dual problems and leads to "primal-dual" algorithms.

Proof. Write the linear system as

$$\begin{array}{rrrcl} & (AT)T^{-1}d_x & & = & 0 \\ (AT)^T d_y & & +Td_s & = & 0 \\ & T^{-1}d_x & +Td_s & = & T^{-1}z. \end{array}$$

The first statement follows immediately.

Note that $T^{-1}d_x \in \ker AT$ while $Td_s \in \mathrm{im}(AT)^T$. It therefore follows that $T^{-1}d_x \perp Td_s$. $\square$

The square root T of T^2 can be thought of as mapping from the dual space to some intermediate "v-space" and also mapping from "v-space" to the primal space.

For positive real β, define the "2-norm β-neighbourhood of the central path":

$$N_2(\beta) := \left\{ (x,s) \in \mathbb{R}^\theta \times \mathbb{R}^\theta : \left|\left| Xs - \frac{x^T s}{\theta} e \right|\right|_2 \leq \beta \frac{x^T s}{\theta} \right\}.$$

The Mizuno-Todd-Ye method alternates predictor and corrector steps. Predictor steps take the iterate from a very small neighbourhood of the central path, namely $N_2(1/12)$, to a larger neighbourhood, namely $N_2(1/4)$. Corrector steps then move the iterate back into the smaller neighbourhood.

The following lemma proves that corrector steps correct—that is that they bring the iterate from $N_2(1/4)$ into a tighter neighbourhood of the central path:

Lemma 2.2.2. *Let $x \in \mathbb{R}^\theta_{++}$, $y \in \mathbb{R}^m$, and $s \in \mathbb{R}^\theta_{++}$ satisfy $Ax = b$ and $A^T y + s = c$. Let $T^2 = XS^{-1}$ Suppose further that $(x,s) \in N_2(1/4)$.*

If d_x, d_y, and d_s satisfy the linear system

$$\begin{array}{rrrcl} & Ad_x & & = & 0 \\ A^T d_y & & +d_s & = & 0 \\ & d_x & +XS^{-1}d_s & = & -x + \mu(x,s)S^{-1}e, \end{array}$$

then

- $A(x + d_x) = b$,
- $A^T(y + d_y) + (s + d_s) = c$,
- $x + d_x > 0$,
- $s + d_s > 0$, *and*
- $(x + d_x, s + d_s) \in N_2(1/12)$.

Proof. Clearly, the two desired equations hold.

Note that

$$(X + D_x)(s + d_s) = Xs + Xd_s + Sd_x + D_x d_s = \mu e + D_x d_s.$$

In particular, $(x + d_x)^T(s + d_s)/\theta = \mu$. Therefore,

$$\left|\left|(X + D_x)(s + d_s) - \frac{(x + d_x)^T(s + d_s)}{\theta} e\right|\right|_2 = ||\mu e + D_x d_s - \mu e||_2 = ||D_x d_s||_2 .$$

By the Cauchy-Schwarz inequality,

$$||D_x d_s||_2 \leq ||D_x d_s||_1 = \langle d_x, d_s \rangle = \left\langle T^{-1} d_x, T d_s \right\rangle \leq \left|\left|T^{-1} d_x\right|\right|_2 ||T d_s||_2 .$$

By Lemma 2.2.1,

$$\begin{aligned} & \left|\left|T^{-1} d_x\right|\right|_2^2 \\ \leq & \left|\left|T^{-1}(-x + \mu S^{-1} e)\right|\right|_2^2 \\ = & \left|\left|(XS)^{-1/2}(Xs - \mu e)\right|\right|_2^2 \\ \leq & \left|\left|(XS)^{-1/2}\right|\right|^2 \frac{\mu^2}{4^2} \\ \leq & \frac{1/\mu}{1 - 1/4} \frac{\mu^2}{16} = \frac{\mu}{12}, \end{aligned}$$

and similarly for $||Td_s||_2^2$. This establishes that, provided $x + d_x > 0$ and $s + d_s > 0$, then $(x + d_x, s + d_s) \in N_2(1/12)$.

It remains to show that $x + d_x > 0$ and $s + d_s > 0$. Since T and T^{-1} both map the positive orthant to itself, it suffices to show that $T^{-1}x + T^{-1}d_x > 0$ and $Ts + Td_s > 0$. Note that $(T^{-1}x)_i = \sqrt{x_i s_i} \geq \sqrt{3\mu/4}$. This is larger than $||T^{-1}d_x||_2$, so no component of $T^{-1}x$ can go negative. An identical argument shows that $s + d_s > 0$. □

The following lemma shows that predictor steps predict, and that the steps one can take in the predictor direction are not unreasonably short: (x,s)

Lemma 2.2.3. *Let $x \in \mathbb{R}^\theta_{++}$, $y \in \mathbb{R}^m$, and $s \in \mathbb{R}^\theta_{++}$ satisfy $Ax = b$ and $A^T y + s = c$. Let $T^2 = XS^{-1}$ and let $\mu = x^T s/\theta$. Suppose further that $(x, s) \in N_2(1/12)$.*

If d_x, d_y, and d_s satisfy the linear system

$$\begin{array}{ccccc} & Ad_x & & = & 0 \\ A^T d_y & & +d_s & = & 0 \\ & d_x & +XS^{-1}d_s & = & -x \end{array}$$

and $\alpha = \frac{1}{\sqrt{6\theta}}$, then

- $A(x + \alpha d_x) = b$,
- $A^T(y + \alpha d_y) + (s + \alpha d_s) = c$,
- $x + \alpha d_x > 0$,
- $s + \alpha d_s > 0$, *and*
- $(x + \alpha d_x, s + \alpha d_s) \in N_2(1/4)$.

Proof. Again, the first two equations clearly hold.

Note that

$$(X + \alpha D_x)(s + \alpha d_s) = Xs + Xd_s + Sd_x + D_x d_s = (1 - \alpha)Xs + \alpha^2 D_x d_s.$$

In particular, $(x + \alpha d_x)^T(s + \alpha d_s) = (1 - \alpha)x^T s$. Thus, by the triangle inequality,

$$||(X + \alpha D_x)(s + \alpha d_s) - (1 - \alpha)\mu e||_2 \leq (1 - \alpha)\,||Xs - \mu e||_2 + \alpha^2\,||D_x d_s||_2\,.$$

By the Cauchy-Schwarz inequality,

$$||D_x d_s||_2 \leq ||D_x d_s||_1 = \langle d_x, d_s \rangle = \left\langle T^{-1}d_x, Td_s \right\rangle \leq \left|\left|T^{-1}d_x\right|\right|_2 ||Td_s||_2\,.$$

By Lemma 2.2.1,

$$\left|\left|T^{-1}d_x\right|\right|_2^2 \leq \left|\left|T^{-1}x\right|\right|_2^2 = \sum_i x_i s_i = x^T s = \theta\mu.$$

Similarly, $||Td_s||_2^2 \leq \theta\mu$. This implies that, provided $x + \alpha d_x > 0$ and $s + \alpha d_s > 0$,

$$(x + \alpha d_x, s + \alpha d_s) \in N_2((1 - \alpha)\frac{1}{12} + \alpha^2\theta) \subseteq N_2(1/4),$$

as desired.

To see that $x + \alpha d_x > 0$, note that each component of $T^{-1}x$ is at least $\sqrt{11\mu/12}$, but the norm of $\alpha T^{-1}d_x$ is at most $\sqrt{\theta\mu}/\sqrt{6\theta}$, which is strictly smaller. A similar argument shows that $s + \alpha d_s > 0$. □

A substantially tighter analysis of both the predictor step and the corrector step is possible here. One improvement in both proofs is to bound

$$||D_x d_s||_2 \leq \frac{\sqrt{2}}{4}\left|\left|X^{1/2}S^{1/2}e - X^{-1/2}S^{-1/2}e\right|\right|_2^2.$$

and to control the 2-norm on the right-hand side using knowledge of which neighbourhood of the central path (x, s) lies in. See [52] for details.

2.3 Centrality in convex optimisation

Interior-point methods solve a primal problem of the form

$$\begin{array}{rl} \inf & c^T x \\ \text{subject to} & Ax = b \\ & x \in K. \end{array} \tag{2.5}$$

for some pointed convex cone K with interior. The dual of this problem is

$$\begin{array}{rl} \sup & b^T y \\ \text{subject to} & A^T y + s = c \\ & s \in K^*. \end{array} \tag{2.6}$$

I will assume hereafter that Slater's condition holds for both (2.5) and (2.6)—that is, that there are $\overset{\circ}{x}$ and $\overset{\circ}{s}$ such that

- $A\overset{\circ}{x} = b$,
- $\overset{\circ}{x} \in \operatorname{int} K$,
- $c - \overset{\circ}{s} \in \operatorname{im} A^T$, and
- $\overset{\circ}{s} \in \operatorname{int} K^*$.

This implies strong duality, therefore I am justified in writing min and max instead of inf and sup in what follows.

Suppose F is a θ-logarithmically homogeneous self-concordant barrier for K. Then, for any choice of $\mu \in \mathbb{R}_{++}$, one can write the problem

$$\begin{array}{rl} \min & c^T x + \mu F(x) \\ \text{subject to} & Ax = b. \end{array} \tag{2.7}$$

It happens that the dual to this problem is

$$\begin{array}{rl} \sup & b^T y - \mu F_*(s) \\ \text{subject to} & A^T y + s = c. \end{array} \tag{2.8}$$

Since F is strictly convex, for every choice of μ both (2.7) and (2.8) have unique optimal solutions, and these solutions again vary continuously with μ. The central path can be defined in this context as well; $x(\mu)$ is the unique optimal solution to (2.7) and $(y(\mu), s(\mu))$ is the unique optimal solution to (2.8).

Writing the first-order optimality conditions for (2.7), $x(\mu)$ must satisfy $Ax = b$ and $c^T + \mu F'(x) + \lambda^T A = 0$ for some choice of $\lambda \in \mathbb{R}^d$—that is,

$$\begin{aligned} Ax(\mu) &= b \text{ and} \\ c^T + \mu F'(x(\mu)) &\in \operatorname{im} A^T. \end{aligned}$$

This last condition means that $-\mu F'(x(\mu))$ must be a feasible choice of s in the dual (2.8).

Doing the same for (2.8), there must exist a real vector λ such that $y(\mu)$ and $s(\mu)$ satisfy $A^T y + s = c$ and $b^T + \lambda^T A^T = 0$ and $-\mu F_*'(s) + \lambda^T = 0$—that is, that $\mu A F_*'(s) = b$. This last condition means that $-\mu F_*'(s)$ must be a feasible solution x for the primal (2.7).

Indeed, at the primal solution $x = -\mu F_*'(s(\mu))$, the gradient of the objective is $c^T + \mu F'(-\mu F_*'(s(\mu)))$. Since F' is homogeneous of degree (-1), this is $c^T + F'(-F_*'(s(\mu)))$. Since $-F'(-F_*'(s)) = s$, this is $c^T - s(\mu)$. Therefore $-\mu F_*'(s(\mu))$ is the optimal solution to (2.7). One can similarly show that $-\mu F'(x(\mu))$ is the optimal solution to (2.8). This means in particular that $x(\mu) = -\mu F_*'(s(\mu))$ and $s(\mu) = -\mu F'(x(\mu))$.

Given x and s not necessarily on the central path, define $\tilde{x} := -F_*'(s)$ and $\tilde{s} = -F'(x)$. Also define $\mu = s^T x/\theta$, $\delta_P = x - \mu\tilde{x}$, and $\delta_D = s - \mu\tilde{s}$. Thus, if (x, s) is central, then $\delta_P = 0$ and $\delta_D = 0$.

Note that $\mu\tilde{x}$ is not necessarily a feasible solution to (2.7) and $\mu\tilde{s}$ is not necessarily a feasible solution to (2.8). However, the differences δ_P and δ_D measure the distance to centrality.

2.4 Primal-dual scalings

Identifying the dual space $\mathbb{E}^*$ with the primal space $\mathbb{E}$ via some inner product on $\mathbb{E}$, Tunçel [85] defines three sets of linear transformations whose squares map $\mathbb{E}^*$ to $\mathbb{E}$, namely $\mathcal{T}_0$, $\mathcal{T}_1$, and $\mathcal{T}_2$. The index indicates how much information about the barrier is preserved by the scaling—zeroth-order information, first-order information, and some second-order information, respectively. To an extent that increases with the index i, $\mathcal{T}_i$ allows an extension of the v-space analysis presented in Section 2.2 to convex optimisation.

Definition 2.4.1. Let K be a convex cone in $\mathbb{E} = \mathbb{R}^d$. For every pair $(x, s) \in \operatorname{int}(K) \oplus \operatorname{int}(K^*)$, define

$$\mathcal{T}_0(x, s) := \left\{ T \in \mathbb{S}^d_{++} : T^2(s) = x \right\}.$$

In words, $\mathcal{T}_0(x, s)$ is the set of all symmetric positive-definite linear operators on $\mathbb{R}^d$ whose squares map s to x.

Wei [94] proved that a certain natural primal-dual affine-scaling algorithm based on $\mathcal{T}_0$ need not converge to an optimal solution.

Definition 2.4.2. Let K be a convex cone in $\mathbb{E} = \mathbb{R}^d$. For every pair $(x, s) \in \text{int}(K) \oplus \text{int}(K^*)$, define

$$\mathcal{T}_1(x, s) := \left\{ T \in \mathbb{S}^d_{++} : T^2(s) = x, T^2(F'(x)) = F'_*(s) \right\}.$$

In words, $\mathcal{T}_1(x, s)$ is the set of all symmetric positive-definite, linear operators on $\mathbb{R}^d$ whose squares map s to x and $F'(x)$ to $F'_*(s)$.

Note that mapping s to x and mapping $F'(x)$ to $F'_*(s)$ is equivalent to mapping s to x and mapping δ_D to δ_P.

Tunçel [85] showed that a certain class of potential-reduction methods based on $\mathcal{T}_1$ always converges to an optimal solution.

Definition 2.4.3. Let K be a convex cone in $\mathbb{E} = \mathbb{R}^d$. For every pair $(x, s) \in \text{int}(K) \oplus \text{int}(K^*)$, and every positive real ξ, define

$$\mathcal{T}_2(\xi, x, s) := \left\{ T \in \mathbb{S}^d_{++} \quad \vdots \quad \begin{array}{ll} T^2(s) = x, & \\ T^2(F'(x)) = F'_*(s), & \\ & \\ \text{For all } z \in \mathbb{E}, & \xi^{-1}(\mu(x,s)F''(x))^{-1}[z,z] \\ & \leq \; T^2[z,z] \\ & \leq \; \xi(\mu(x,s)F''(x))^{-1}[z,z] \\ & \\ \text{For all } z \in \mathbb{E}, & \xi^{-1}(\mu(x,s)F''_*(s))[z,z] \\ & \leq \; T^2[z,z] \\ & \leq \; \xi(\mu(x,s)F''_*(s))[z,z] \end{array} \right\}.$$

In words, $\mathcal{T}_2(\xi, x, s)$ is the set of all symmetric positive-definite linear operators on $\mathbb{R}^d$ whose squares map s to x and $F'(x)$ to $F'_*(s)$, and which approximate $\mu(x, s)F''_*(s)$ and $(\mu(x, s)F''(x))^{-1}$ within a factor ξ.

The meaning of ξ here is at odds with [85], in which ξ is measured relative to a measure of centrality.

Nesterov and Todd proved the following strong result for symmetric cones. The number $\langle x, s\rangle \langle F'(x), F'(s)\rangle / \theta - \theta$ is a measure of centrality analogous to the arithmetic-harmonic mean gap. The factor $\frac{4}{3}$ in the following theorem cannot be reduced below 1.

Theorem 2.4.4 ([62])**.** *Let K be a symmetric cone and let F be a self-scaled θ-LHSCB for K. Let $x \in \text{int}\, K$ and $s \in \text{int}\, K^*$. Let*

$$\xi \geq 1 + \frac{4}{3}\left(\frac{\langle x, s\rangle \langle F'(x), F'(s)\rangle}{\theta} - \theta \right).$$

Then there exists a $w \in \text{int}\, K^*$ such that $F''_*(w) \in \mathcal{T}_2(\xi, x, s)$.

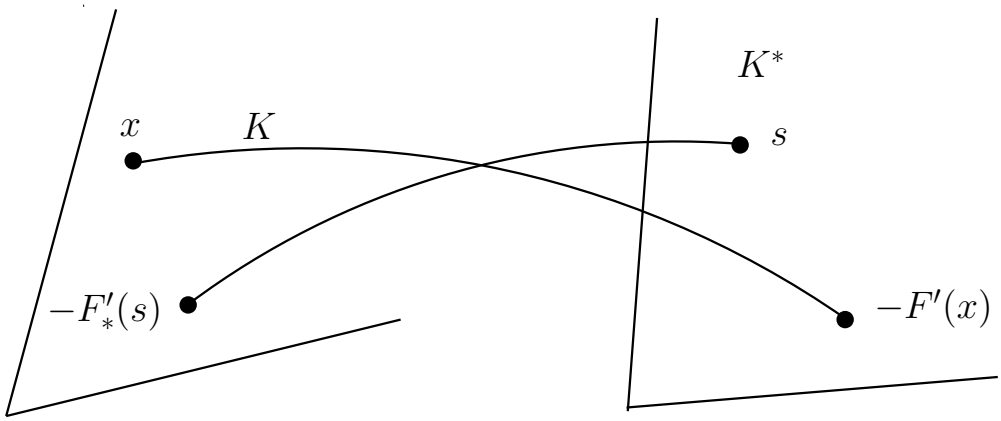

Figure 2.1: The action of $-F'$ on x and $-F'_*(s)$.

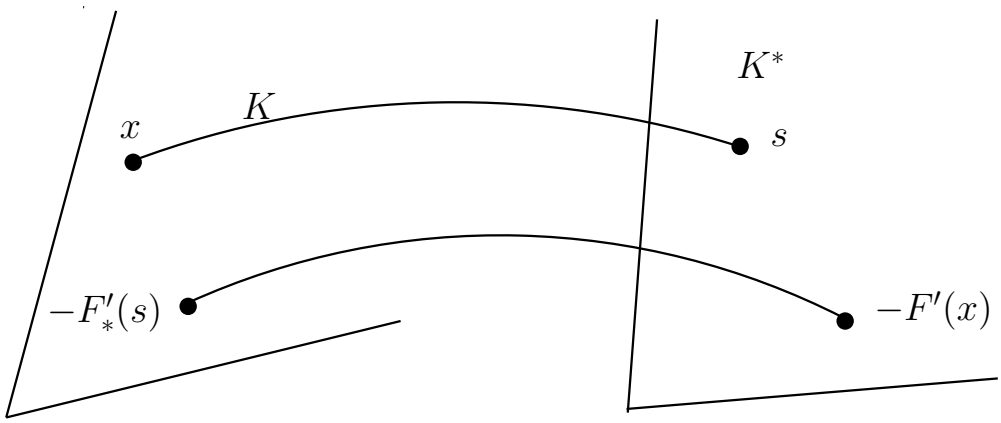

Figure 2.2: The action of a primal-dual scaling T^2 for T in $\mathcal{T}_1$ on x and $-F'_*(s)$.

This theorem gives a sufficient condition for $\mathcal{T}_2(\xi, x, s)$ to be nonempty when F is a self-scaled barrier.

Endow $\operatorname{int} K$ with the Riemannian metric induced by F''. By the Hopf-Rinow theorem from Riemannian geometry, between any two points of $\operatorname{int} K$ there is a length-minimising geodesic. This geodesic can be computed explicitly for products of the nonnegative ray, second-order cones, and real symmetric positive semidefinite matrix cones; see Section 6 of [63] for formulae and proofs.

In the three cases mentioned above, there is an explicit formula for the scaling point w; w is the midpoint of the geodesic between x and $-F'_*(s)$. Further, in these cases, $F''(w)$ is the midpoint of the geodesic between $F''(-F'_*(s))$ and $F''(x)$, where the space of symmetric bilinear forms on the affine hull of K is given the Riemannian metric induced by the Hessian of $-\log\det$.

In the case of semidefinite optimisation, where K is the set of $n \times n$ symmetric real positive semidefinite matrices and the barrier F is given by $F(x) = -\log\det x$, the geodesic between positive definite matrices A and B has the constant-speed parametrisation $t \mapsto A^{1/2}(A^{-1/2}BA^{-1/2})^t A^{1/2}$. Thus the midpoint of the geodesic between x and $\tilde{x} = -F'(s)$ is

$$\begin{aligned} & x^{1/2}(x^{-1/2}\tilde{x}x^{-1/2})^{1/2}x^{1/2} \\ = \; & \tilde{x}^{1/2}(\tilde{x}^{-1/2}x\tilde{x}^{-1/2})^{1/2}\tilde{x}^{1/2}. \end{aligned}$$

The case of self-scaled barriers is interesting in that a primal-dual scaling in $\mathcal{T}_1$ given by $(F''(w))^{1/2}$ for some point $w \in \operatorname{int} K$ always exists. Despite being the intersection of a symmetric cone with a linear space, it can be shown that the *Vinberg cone*—the 5-dimensional cone given by

$$\left\{ (a,b,c,d,e) \in \mathbb{R}^5 : \begin{pmatrix} a & b & 0 \\ b & c & d \\ 0 & d & e \end{pmatrix} \text{ is positive semidefinite} \right\}$$

—does not have this property. Indeed, it can be shown that the only linear slices of the positive semidefinite cone for which a $\mathcal{T}_1$ scaling always exists are those that are closed under the Jordan product $(AB + BA)/2$, which implies that they are themselves symmetric cones. See Proposition 6.2 of [56] for a proof of this fact.

2.5 Primal-dual scalings based on Hessian integration

Theorem 2.5.1. *Let F be a LHSCB for K and $(x, s) \in \operatorname{int}(K) \oplus \operatorname{int}(K^*)$. Let $\mu = \mu(x, s)$. Let $\tilde{x} = -F'_*(s)$ and $\tilde{s} = -F'(x)$. Let $\delta_P = x - \mu\tilde{x}$ and $\delta_D = s - \mu\tilde{s}$. Then the linear*

transformation

$$T_D^2 := \mu \int_0^1 F_*''(s - t\delta_D)dt \tag{2.9}$$

is symmetric, positive-definite, maps s to x, and maps $\tilde{s}$ to $\tilde{x}$. Therefore, its unique symmetric positive-definite square root T_D is in $\mathcal{T}_1(x, s)$.

Proof. Using the fundamental theorem of calculus (for the second equation below) followed by the property $-F_*'(-F'(x)) = x$ from Theorem 1.5.1 (for the third equation below), we obtain

$$T_D^2\delta_D = \mu \int_0^1 F_*''(s - t\delta_D)\delta_D dt = \mu\left(F_*'(s - \delta_D) - F_*'(s)\right) = \mu\left(x/\mu - \tilde{x}\right) = \delta_P.$$

Next compute, using the substitution $\tilde{t} = 1/t$ and the degree-(-2)-homogeneity of F'',

$$\begin{aligned} T_D^2 s &= \mu \int_0^1 F_*''(s - t\delta_D)s dt \\ &= \mu \int_0^1 \frac{1}{t^2} F_*''(s/t - \delta_D)s dt \\ &= \mu \int_1^\infty F_*''(\tilde{t}s - \delta_D)s d\tilde{t} \\ &= -\mu F'(s - \delta_D) = x. \end{aligned}$$

Further, T_D^2 is the mean of some symmetric positive-definite linear transformations, so T_D^2 itself is symmetric and positive-definite. □

This scaling operator is called the *dual integral scaling*. Note that the above theorem holds under weaker assumptions (we only used the facts that F is $\mathcal{C}^3$ and logarithmically homogeneous and that F and F_* are Legendre-type functions in the sense of Rockafellar [75]). The dual integral scaling is expected to inherit many of the nice properties of the dual Hessian. Thus, if F_* is well-behaved, then one can prove nice bounds on the deviation of dual integral scaling from the dual Hessian at s and $\mu\tilde{s}$:

Theorem 2.5.2. *If $\sigma < 1$ is such that, for every $t \in [0, 1]$ and every $h \in \mathbb{R}^d$*

$$(1 - t\sigma)^2 F_*''(s)[h, h] \leq F_*''(s - t\delta_D)[h, h] \leq \frac{1}{(1 - t\sigma)^2} F_*''(s)[h, h],$$

then for every $h \in \mathbb{R}^d$

$$(1 - \sigma)\mu(x, s)F_*''(s)[h, h] \leq T_D^2[h, h] \leq \frac{1}{1 - \sigma}\mu(x, s)F_*''(s)[h, h].$$

Proof. This follows directly from the definition of T_D^2. □

Interestingly, the dual integral scaling (the mean of Hessians along the line segment joining s and $\mu(x,s)\tilde{s}$) is not as "canonical" as the Nesterov–Todd scaling (the geodesic mean of the Hessians joining the same two points in the interior of K^*) in terms of primal-dual symmetry properties. For the remainder of this section, we elaborate on this and related issues. In Chapter 4 when we specialise to *hyperbolicity cone optimisation* problems, we show that the integral scaling can have advantages when one of the primal and dual problems is somehow "nicer" for the approach at hand.

Notice that the dual integral scaling

$$\int_0^1 \mu(x,s)F_*''(ts+(1-t)\mu\tilde{s})dt$$

and the primal integral scaling

$$T_P^2 := \left(\int_0^1 \mu(x,s)F''(tx+(1-t)\mu(x,s)\tilde{x})dt\right)^{-1}$$

are both scalings that map s to x and $\tilde{s}$ to $\tilde{x}$. These are not in general the same, although they do coincide with the usual scaling XS^{-1} in the case of linear optimisation.

Example 2.5.3. (A comparison of primal-dual local metrics for the positive semidefinite cone)
We work out the integral scaling for the positive semidefinite cone $\mathbb{S}_+^\theta$ equipped with the self-scaled barrier $F(X) = -\log\det X$. If X is the primal iterate, $\tilde{X} = S^{-1}$ is the barrier gradient at the dual iterate, and $\mu = \langle X, S\rangle/\theta$, then we see that

$$T_D^2[H,H] = \mu\int_0^1 \left\langle (tX+(1-t)\mu\tilde{X})^{-1}H(tX+(1-t)\mu\tilde{X})^{-1}, H\right\rangle dt.$$

One can make this slightly more explicit. There always exists a $U \in GL(n)$ such that $UXU^\top = I$ and $U\mu\tilde{X}U^\top$ is diagonal; one can compose a Q that orthogonally diagonalises X, an S that scales $Q^\top XQ$ to the identity matrix, and a Q' that orthogonally diagonalises $SQ^\top\mu\tilde{X}QS$. Say $U\mu\tilde{X}U^\top = D$. Then we can compute

$$T_D^2[UHU^\top, UHU^\top] = \mu\int_0^1 \left\langle (tI+(1-t)D)^{-1}H(tI+(1-t)D)^{-1}, H\right\rangle dt.$$

In particular, if H is E_{ij}, the matrix with a 1 in the ij-position and zeroes elsewhere, and $D_i \neq D_j$, we have

$$T_D^2[UE_{ij}U^\top, UE_{ij}U^\top] = \mu\int_0^1 (t+(1-t)D_i)^{-1}(t+(1-t)D_j)^{-1}dt = \mu\frac{\ln D_j - \ln D_i}{D_j - D_i}.$$

Special attention needs to be given to the case when $D_i = D_j$; here, the integral evaluates to μ/D_i.

If $H = E_{ij} + E_{kl}$, we have $T_D^2[U(E_{ij} + E_{kl})U^\top, U(E_{ij} + E_{kl})U^\top]$ given by

$$\mu \int_0^1 (t + (1-t)D_i)^{-1}(t + (1-t)D_j)^{-1} + (t + (1-t)D_k)^{-1}(t + (1-t)D_l)^{-1} dt.$$

This is the sum of $T^2[UE_{ij}U^\top, UE_{ij}U^\top]$ with $T_D^2[UE_{kl}U^\top, UE_{kl}U^\top]$, meaning that

$$T_D^2[UE_{ij}U^\top, UE_{kl}U^\top] = 0$$

if $i \neq k$ or $j \neq l$. That is, $T_D^2[U - U^\top, U - U^\top]$ is diagonal with respect to the standard basis. Put another way, the operator $C_{U^\top} T_D^2 C_U$ is diagonal where C_U is the conjugation operator given by $C_U(Z) = UZU^\top$. For every nonsingular U, the map C_U preserves operator geometric means; that is,

$$\begin{aligned} &UA^{1/2}\left(A^{-1/2}BA^{-1/2}\right)^{1/2}A^{1/2}U^\top \\ &\qquad = (UAU^\top)^{1/2}\left((UAU^\top)^{-1/2}UBU^\top(UAU^\top)^{-1/2}\right)^{1/2}(UAU^\top)^{1/2}. \end{aligned}$$

One can show this as follows: The geometric mean of $X \succ 0$ and $Y \succ 0$ is the unique positive definite G such that $GX^{-1}G = Y$. Taking $H = UGU^\top$, we have

$$HU^{-\top}X^{-1}U^{-1}H = UGU^\top U^{-\top}X^{-1}U^{-1}UGU^\top = UGX^{-1}GU^\top = UYU^\top.$$

Interestingly, Molnár [53] proved that *every* linear automorphism of the semidefinite cone over a complex Hilbert space that preserves geometric means is a conjugation operator. The converse is also true, since the set of automorphisms of $\mathbb{S}_+^\theta$ is the set of conjugations given by the nonsingular U (see a discussion in [84] of Güler's proof utilizing the proof technique of Waterhouse [93]), and as we observed above, it is easy to verify that the conjugation operator preserves operator geometric means. Thus, we can make a natural comparison with the Nesterov–Todd scaling given by

$$N[H, H] = \langle (VD'^{1/2}V^\top)^{-1}H(VD'^{1/2}V^\top)^{-1}, H\rangle,$$

where $VD'V^\top$ is the spectral decomposition of operator geometric mean of X and $\mu\tilde{X}$. Notice that

$$N[VE_{ij}V^\top, VE_{ij}V^\top] = \langle D'^{-1/2}E_{ij}D'^{-1/2}, E_{ij}\rangle = \frac{1}{\sqrt{D'_i D'_j}}$$

and that N is similarly diagonal with respect to that basis. (The conjugation $C_U N C_U^\top$ does not in general result in a diagonal matrix, so we need to take this different V instead in order to diagonalise N.) Notice that

$$N[UE_{ij}U^\top, UE_{ij}U^\top] = e_i^\top U^\top GUe_i e_j^\top U^\top GUe_j$$

whatever U is. Thus, when we form the matrix whose ij entry is $N[UE_{ij}U^\top, UE_{ij}U^\top]$, we always obtain a rank-one matrix. However, such a matrix formed from the integral scaling

T_D^2 can have full rank. We proceed with an example. Consider $D = (\epsilon, \epsilon^2, \ldots, \epsilon^n)$. Notice that $\ln D_i - \ln D_j = (i-j)\ln\epsilon$, while $D_i - D_j = \epsilon^i - \epsilon^j$. Thus, the (i,j) entry of this matrix is on the order of $\epsilon^{-\min(i,j)}$. For sufficiently small ϵ, then, the determinant of this matrix is dominated by the product along the diagonal, which is positive—it follows that this matrix is nonsingular.

It is not always clear how (or even whether) T_D^2 or T_P^2 can be computed explicitly. Algorithms based on the primal or dual integral scalings may need to work with approximations instead. In Sections 2.7 and 2.9, I show that one particular approximation to the dual integral scaling corresponding to the midpoint quadrature rule leads to a direct generalisation of the Mizuno-Todd-Ye method of Section 2.2 to convex optimisation.

2.6 Primal-dual scalings by low-rank update formulae

This section presents a rank-four update that turns an approximation to a primal-dual scaling into a primal-dual scaling. In this section, μ is used either as $\mu(\tilde{x}, \tilde{s})$ or as μ alone; in the latter case, $\mu(x, s)$ is meant.

A positive-definite approximation H to the dual integral scaling need not satisfy the equations $Hs = x$ and $H\tilde{s} = \tilde{x}$. Tunçel [85] constructed the following low-rank update which "repairs" such problems encountered by *any* symmetric, positive-definite H:

$$T_H^2 := H + a_1 xx^T + g_1 Hss^T H + \tilde{a_1}\tilde{x}\tilde{x}^T + \tilde{g_1} H\tilde{s}\tilde{s}^T H + a_2(x\tilde{x}^T + \tilde{x}x^T) + g_2(Hs\tilde{s}^T H + H\tilde{s}s^T H), \tag{2.10}$$

where

$$\tilde{\mu} = \frac{\tilde{s}^T\tilde{x}}{\theta} = \mu(\tilde{x}, \tilde{s}),$$

$$a_1 = \frac{\tilde{\mu}}{\theta(\mu\tilde{\mu} - 1)}, \tilde{a}_1 = \frac{\mu}{\theta(\mu\tilde{\mu} - 1)}, a_2 = \frac{-1}{\theta(\mu\tilde{\mu} - 1)},$$

$$g_1 = \frac{\tilde{s}^T H\tilde{s}}{s^T Hs\tilde{s}^T H\tilde{s} - (\tilde{s}Hs)^2}, \tilde{g}_1 = \frac{s^T Hs}{s^T Hs\tilde{s}^T H\tilde{s} - (\tilde{s}Hs)^2}, g_2 = \frac{s^T H\tilde{s}}{s^T Hs\tilde{s}^T H\tilde{s} - (\tilde{s}Hs)^2}.$$

Tunçel [85] proved that, as long as H is positive-definite, T_H^2 is positive-definite, maps s to x, and maps $\tilde{s}$ to $\tilde{x}$.

As written, Equation (2.10) is not easy to analyse. It is not immediately obvious that, in the case that the pair (x, s) satisfy the centrality condition $x = \mu\tilde{x}$, the formula collapses to a rank-two update. (Indeed, it has a singularity there since $\mu\tilde{\mu} = 1$ on the central path.) The above update can be written equivalently as two consecutive classical rank-two updates:

Theorem 2.6.1. *Let $x \in \mathbb{E}, \tilde{x} \in \mathbb{E}$, $s \in \mathbb{E}^*, \tilde{s} \in \mathbb{E}^*$, $\mu \in \mathbb{R}_{++}$, and $\theta \in \mathbb{R}_{++}$ satisfy the following conditions (under the definitions $\delta_P := x - \mu\tilde{x}$ and $\delta_D := s - \mu\tilde{s}$):*

- $0 < \langle \tilde{s}, x \rangle = \langle s, \tilde{x} \rangle =: \theta$
- $0 < \langle s, x \rangle =: \theta\mu$
- $0 < \langle \tilde{s}, \tilde{x} \rangle =: \theta\tilde{\mu}$
- $\langle \delta_D, \delta_P \rangle > 0$.

Further let H be some symmetric positive-definite matrix. Then, H_2 in the following formula is symmetric, positive-definite, maps s to x, and maps $\tilde{s}$ to $\tilde{x}$:

$$\begin{aligned} H_1 &:= H + \frac{1}{\langle s, x \rangle} x x^T - \frac{1}{\langle s, Hs \rangle} H s s^T H \\ H_2 &:= H_1 + \frac{1}{\langle \delta_D, \delta_P \rangle} \delta_P \delta_P^T - \frac{1}{\langle \delta_D, H_1 \delta_D \rangle} H_1 \delta_D \delta_D^T H_1. \end{aligned} \tag{2.11}$$

Proof. Notice that $H_1 s = Hs + x - Hs = x$. Notice further that $\langle s, \delta_P \rangle = \langle \delta_D, x \rangle = 0$ by expanding the inner product conditions, so $H_2 s = H_1 s$. Thus, H_2 maps s to x. Next, note that $H_2 \delta_D = H_1 \delta_D + \delta_P - H_1 \delta_D = \delta_P$. Thus, H_2 also maps δ_D to δ_P. Hence, H_2 maps $\tilde{s}$ to $\tilde{x}$.

We recall from the theory of quasi-Newton updates (see for instance Lemma 9.2.1 in [17]) that the "curvature condition" $\langle s, x \rangle > 0$ is necessary and sufficient to guarantee that H_1 is positive definite, and, given that H_1 is positive-definite, the curvature condition $\langle \delta_D, \delta_P \rangle > 0$ is necessary and sufficient for H_2 to be positive-definite. Therefore, H_2 is positive-definite as well. □

Note that the positivity of the scalar products $\langle s, x \rangle$ and $\langle \delta_D, \delta_P \rangle$, together with the orthogonality conditions $\langle s, \delta_P \rangle = \langle \delta_D, x \rangle = 0$ suffice for the above theorem to hold. There may be a potential use of these formulae in quasi-Newton methods for unconstrained minimisation. Such considerations are left for future work.

We remark that we can apply the above formulas after switching x and s and then inverting the resulting T^2 to obtain the following low-rank updates (under the same conditions):

Theorem 2.6.2. *Let H, x, $\tilde{x}$, s, $\tilde{s}$, μ, and θ be as in Theorem 2.6.1. Then H_2 in the following formula is symmetric, positive-definite, maps s to x, and maps $\tilde{s}$ to $\tilde{x}$:*

$$\begin{aligned} H_1 &:= \left(I - \frac{x s^\top}{\langle s, x \rangle} \right) H \left(I - \frac{s x^\top}{\langle s, x \rangle} \right) + \frac{x x^\top}{\langle s, x \rangle} \\ H_2 &:= \left(I - \frac{\delta_P \delta_D^\top}{\langle \delta_D, \delta_P \rangle} \right) H_+ \left(I - \frac{\delta_D \delta_P^\top}{\langle \delta_D, \delta_P \rangle} \right) + \frac{\delta_P \delta_P^\top}{\langle \delta_D, \delta_P \rangle}. \end{aligned} \tag{2.12}$$

Proof. We again see that $H_1 s = x$ and $H_2 \delta_D = \delta_P$ by correctness of the rank-two updates above. We compute

$$H_2 s = \left(I - \frac{\delta_P \delta_D^\top}{\langle \delta_D, \delta_P \rangle}\right) H_1 \left(I - \frac{\delta_D \delta_P^\top}{\langle \delta_D, \delta_P \rangle}\right) s + \frac{\delta_P \delta_P^\top}{\langle \delta_D, \delta_P \rangle} s = \left(I - \frac{\delta_P \delta_D^\top}{\langle \delta_D, \delta_P \rangle}\right) x + 0 = x.$$

H_1 is positive-definite because the curvature condition $\langle s, x \rangle > 0$ is satisfied. H_2 is positive-definite because the curvature condition $\langle \delta_D, \delta_P \rangle > 0$ is satisfied. □

The rank-two updates used in the above theorems also appear in the DFP and BFGS quasi-Newton algorithms for unconstrained optimisation. See Nocedal and Wright's book [67] for descriptions of these methods.

In Theorems 2.6.1 and 2.6.2, H_1 is the closest (in some metric) self-adjoint operator to H satisfying $H_1 s = x$ and H_2 is the closest self-adjoint operator to H_1 satisfying $H_2 \delta_D = \delta_P$. Analogous to *Broyden's convex class* of rank-two updates, convex combinations of the formulae defining H_1 and H_2 can be used as well, and the convex combination does not necessarily need to be the same for H_1 and H_2.

2.7 Approximating the dual integral scaling by the midpoint rule

The purpose of this section is to prove that, within a usefully large neighbourhood of the central path, $\mu F_*''((s + \mu\tilde{s})/2)$ updated with (2.11) gives a primal-dual scaling in $\mathcal{T}_2(\xi, x, s)$ for some uniformly bounded ξ. These results are valid within the neighbourhood $||\delta_D||_s \leq \frac{1}{50}$.

Write $\check{s} := (s + \mu\tilde{s})/2$.

A sketch of this proof is as follows:

- The approximation error in direction s is small; $||F_*''(\check{s})s - x||_s^* \in O(\mu\,||\delta_D||_s)$.
- Therefore, the first rank-two update is small; in a certain sensible operator norm, it is $O(\mu\,||\delta_D||_s)$.
- The approximation error in direction δ_D is also small; $||F_*''(\check{s})\delta_D - \delta_P||_s^* \in O(\mu\,||\delta_D||_s^2)$.
- The first rank-two update does not substantially worsen the error in direction δ_D.
- The second rank-two update does not disturb s.
- The second rank-two update is small; in operator norm, it is $O(\mu\,||\delta_D||_s)$

- The midpoint approximation had $\xi \in O(1)$ and we applied two updates that push ξ up by $O(||\delta_D||_s)$ with $||\delta_D||_s \in O(1)$, so we get a primal-dual scaling with $\xi \in O(1)$.

The "certain sensible operator norm" above is the operator norm induced by $||\cdot||_s$, which is defined by

$$||H||_s := \max_{||v||_s=1} ||Hv||_s^*.$$

The following lemmata are used many times.

Lemma 2.7.1. *Let $s \in \text{int}(K^*)$, $h \in \mathbb{E}^*$ such that $||h||_s < 1$. Then*

$$||F_*''(s+h)||_s = \max_{u \in \mathbb{E}^*: ||u||_s=1} ||F_*''(s+h)u||_s^* \leq \frac{1}{(1-||h||_s)^2}.$$

Proof. Let s and h be as in the statement of the lemma. Then, for every $z \in \mathbb{R}^d$,

$$F_*''(s+h)[z,z] \leq \frac{1}{(1-||h||_s)^2} F_*''(s)[z,z]$$

by the Dikin ellipsoid bound. Thus, the maximum eigenvalue of

$$[F_*''(s)]^{-1/2} F_*''(s+h) [F_*''(s)]^{-1/2}$$

is bounded above by

$$\frac{1}{(1-||h||_s)^2}.$$

The square of this quantity is an upper bound on the largest eigenvalue of

$$[F_*''(s)]^{-1/2} F_*''(s+h) [F_*''(s)]^{-1} F_*''(s+h) [F_*''(s)]^{-1/2}.$$

Therefore, the supremum in the statement of the lemma is bounded above by $\frac{1}{\left(1-||h||_s\right)^2}$ as desired. □

Lemma 2.7.2. *Let h and u lie in $\mathbb{E}$ and $s \in \text{int}(K^*)$. Then*

$$||hh^\top - uu^\top||_s \leq ||h-u||_s^* ||h+u||_s^*.$$

Proof. By the triangle inequality,

$$\begin{aligned} 2||hh^\top - uu^\top||_s &= \left|\left|(h-u)(h+u)^\top + (h+u)(h-u)^\top\right|\right|_s \\ &\leq \left|\left|(h-u)(h+u)^\top\right|\right|_s + \left|\left|(h+u)(h-u)^\top\right|\right|_s. \end{aligned}$$

Now we compute

$$\begin{aligned}\left|\left|(h-u)(h+u)^\top\right|\right|_s &= \sup_{||z||_s\le 1} ||(h-u)\langle z, h+u\rangle||_s^* \\ &= \sup_{||z||_s\le 1} \langle z, h+u\rangle \, ||h-u||_s^* = ||h+u||_s^* \, ||h-u||_s^* ;\end{aligned}$$

and similarly

$$\left|\left|(h+u)(h-u)^\top\right|\right|_s = ||h+u||_s^* \, ||h-u||_s^* .$$

Adding these together gives the desired result. □

The result advertised at the beginning of this section reads as follows:

Theorem 2.7.3. *If $||\delta_D||_s \le 1/50$, then one can construct a $T \in \mathbb{S}^n$ satisfying the following properties:*

- *T is positive definite;*
- *$T^2 s = x$;*
- *$T^2 \tilde{s} = \tilde{x}$;*
- *For every $z \in \mathbb{R}^d$,*

$$0.814905\, \mu F_*''(s)[z,z] \le T^2[z,z] \le 1.185500\, \mu F_*''(s)[z,z]$$

- *For every $z \in \mathbb{R}^d$,*

$$0.808093\, (\mu F''(x))^{-1}[z,z] \le T^2[z,z] \le 1.192311\, (\mu F''(x))^{-1}[z,z].$$

That is, $T \in \mathcal{T}_2(1.237483; x, s)$.

Proof. Repeatedly use the triangle inequality, the Cauchy-Schwarz inequality, Lemma 2.7.2, and Lemma 2.7.1 on the update in Theorem 2.6.1 applied to $\mu F''((s+\mu\tilde{s})/2)$.

See Appendix A for details. □

2.8 Primal-dual scalings from other symmetric linear maps

The previous section gave, within a constant-sized neighbourhood of the central path, an explicit construction of a primal-dual scaling approximating $\mu F_*''(s)$ and $\frac{1}{\mu}(F''(x))^{-1}$ using

the midpoint rule for integration. One may want to use a quadrature rule other than the midpoint rule, and one may also want to measure centrality in a norm more closely approximating the shape of the cone K. Much of last section's analysis can be done in this context, and the purpose of this section is to show how.

Let K be a convex cone in some real vector space $\mathbb{E}$ and let K^* be its dual in $\mathbb{E}^*$. For $x \in \operatorname{int} K$, define $||v||_{K,x}$ to be the smallest positive λ such that $x + v/\lambda$ and $x - v/\lambda$ both lie in K. One can check that $||v||_{K,x}$ is a norm. Let $||v||^*_{K,x}$ be its dual norm. If $H : \mathbb{E}^* \times \mathbb{E}^* \to \mathbb{R}$ is a symmetric bilinear form, define

$$||H||_{K,x} := \max_{||v||^*_{K,x} \leq 1} \max_{||w||^*_{K,x} \leq 1} H[v, w];$$

this is the operator norm induced by the norm $||\cdot||_{K,x}$.

Good bounds on the difference between two vectors in $||\cdot||_{K,x}$ imply some very useful conic inequalities:

Proposition 2.8.1. *If $||v - w||_{K,w} \leq \delta < 1$, then there are x and y in K such that $v = (1 - \delta)w + x = (1 + \delta)w - y$.*

Proof. Let $s \in K^*$. Then $s^T w \geq 0$ and $s^T(w \pm (v - w)/\delta) \geq 0$. Thus $s^T((\delta - 1)w + v) \geq 0$ and $s^T((1 + \delta)w - v) \geq 0$. This is true for every $s \in K^*$, so $(\delta - 1)w + v$ and $(1 + \delta)w - v$ both lie in K, as desired. □

The operator norm still behaves as expected on rank-one matrices:

Proposition 2.8.2. *Let $v, w \in \mathbb{E}$. Then the operator norm of the bilinear form vw^T is*

$$\left|\left|vw^T\right|\right|_{K,x} = ||v||_{K,x} \, ||w||_{K,x} \, .$$

Proof.

$$\left|\left|vw^T\right|\right|_{K,x} = \max_{||y||^*_{K,x} \leq 1} \max_{||z||^*_{K,x} \leq 1} v^T y w^T z = \left(\max_{||y||^*_{K,x} \leq 1} v^T y \right) \left(\max_{||z||^*_{K,x} \leq 1} w^T z \right) = ||v||_{K,x} \, ||w||_{K,x} \, .$$

□

The difference-of-squares bound also still holds:

Proposition 2.8.3. *Let $v, w \in \mathbb{E}$. Then*

$$\left|\left|vv^T - ww^T\right|\right|_{K,x} \leq ||v - w||_{K,x} \, ||v + w||_{K,x}$$

Proof.

$$vv^T - ww^T = \frac{1}{2}\left((v-w)(v+w)^T + (v+w)(v-w)^T\right).$$

Now apply the triangle inequality and the previous proposition. □

In the circumstances that arise in this thesis, one can control the $||\cdot||_{K,x}$ norm of the quasi-Newton updates from Section 2.6 in terms of the $||\cdot||_{K,x}$ norm of the correction to be made. This is the purpose of the two following theorems. The following theorem makes special use of the fact that s lies in K^* and x in K to get a strong bound:

Theorem 2.8.4. *Let $K \subseteq \mathbb{E}$ be a convex cone. Let $x \in \operatorname{int} K$ and $s \in \operatorname{int} K^*$. Let $H : \mathbb{E}^* \to \mathbb{E}$ be a self-adjoint, positive definite linear map. Suppose $||Hs - x||_{K,x} \leq \delta < 1$. Then*

$$\left|\left|\frac{Hss^TH}{s^THs} - \frac{xx^T}{s^Tx}\right|\right|_{K,x} \leq \delta\frac{3+\delta}{1-\delta}.$$

Proof. By the triangle inequality,

$$\left|\left|\frac{Hss^TH}{s^THs} - \frac{xx^T}{s^Tx}\right|\right|_{K,x} \leq \left|\frac{s^T(x-Hs)}{s^Txs^THs}\right| \left|\left|Hss^TH\right|\right|_{K,x} + \frac{1}{s^Tx}\left|\left|xx^T - Hss^TH\right|\right|_{K,x}. \quad (2.13)$$

Use Proposition 2.8.1 to write $Hs = (1-\delta)x + y = (1+\delta)x - z$ for y and z in K. Then $s^T(x-Hs) = s^T(\delta x - y) \leq \delta s^Tx$ and $s^T(x-Hs) = s^T(z-\delta x) \geq -\delta s^Tx$. Thus

$$\left|s^T(x-Hs)\right| \leq \delta s^Tx.$$

Thus

$$\left|\frac{s^T(x-Hs)}{s^Txs^THs}\right| \leq \left|\frac{\delta s^Tx}{s^Tx(s^Tx + s^T(Hs-x))}\right| \leq \frac{\delta}{(1-\delta)s^Tx}.$$

Since $||Hs||_{K,x} \leq ||x||_{K,x} + ||Hs-x||_{K,x} \leq 1+\delta$, It follows that the first term on the right-hand side of (2.13) is bounded above by

$$\frac{\delta\,||Hs||_{K,x}^2}{(1-\delta)s^Tx} \leq \frac{\delta(1+\delta)^2}{(1-\delta)s^Tx}.$$

The second term on the right-hand side of (2.13) can be bounded above by

$$\frac{(2+\delta)\delta}{s^Tx}.$$

This implies the desired bound. □

The following theorem is very similar, but it applies more generally, the bound is weaker, and the setup is more awkward:

Theorem 2.8.5. *Let $K \subseteq \mathbb{E}$ be a convex cone. Let $\delta_P \in \mathbb{E}$ and $\delta_D \in \mathbb{E}^*$ be such that $\delta_D^T \delta_P > 0$. Let $x \in \operatorname{int} K$. Let $H : \mathbb{E}^* \to \mathbb{E}$ be a symmetric, positive definite linear map and let T^2 be some symmetric, positive definite linear map such that $T^2 \delta_D = \delta_P$. Suppose $\sigma < 1$ is such that, for every $z \in \mathbb{E}$,*

$$(1-\sigma)^2 T^2[z,z] \leq H[z,z] \leq \frac{T^2[z,z]}{(1-\sigma)^2}.$$

Suppose $\delta < 1$ is such that $||H\delta_D - \delta_P||_{K,x} \leq \delta \, ||H\delta_D||_{K,x}$. Then

$$\left|\left| \frac{\delta_P \delta_P^T}{\delta_D^T \delta_P} - \frac{H\delta_D \delta_D^T H}{\delta_D^T H \delta_D} \right|\right|_{K,x} \leq \left(\frac{\sigma(2+\sigma)}{1-4\sigma} + \delta(2+\delta) \right) \frac{||H\delta_D||_{K,x}^2}{\delta_D^T \delta_P}.$$

Proof. Note that

$$\begin{aligned} & \left|\left| \frac{\delta_P \delta_P^T}{\delta_D^T \delta_P} - \frac{H\delta_D \delta_D^T H}{\delta_D^T H \delta_D} \right|\right|_{K,x} \\ \leq & \left| \frac{\delta_D^T (H\delta_D - \delta_P)}{\delta_D^T \delta_P \delta_D^T H \delta_D} \right| \left|\left| H\delta_D \delta_D^T H \right|\right|_{K,x} + \frac{1}{\delta_D^T \delta_P} \left|\left| \delta_P \delta_P^T - H\delta_D \delta_D^T H \right|\right| \\ \leq & \left| \frac{\delta_D^T (H\delta_D - \delta_P)}{\delta_D^T H \delta_D} \right| \frac{||H\delta_D||_{K,x}^2}{\delta_D^T \delta_P} + \frac{\delta(2+\delta) \, ||H\delta_D||_{K,x}^2}{\delta_D^T \delta_P}. \end{aligned}$$

Write $\delta_D^T (H\delta_D - \delta_P) = \delta_D^T (H - T^2)\delta_D$. Then

$$\left((1-\sigma)^2 - 1\right) \delta_D^T T^2 \delta_D \leq \delta_D^T (H - T^2) \delta_D \leq \left(\frac{1}{(1-\sigma)^2} - 1 \right) \delta_D^T T^2 \delta_D.$$

Thus

$$\left| \delta_D^T (H\delta_D - \delta_P) \right| \leq \frac{2\sigma + \sigma^2}{(1-\sigma)^2} \delta_D^T \delta_P.$$

Writing $\delta_D^T H \delta_D = \delta_D^T \delta_P + \delta_D^T (H\delta_D - \delta_P)$, it follows that

$$\left| \frac{\delta_D^T (H\delta_D - \delta_P)}{\delta_D^T H \delta_D} \right| \leq \frac{2\sigma + \sigma^2}{1 - 4\sigma}.$$

Thus

$$\begin{aligned} & \left| \frac{\delta_D^T (H\delta_D - \delta_P)}{\delta_D^T \delta_P \delta_D^T H \delta_D} \right| \left|\left| H\delta_D \delta_D^T H \right|\right|_{K,x} \\ \leq & \frac{2\sigma + \sigma^2}{1 - 4\sigma} \frac{||H\delta_D||_{K,x}^2}{\delta_D^T \delta_P}. \end{aligned}$$

The desired result follows. □

2.9 A Mizuno-Todd-Ye variant for convex optimisation

Tunçel [85] gave a potential-reduction method based on the primal-dual scaling machinery and proved that it generates an ϵ-optimal solution within $O(\sqrt{\theta}\log(1/\epsilon))$ iterations if, for every $x \in \operatorname{int} K$ and $s \in \operatorname{int} K^*$, one can always find a primal-dual scaling in $\mathcal{T}^2(\xi, x, s)$ for $\xi \leq C(\theta\mu\tilde{\mu} - \theta + 1)$ for a universal positive constant C. If the choice of primal-dual scaling is invariant under interchange of primal and dual problems, this method is primal-dual symmetric.

It is likely straightforward to modify Tunçel's method to work under the weaker assumption that there exist positive constants C_1 and C_2 such that $\xi \leq C_1$ as long as $||\delta_D||_s \leq C_2$, which in light of Theorem 2.7.3 is the present situation. Instead, this section gives a variant of the Mizuno-Todd-Ye algorithm described in 2.2 that works for general convex optimisation and uses the primal-dual scaling machinery. If the choice of T^2 is invariant under interchange of primal and dual problems—for example, the operator geometric mean of the primal and dual integral scalings—this method is also primal-dual symmetric.

This section gives an explicit analysis of the following feasible-start primal-dual interior point algorithm; α and γ will be chosen to guarantee the desired iteration complexity bound:

Take $k := 0$ and $\left(x^{(0)}, y^{(0)}, s^{(0)}\right)$ to be feasible and central (we can also accept approximately central points).
while $\langle x^{(k)}, s^{(k)}\rangle > \epsilon\theta$ **do**
  Take T^2 as in Theorem A.2.3
  Select $\gamma \in [0, 1]$.
  Solve

$$\begin{pmatrix} 0 & A^\top & I \\ A & 0 & 0 \\ I & 0 & T^2 \end{pmatrix} \begin{pmatrix} d_x \\ d_y \\ d_s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -x^{(k)} - \gamma\mu F'_*(s) \end{pmatrix}.$$

  Select $\alpha_k \in [0, \infty)$.
  $\left(x^{(k+1)}, y^{(k+1)}, s^{(k+1)}\right) \leftarrow \left(x^{(k)}, y^{(k)}, s^{(k)}\right) + \alpha(d_x, d_y, d_s)$.
  $k \leftarrow k + 1$.
end while

Lemma 2.9.1. *Let $r_v := -v + \gamma\mu w$. Then, the system of equations in Line 3 of the above algorithm imply*

$$T^{-1}d_x = \operatorname{proj}(\ker(AT))r_v, \quad Td_s = \operatorname{proj}(\operatorname{im}(AT)^\top)r_v$$

where $\operatorname{proj}(S)$ *for a linear subspace S is orthogonal projection onto S.*

In particular, $||T^{-1}d_x|| \leq ||r_v||$ and $||Td_s|| \leq ||r_v||$.

Proof. The third equation ensures that $T^{-1}d_x + Td_s = r_v$. The first two equations imply that $T^{-1}d_x$ must lie in $\ker(AT)$ while Td_s must lie in $\operatorname{im}(AT)^\top$. Since these two linear spaces are orthogonal, the result follows. □

The following result, which proves linear convergence (i.e. to get a k-digit accurate result, use $O(k)$ iterations), does not hint at quadratic convergence (i.e. for a k-digit accurate result, use $O(\log k)$ iterations). However, a tighter analysis of the low-rank updates showing that the approximation error is linear in $||\delta_D||_s$ within the $\frac{1}{50}$-neighbourhood would suffice to establish quadratic convergence. This is not hard to do; the ingredients are already given above. Quadratic convergence of centring is not necessary to establish the desired complexity result.

Lemma 2.9.2. *Suppose $x^{(k)} \in \mathrm{int}(K)$ and $s^{(k)} \in \mathrm{int}(K^*)$ define a feasible solution. If $\gamma_k = 1$ and $\alpha_k = 1$ and $||\delta_D^k||_{s^{(k)}} \leq \frac{1}{50}$, then*

- *$Ax^{(k+1)} = b$ and $A^\top y^{(k+1)} + s^{(k+1)} = c$.*
- *$x^{(k+1)} \in \mathrm{int}(K)$ and $s^{(k+1)} \in \mathrm{int}(K^*)$.*
- *$||\delta_D^{k+1}||_{s^{(k+1)}} \leq 0.007533$.*
- *$\mu_{k+1} = \mu_k$.*

Proof. We drop the superscript k when speaking of the kth iterate in this proof. The system of linear equations that determine d_x, d_y, and d_s guarantee that $d_x \in \ker A$ and $d_s = -A^\top d_y$; since $Ax^{(k)} = b$ and $A^\top y^{(k)} + s^{(k)} = c$, it follows that $Ax^{(k+1)} = b$ and $A^\top y^{(k+1)} + s^{(k+1)} = c$.

Notice that, with this choice of γ,

$$T^{-1}d_x + Td_s = -\delta_v.$$

Since

$$||d_x||_x \leq \frac{\sqrt{1.192311}}{\sqrt{\mu}}||T^{-1}d_x|| \leq \frac{\sqrt{1.192311}}{\sqrt{\mu}}||\delta_v|| \leq 0.024226 < 1,$$

strict primal feasibility is retained. A similar argument shows that strict dual feasibility is retained.

By Taylor's theorem, there exists an $\bar{x}$ on the segment $[x, x + d_x]$ such that $F'(x + d_x) = F'(x) + F''(\bar{x})d_x$. We therefore compute

$$\begin{aligned} ||T(s + d_s + \mu F'(x + d_x))|| &= ||T(\delta_D + d_s + \mu F''(\bar{x})d_x)|| \\ &\leq ||\delta_v + Td_s + T^{-1}d_x|| + ||T(T^{-2} - \mu F''(\bar{x}))d_x||. \end{aligned}$$

The first term is, of course, zero. However, notice that, by the Dikin ellipsoid bound and Theorem 2.7.3, for every $z \in \mathbb{R}^d$,

$$\mu F''(\bar{x})[z, z] \leq 1.299692T^{-2}[z, z]$$

and, similarly,

$$\mu F''(\bar{x})[z, z] \geq 0.798562T^{-2}[z, z]$$

We therefore bound

$$\left|\left|T(T^{-2} - \mu F''(\bar{x}))d_x\right|\right| \leq 0.299692 \left|\left|T^{-1}d_x\right|\right| = 0.299692||\delta_v|| < 0.006527\sqrt{\mu},$$

which implies, by Lemma A.3.1 part (5), the advertised bound on the new $||\delta_D||_s$.

As observed earlier, μ is unchanged by a centring iteration. □

Lemma 2.9.3. *If $\gamma_k = 0$ and $\alpha_k = \frac{0.047464}{\sqrt{\theta}}$ and $||\delta_D^k||_{s^{(k)}} \leq 0.007533$, then*

- *$Ax^{(k+1)} = b$ and $A^\top y^{(k+1)} + s^{(k+1)} = c$;*
- *$x^{(k+1)} \in \text{int}(K)$ and $s^{(k+1)} \in \text{int}(K^*)$;*
- *$\mu_{k+1} \leq (1-\alpha)\mu_k$;*
- *$||\delta_D^{k+1}||_{s^{(k+1)}} \leq \frac{1}{50}$.*

Proof. We recall that $||\delta_D||_s \leq 0.007533$ means that $||\delta_v|| \leq 0.008202\sqrt{\mu}$. Notice that

$$||d_x||_x \leq \frac{1}{\sqrt{0.808093\mu}}||T^{-1}d_x|| \leq \frac{1}{\sqrt{0.808093\mu}}||\delta_v|| \leq 0.009125$$

Consequently, any step with $\alpha < \frac{1}{2\sqrt{\theta}}$ retains strict primal feasibility. A similar analysis (due to the primal-dual symmetry of our set-up) reveals that $||d_s||_s \leq 0.008931\sqrt{\theta}$ and hence the dual step retains strict dual feasibility for α similarly bounded. Notice that $||\alpha d_s||_s \leq 1/25$; this permits us to use Lemma A.3.1 part (6) later.

As expected, $\langle s(\alpha), x(\alpha)\rangle = (1-\alpha)\theta\mu$. This establishes the desired reduction in μ.

We compute

$$\begin{aligned}
\left|\left|T\delta_D^{k+1}\right|\right| &= ||T(s + \alpha d_s + \mu F'(x + \alpha d_x))|| \\
&= ||T(s + \alpha d_s + \mu F'(x) + \alpha\mu F''(\bar{x})d_x)|| \\
&\leq ||\delta_v|| + \alpha\,||Td_s + \mu TF''(\bar{x})d_x||\,.
\end{aligned}$$

Let us write

$$E := \mu F''(\bar{x}) - T^{-2}.$$

Then, by Theorem 2.7.3, for every $z \in \mathbb{R}^d$,

$$-0.808093T^{-2}[z,z] \leq E[z,z] \leq 1.192311T^{-2}[z,z].$$

We thus get an upper bound of

$$\begin{aligned}
\left|\left|T\delta_D^{k+1}\right|\right| &\leq ||\delta_v|| + \alpha\left|\left|v + Td_s + T^{-1}d_x\right|\right| + \alpha\,||TEd_x|| \\
&\leq ||\delta_v|| + 0 + 0.192311\alpha\,||v|| \\
&\leq 0.008202\sqrt{\mu} + 0.192311 \cdot 0.047464\sqrt{\mu} \\
&< 0.017330\sqrt{\mu}.
\end{aligned}$$

This implies, by Lemma A.3.1 part (6), that $\left|\left|\delta_D^{k+1}\right|\right|_{s^{(k+1)}} \leq \frac{1}{50}$, as desired. □

Corollary 2.9.4. *Starting from an initial feasible central point, one can alternately apply the predictor and corrector steps outlined from the last two lemmata and recover an algorithm that takes at most* $42\sqrt{\theta}$ *iterations to reduce* μ *by a factor of one-half. In particular, this gives an* $O\left(\sqrt{\theta}\ln\left(1/\epsilon\right)\right)$ *bound on the iteration complexity of the algorithm using this choice of* γ*.*

Chapter 3

Hyperbolic polynomials and hyperbolicity cones

Hyperbolic polynomials arose first in the context of partial differential equations—a homogeneous, linear PDE in d variables is hyperbolic if it can be written as

$$p\left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_d}\right) f(x_1, \ldots, x_d) = 0$$

where p is a hyperbolic polynomial. To each hyperbolic polynomial is associated a hyperbolicity cone. In the PDE context, there is a "causal" relationship between a solution's value at some point x and its value at points y such that $y - x$ lies in the hyperbolicity cone. A fundamental example of a hyperbolic PDE is the wave equation

$$\left(\sum_i \frac{\partial^2}{\partial x_i^2}\right) f(x,t) = \frac{\partial^2}{\partial t^2} f(x,t).$$

The underlying polynomial here is

$$p(t, x_1, \ldots, x_d) = t^2 - x_1^2 - \cdots - x_d^2,$$

and the resulting hyperbolicity cone (in direction $(1, 0, \ldots, 0)$) is the cone

$$\left\{(t, x_1, \ldots, x_d) : t \geq \sqrt{x_1^2 + \cdots + x_d^2}\right\}.$$

Refer to Atiyah, Bott, and Gårding [4] or Hörmander's book [37] for further details on the connection to partial differential equations.

Optimisation over the intersection of a hyperbolicity cone with an affine space generalises optimisation over symmetric cones. (Whether it is impossible or simply inefficient to formulate all such problems using a symmetric cone instead of a hyperbolicity cone remains an open problem, however; this is sometimes called the "generalised Lax conjecture.")

Güler [29] noted that hyperbolicity cones have a natural barrier, given by the negative logarithm of the underlying polynomial, and that this barrier enjoys some of the same regularity properties as self-scaled barriers.

Hyperbolic polynomials have also found use elsewhere in mathematics. Notably, Gurvits [32] proved a generalisation of van der Waerden's conjecture and of the Schrijver-Valiant conjecture. Also notably, hyperbolic polynomials play a role in the "method of interlacing families" of Marcus, Spielman, and Srivastava used to resolve the Kadison-Singer problem ([45, 46]; see also [8]).

This chapter discusses some of these connections and develops the theory needed for an interior-point method for hyperbolicity cones.

3.1 Definitions and basic theory

Definition 3.1.1. Let $p \in \mathbb{R}[x_1, x_2, \ldots, x_d]$ be a polynomial. p is said to be *homogeneous* if every term in p has the same total degree.

Definition 3.1.2. Let $p \in \mathbb{R}[x_1, x_2, \ldots, x_d]$ be a homogeneous polynomial. p is said to be *hyperbolic in direction* $e \in \mathbb{R}^d$ if

- $p(e) > 0$, and
- for every $x \in \mathbb{R}^d$, the univariate polynomial $t \mapsto p(x + te)$ only has real roots.

Here are some fundamental examples of hyperbolic polynomials:

- If $f \in (\mathbb{R}^d)^* \setminus \{0\}$ is a nonzero linear functional on $\mathbb{R}^d$, then $x \mapsto fx$ is hyperbolic in any direction e such that $fe > 0$.
- The polynomial $p(t, x_1, \ldots, x_{d-1}) = t^2 - x_1^2 - \cdots - x_{d-1}^2$ is hyperbolic in direction $(1, 0, \ldots, 0)$.
- The polynomial $p(X) = \det X$, defined on the space of symmetric $n \times n$ matrices, is hyperbolic in direction I.
- If p and q are both hyperbolic in direction e, then their product pq is hyperbolic in direction e.
- If p is hyperbolic in direction e, q divides p, and $q(e) > 0$, then q is also hyperbolic in direction e.

- E_k, the elementary symmetric polynomial of degree k, is hyperbolic in direction $(1, \ldots, 1)$ on $\mathbb{R}^d$ for every $d \geq k$. (The elementary symmetric polynomial of degree k has one monomial for every set of k variables; for example, $E_2(a, b, c) = ab + ac + bc$.)

Other interesting examples of hyperbolic polynomials will be considered later.

Definition 3.1.3. Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e. The *eigenvalues* of $x \in \mathbb{R}^d$ with respect to p and e are the roots of the univariate polynomial $t \mapsto p(x - te)$.

Consider the example of det on the symmetric matrices. The eigenvalues of a symmetric matrix M with respect to det and I are the usual eigenvalues of M from linear algebra. Eigenvalues with respect to a hyperbolic polynomial are thus a generalisation of eigenvalues of symmetric matrices.

Definition 3.1.4. Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e. The *hyperbolicity cone* of p in direction e, $\Lambda_+(p, e)$, is defined to be the set of points in $\mathbb{R}^d$ that have no negative eigenvalues with respect to p and e. Written another way:

$$\Lambda_+(p, e) = \{x \in \mathbb{R}^d : p(x + \lambda e) \geq 0 \ \ \forall \lambda \geq 0\}.$$

Its interior is given by

$$\Lambda_{++}(p, e) = \{x \in \mathbb{R}^d : p(x + \lambda e) > 0 \ \ \forall \lambda \geq 0\},$$

which is the set of points that have only positive eigenvalues with respect to p and e.

The next three theorems are fundamental. Proofs can be found in [72].

Theorem 3.1.5 (Gårding [19])**.** *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e. Let $e' \in \Lambda_{++}(p, e)$. Then p is also hyperbolic in direction e'.*

Theorem 3.1.6 (Gårding [19])**.** *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e. Then $\Lambda_+(p, e)$ is a closed convex cone and $\Lambda_{++}(p, e)$ is an open convex cone.*

Theorem 3.1.7 (Renegar [72])**.** *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e. Then every face of $\Lambda_+(p, e)$ is exposed.*

Definition 3.1.8. Let $HomP(n, d)$ be the set of homogeneous polynomials of degree n in d variables.

Let $HypP(n, d, e)$ be the set of homogeneous polynomials of degree n in d variables that are hyperbolic in direction e.

Let $SHypP(n, d, e)$ be the set of hyperbolic polynomials $p \in HypP(n, d, e)$ such that, for every $x \in \mathbb{R}^d$ not a multiple of e, $t \mapsto p(x + te)$ has no multiple roots.

The following theorem, due to Wim Nuij [68], gives some idea of what the set of hyperbolic polynomials looks like.

Theorem 3.1.9 (Nuij [68])**.** *Suppose $n > 1$ and $d > 1$ are integers.*

Then

- $SHypP(n, d, e) \subsetneq HypP(n, d, e) \subsetneq HomP(n, d)$.
- $HypP(n, d, e)$ *is a closed subset of* $HomP(n, d)$.
- $SHypP(n, d, e)$ *is an open subset of* $HomP(n, d)$.
- $HypP(n, d, e)$ *and* $SHypP(n, d, e)$ *are both connected.*
- $HypP(n, d, e)$ *and* $SHypP(n, d, e)$ *are both simply connected.*

Remark 3.1.10. This result implies, in particular, that the set of hyperbolic polynomials of degree n in d variables is a full-dimensional subset of the set of homogeneous polynomials of degree n in d variables—there is a $\binom{n+d-1}{d}$-parameter family of hyperbolic polynomials of degree n in d variables. Taking, for instance, $d = n$ and crudely bounding the binomial coefficient from below, one needs $2^{2n+1}/n$—exponentially many—variables to describe a hyperbolic polynomial of "size" n.

It is awkward to speak about "polynomial time" in the context of algorithms that take a hyperbolic polynomial as input. It is also somewhat awkward to speak formally about why this is the case. At the risk of cluttering this discussion with irrelevant details, there are a number of models of general computation amenable to various kinds of theoretical analysis that model aspects of practical computation. One example is the deterministic Turing machine model; see [81] for a formal definition. In the Turing machine model, the input is represented as a finite sequence of symbols drawn from a finite alphabet. In order to represent all hyperbolic polynomials of degree n in n variables with integer coefficients at most L in absolute value, one needs at least $4^{\Omega(n)} \log L$ symbols. The story is similar for various other models of computation, such as the bit model, the RAM model, the word-RAM model, and so forth. A modification of the RAM-type models is the "real RAM model" in which each memory location can hold a real number and each "instruction" in the program is an addition, a negation, a multiplication, or a solution to an algebraic equation of fixed degree. In this model, the obvious representation of a hyperbolic polynomial will again use $4^{\Omega(n)}$ real numbers, but the fact that the "information" that can be stored per cell is not bounded above may leave room for some clever trickery. Regardless, when one speaks about "polynomial time" in any of these models, one means that the time needed for an algorithm to run to completion is always bounded above by a polynomial function of the number of symbols in the input. The number of symbols used to write down a general hyperbolic polynomial of degree n in n variables, at least in models of computation in which each input symbol can be one of finitely many things, is exponentially large.

I will generally avoid this difficulty by not stating results in terms of overall running time bounds. I will instead use bounds on the number of iterations, or the number of function, gradient, and Hessian evaluations, considering the hyperbolic polynomial as a black box.

3.2 Further examples

An important construction of hyperbolic cones is given by directional differentiation of hyperbolic polynomials. Renegar worked out much of the theory of these derivative polynomials and their associated cones.

Theorem 3.2.1 (Renegar [72])**.** *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e and of degree at least 2. Then the polynomial p'_e given by*

$$p'_e(x) := (\nabla p)(x)[e]$$

is also hyperbolic in direction e.

Furthermore,

$$\Lambda_+(p'_e, e) \supseteq \Lambda_+(p, e).$$

Proof. Let $x \in \mathbb{R}^d$. Since the univariate polynomial $q(t) := p(x + te)$ only has real roots, its derivative dq/dt also only has real roots. Note that

$$p'_e(e) = (\nabla p)(e)[e] = \lim_{h \to 0}(p((1+h)e) - p(e))/h = \lim_{h \to 0}((1+h)^n - 1)/hp(e) = np(e) > 0$$

as well, so p'_e is hyperbolic in direction e.

If $x \in \Lambda_+(p, e)$, then the univariate polynomial $q(t) = p(x + te)$ only has nonpositive real roots. Thus, by interlacing of roots, its derivative $dq/dt = p'_e(x+te)$ also only has nonpositive real roots. Thus $x \in \Lambda_+(p'_e, e)$, proving the claimed containment. □

Remark 3.2.2. (I am not certain who first observed this, but I am certain that this is well-known. Work of Gurvits from 2004 [31] points out the particular reduction I give below, but it appears to leave the hardness result implicit.)

Write $\operatorname{per} M$ for the permanent of the square matrix M. Observe that

$$E_n(x_1, \ldots, x_n) = \frac{1}{n!} \operatorname{per} \begin{pmatrix} x_1 & x_2 & \ldots & x_n \\ x_1 & x_2 & \ldots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1 & x_2 & \ldots & x_n \end{pmatrix}.$$

If $v_1, \ldots, v_{n-1}$ are real n-vectors with only positive components, then

$$(\ldots((E_n(x_1, \ldots, x_n))'_{v_1})'_{v_2})\ldots)'_{v_{n-1}}$$
$$= \operatorname{per} \begin{pmatrix} v_{11} & v_{12} & \ldots & v_{1n} \\ v_{21} & v_{22} & \ldots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{(n-1)1} & v_{(n-1)2} & \ldots & v_{(n-1)n} \\ x_1 & x_2 & \ldots & x_n \end{pmatrix}.$$

This is a linear function that, when evaluated at a vector v_n, gives the permanent of the matrix formed by vertically concatenating the row vectors $v_1^T, \ldots, v_n^T$. Computing the permanent of a matrix is #P-hard, so it must also be #P-hard to compute the value of a high-order derivative polynomial of a hyperbolic polynomial when each derivative can be taken in an arbitrary direction inside the cone. (#P-hard means that, if the permanent of a matrix can be computed in polynomial time, then counting the number of solutions to any problem in NP can also be done in polynomial time. In particular, if the permanent can be computed in polynomial time, then P = NP [88].)

If all derivatives are taken in the same direction e and p can be computed exactly in polynomial time, the situation is different. Using finite differences, one can compute arbitrarily good approximations to the kth directional derivative of p in direction e using $k+1$ evaluations of p.

Renegar showed that, near every point of the boundary of a derivative cone $\Lambda_+(p'_e, e)$ outside $\Lambda_+(p, e)$, the boundary of the derivative cone has positive curvature in all directions but one:

Theorem 3.2.3 ([72]). *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e and of degree at least 2. Suppose $\Lambda_+(p, e)$ is pointed and has nonempty interior. Let $x \in \operatorname{bd} \Lambda_+(p'_e, e) \setminus \Lambda_+(p, e)$ be such that $(\nabla p'_e)(x) \neq 0$ and let $h \perp (\nabla p'_e)(x)$. Then either h is a scalar multiple of x or $(Hp'_e)(x)[h, h] < 0$.*

The following result apparently first appeared in my Master's thesis; it shows that every nontrivial hyperbolicity cone is the intersection of its derivatives.

Theorem 3.2.4 ([55]). *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e and of degree at least 2. Then*

$$\Lambda_+(p, e) = \cap_{f \in \Lambda_{++}(p,e)} \Lambda_+(p'_f, e).$$

Other very interesting hyperbolic polynomials are the bases generating polynomials of certain matroids—those with the "half-plane property" [13] [6]. These polynomials have one variable for each element of the matroid's ground set and one monomial for each basis and they are hyperbolic in any direction in the positive orthant. One polynomial from this class is an especially simple counterexample to a nontrivial generalisation of the Lax conjecture, also given below.

3.3 The Lax conjecture and generalisations

Helton and Vinnikov [36] proved the Lax conjecture [42]—that every hyperbolic polynomial in three variables arises as a three-dimensional slice of the determinant on symmetric matrices. More formally, if $p(x, y, z)$ has degree n and is hyperbolic in direction $(1, 0, 0)$, then there exist symmetric $n \times n$ matrices A and B such that $p(x, y, z) = \det(xI + yA + zB)$. Lewis, Parrilo, and Ramana [43] give a detailed exposition of the connection between the Lax conjecture and the result proved by Helton and Vinnikov.

There are a number of natural generalisations of the Lax conjecture to more than three variables. Some of those listed below have been called *the* generalised Lax conjecture by various authors.

- Conjecture: Let $d \geq 2$ be an integer. If p is a polynomial in d variables that is hyperbolic in direction $(1, 0, \ldots, 0)$, then there exist symmetric matrices $A_2, \ldots, A_d$ such that
$$p(x_1, \ldots, x_d) = \det(x_1 I + x_2 A_2 + \ldots x_d A_d).$$
This one is particularly easy to disprove. Both sides must be homogeneous of degree $\deg p$, so the matrices on the right-hand side must be $d \times d$ symmetric matrices. By 3.1.9, there is a $\binom{2d}{d} = 2^{\Omega(d)}$-dimensional family of hyperbolic polynomials of degree d in d variables. However, there are only $O(d^3)$ parameters on the right-hand side. Thus, there is a $d \geq 4$ and a degree-d polynomial in d variables that cannot be written in the desired form.

- Conjecture: Let $d \geq 2$ be an integer. If p is a polynomial in d variables that is hyperbolic in direction $(1, 0, \ldots, 0)$, then there exist symmetric matrices $A_2, \ldots, A_d$ such that
$$p(1, x_2, \ldots, x_d) = \det(I + x_2 A_2 + \ldots x_d A_d).$$
Helton and Vinnikov [36] advanced this conjecture. Here, the dimension argument from the previous conjecture does not directly work. The matrices $A_2, \ldots, A_d$ must have rank $\deg p$, but need not themselves be operators on a d-dimensional space. However, Brandén [7] was able to show that, if such a representation exists, then there is a representation in which $A_2, \ldots, A_d$ are all symmetric $\deg p \times \deg p$ matrices. Brandén then appeals to the same dimension argument as before to disprove this conjecture.

- Conjecture: Let $d \geq 2$ be an integer. If p is a polynomial in d variables that is hyperbolic in direction $(1, 0, \ldots, 0)$, then there is a power K and symmetric matrices $A_2, \ldots, A_d$ such that
$$(p(x_1, \ldots, x_d))^K = \det(x_1 I + x_2 A_2 + \ldots x_d A_d). \tag{3.1}$$

Brändén [7] proved that the bases generating polynomial of the Vámos cube gives a counterexample to this conjecture. Below is a sketch of the argument.

Define

$$V_8(a,b,c,d,e,f,g,h) \\ = E_4(a,b,c,d,e,f,g,h) - abcd - abef - abfh - cdef - efgh. \quad (3.2)$$

V_8 is the bases generating polynomial of the Vámos cube. Wagner and Wei [92] proved that V_8 is hyperbolic in direction $(1,1,\ldots,1)^T$.

A theorem of Gurvits [31] states that, if p is hyperbolic in direction e and $v_1,\ldots,v_k$ are vectors in $\Lambda_+(p,e)$, then the function $r : 2^{\{1,\ldots,k\}} \to \mathbb{N}$ giving, at S, the degree of $t \mapsto p(e + \sum_{i\in S} v_i)$, is the rank function of a polymatroid.

When p is a power of V_8, taking $v_1,\ldots,v_8$ to be the eight standard basis vectors of $\mathbb{R}^8$ makes r above a multiple of the rank function of the Vámos matroid.

Brändén shows that, if K and $A_1,\ldots,A_d$ as in (3.1) exist for a polynomial, then, no matter the choice of vectors $v_1,\ldots,v_k$, the rank function r must be the rank function of a real-representable polymatroid. No multiple of the Vámos matroid's rank function is the rank function of a real-representable polymatroid, however, therefore no power of V_8 admits a determinantal representation.

- Conjecture: Let $d \geq 2$ be an integer. If p is a polynomial in d variables that is hyperbolic in direction $(1,0,\ldots,0)$, then there is a hyperbolic polynomial q in d variables and symmetric $A_2,\ldots,A_d$ such that

$$q(x_1,\ldots,x_d)p(x_1,\ldots,x_d) = \det(x_1 I + x_2 A_2 + \ldots x_d A_d).$$

  I believe this is still open. If one restricts q to have $\Lambda_+(q,e) \supseteq \Lambda_+(p,e)$, this is also still open and it implies the next conjecture.

- Conjecture: Let $d \geq 2$ be an integer. If p is a polynomial in d variables that is hyperbolic in direction $(1,0,\ldots,0)$, then there is a dimension D and an affine subspace A of the symmetric $D \times D$ real matrices such that $\Lambda_+(p,e)$ is linearly isomorphic to $\{M \in A : M \succeq 0\}$.

  A number of authors call this "*the* generalised Lax conjecture." It is still open. However, using results of Helton and Nie [35], Netzer and Sanyal [64] proved that every hyperbolicity cone that (apart from 0) only has smooth boundary points is a projection of a slice of a semidefinite cone.

- Conjecture: Let $d \geq 2$ be an integer. If p is a polynomial in d variables that is hyperbolic in direction $(1,0,\ldots,0)$, then there is a dimension D and an affine subspace A of the symmetric $D\times D$ real matrices such that $\Lambda_+(p,e)$ is equal to a linear projection of $\{M \in A : M \succeq 0\}$.

This is weaker than the conjecture from the previous bullet since it allows a linear projection. It has been proven in a special case; Netzer and Sanyal [64] have proven it to be true whenever p belongs in the set $SHypP$ from Theorem 3.1.9.

It does not appear that one can directly "take a limit of" Netzer and Sanyal's result to prove this conjecture. In their construction, the dimension D is always finite, but it depends on p and they do not bound it in terms of d.

The bases generating polynomial of the Vámos matroid may serve as an interesting test case for those of the above conjectures that have not already been proven false. I do not know an explicit determinantal representation of any nonzero multiple of V_8 or an explicit representation of its hyperbolicity cone as either a slice of, or a projection of a slice of, a semidefinite cone. I also do not know a finite procedure that will either find such a representation or prove that none exists. In this vein, however, Netzer and Thom [65] state a sufficient condition using Clifford algebras for a hyperbolicity cone to have a semidefinite representation.

It is not clear that a constructive, positive resolution to any of the conjectures above would render obsolete all special-purpose hyperbolic optimisation algorithms. The huge dimension of the space of hyperbolic polynomials means that most hyperbolic polynomials of a given size cannot be represented concisely by any program, but also that the smallest-dimensional determinantal representation of most hyperbolic polynomials is exponentially large. If the set of hyperbolic polynomials that can be evaluated in polynomial time on a computer contains polynomials whose smallest determinantal representation is exponentially large, then special-purpose hyperbolic optimisation algorithms can offer an exponential speedup over approaches via determinantal representation and semidefinite optimisation.

3.4 Analytic properties of hyperbolic polynomials

The geometric mean of the eigenvalues is a concave function on the Hermitian positive definite matrices. Gårding proved a direct generalisation of this to hyperbolic polynomials:

Lemma 3.4.1. *(Gårding [19]) Let $f(x) = c(x - r_1)(x - r_2)\dots(x - r_k)$ be a univariate polynomial with only real roots, ordering the roots by $r_1 \leq r_2 \leq \cdots \leq r_k$. Then $|f(x)|^{1/k}$ is concave on (r_i, r_{i+1}) for each $i \in \{1, \dots, k-1\}$.*

Proof. Fix i. Assume without loss of generality that $f(x)$ is positive on (r_i, r_{i+1}). Fix an $x \in (r_i, r_{i+1})$ and let v be the k-vector whose ith component is $1/(x - r_i)$. Note that

$$f'(x) = \left(\sum_i \frac{1}{x - r_i} \right) f(x) = (e^T v) f(x).$$

Also note that

$$f''(x) = \left(\sum_i \sum_{j \neq i} \frac{1}{x - r_i}(x - r_j) \right) f(x) = (v^T(ee^T - I)v)f(x).$$

We compute

$$\begin{aligned} (f^{1/n})''(x) &= \frac{f^{1/n}(x)}{nf^2(x)} \left(\frac{f'^2(x)}{n} + f''(x)f(x) - f'^2(x) \right) \\ &= \frac{f^{1/n}(x)}{n^2 f^2(x)} \left((1-n)f'^2(x) + nf''(x)f(x) \right). \end{aligned}$$

Note that

$$\begin{aligned} (1-n)f'^2(x) + nf''(x)f(x) &= f^2(x) \left((1-n)v^T ee^T v + nv^T(ee^T - I)v \right) \\ &= f^2(x)v^T \left(ee^T - nI \right) v. \end{aligned}$$

The $n \times n$ matrix $ee^T - nI$ is negative semidefinite, having one eigenvalue 0 and $(n-1)$ eigenvalues $-n$. Thus $(f^{1/n})''(x)$ exists and is nonpositive for every x in (r_i, r_{i+1}); concavity follows. □

Theorem 3.4.2 (Gårding [19])**.** *If $p \in \mathbb{R}[x_1, \ldots, x_d]$ is hyperbolic in direction e and p has degree n, then $p^{1/n}$ is concave on $\Lambda_{++}(p, e)$.*

Proof. Let x and y lie in $\Lambda_{++}(p, e)$. Consider the function $f(\lambda) = p(x + \lambda(y - x))$. Then f is a univariate polynomial of degree n with only real roots, none of which are in $[0, 1]$. From the previous lemma, it follows that $f^{1/n}$ is concave on a neighbourhood of $[0, 1]$, from which the result follows. □

Definition 3.4.3. Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e. Define the *hyperbolic barrier* on $\Lambda_{++}(p, e)$ by

$$F(x) = -\log(p(x)).$$

The following result is straightforward and it gives some meaning to the directional derivatives of F in terms of the zeroes of p:

Theorem 3.4.4. *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e of degree n. Define $F : \Lambda_{++}(p, e) \to \mathbb{R}$ by $F(x) = -\log p(x)$. Let $x \in \Lambda_{++}(p, e)$ and let $h \in \mathbb{R}^d$. Then there exists a real number C and nonzero real numbers $r_1, \ldots, r_n$ such that*

$$F(x + th) = C - \sum_{i=1}^{n} \log(t - r_i)$$

and, for every integer $a \geq 1$,

$$F^{(a)}(x)[h, \ldots, h] = (-1)^a (a-1)! \sum_{i=1}^{n} (-r_i)^{-a}.$$

Proof. Take $r_1, \ldots, r_n$ to be the roots of $t \mapsto p(x + th)$. The rest follows by definition of F, differentiation, and substitution. □

If p has degree n, then F is n-logarithmically homogeneous. Further, if $\{x_i\}_{i=1}^{\infty}$ is a sequence drawn from $\Lambda_{++}(p, e)$ that converges to a boundary point of $\Lambda_{++}(p, e)$, then $F(x_i) \to \infty$. The following theorem proves (in the special case of $a = 2$ and $b = 3$) that F is convex on $\Lambda_{++}(p, e)$ and self-concordant:

Theorem 3.4.5 (Güler [29], Theorem 4.1)**.** *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e. Define $F : \Lambda_{++}(p, e) \to \mathbb{R}$ by $F(x) = -\log p(x)$. For every $1 \leq a < b$ such that a and b are integer and a is even, the following inequalities hold for every $x \in \Lambda_{++}(p, e)$ and $h \in \mathbb{R}^d$:*

$$F^{(a)}(x)[h, \ldots, h] \geq 0$$

$$\left(\frac{F^{(a)}(x)[h, \ldots, h]}{(a-1)!}\right)^{1/a} \geq \left(\frac{\left|F^{(b)}(x)[h, \ldots, h]\right|}{(b-1)!}\right)^{1/b}.$$

Proof. There are reals $r_1, \ldots, r_n$ such that

$$F^{(a)}(x)[h, \ldots, h] = (a-1)! \sum_{i=1}^{n} (-r_i)^{-a}.$$

Since a is even, every term in the summation is positive; this establishes the first inequality.

The second inequality reads

$$\left(\sum_{i=1}^{n} (-r_i)^{-a}\right)^{1/a} \geq \left(\left|\sum_{i=1}^{n} (-r_i)^{-b}\right|\right)^{1/b}.$$

Taking $s_i = |1/r_i|$,

$$\left(\sum_{i=1}^{n} (-r_i)^{-a}\right)^{1/a} = \left(\sum_{i=1}^{n} s_i^a\right)^{1/a} \geq \left(\sum_{i=1}^{n} s_i^b\right)^{1/b} \geq \left(\left|\sum_{i=1}^{n} (-r_i)^{-b}\right|\right)^{1/b}.$$

The first $\geq$ follows because $||v||_p \leq ||v||_q$ whenever $p > q$; the second $\geq$ is the triangle inequality for real numbers. □

Corollary 3.4.6 (Güler [29])**.** *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e of degree n. Define $F : \Lambda_{++}(p, e) \to \mathbb{R}$ by $F(x) = -\log p(x)$. Then F is a n-logarithmically-homogeneous self-concordant barrier for $\Lambda_{++}(p, e)$.*

The following result is the "long-step Hessian estimation property" for hyperbolic barriers. It will be very useful in Chapter 4.

All of the proofs below use the Helton-Vinnikov theorem. However, all of these results were known before the Helton-Vinnikov theorem and Güler wrote proofs in [29] that do not use the Helton-Vinnikov theorem.

Theorem 3.4.7 (Güler [29], Theorem 6.1)**.** *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e of degree n. Define $F : \Lambda_{++}(p, e) \to \mathbb{R}$ by $F(x) = -\log p(x)$. Let $y \in \Lambda_{++}(p, e)$. Then $x \mapsto -F'(x)[y]$ is a convex function of $x \in \Lambda_{++}(p, e)$.*

Proof. Fix a direction $z \in \mathbb{R}^d$. Using the Helton-Vinnikov theorem, write

$$p(x + \alpha z + \beta y) = K \det(I + \alpha A + \beta B).$$

Here, B is positive definite, since $ty \in \Lambda_{++}(p, e)$ for every $t > 0$.

The polynomial $t \mapsto p(x + tz)$ has $\deg p$ real roots for almost all choices of z. What follows proves that $t \mapsto F'(x + tz)[y]$ is convex for every such z. Since $x \mapsto F'(x)[y]$ is $\mathcal{C}^2$, convexity everywhere easily follows.

Compute

$$F''(x + ty)[z, z] = \operatorname{Tr}\left(((I + tB)A^{-1})^2\right) = \operatorname{Tr}(A^{-2}) + 2t \operatorname{Tr}(A^{-1}BA^{-1}) + t^2 \operatorname{Tr}(BA^{-1}BA^{-1}).$$

Thus

$$\begin{aligned} F'''(x)[y, z, z] &= \lim_{t \to 0} \frac{2t \operatorname{Tr}(A^{-1}BA^{-1}) + t^2 \operatorname{Tr}(BA^{-1}BA^{-1})}{t} \\ &= 2 \operatorname{Tr}(A^{-1}BA^{-1}) \\ &= 2 \operatorname{Tr}((B^{1/2}A^{-1})^T (B^{1/2}A^{-1})) > 0. \end{aligned}$$

From this the desired result follows. □

Theorem 3.4.8 (Güler [29], Theorem 7.1)**.** *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e of degree n. Define $F : \Lambda_{++}(p, e) \to \mathbb{R}$ by $F(x) = -\log p(x)$. Let $x \in \Lambda_{++}(p, e)$ and let $h \in \mathbb{R}^d$ be such that $x + h \in \Lambda_{++}(p, e)$. Let*

$$\sigma_x(h) = \frac{1}{\max\{t : x - th \in \Lambda_+(p, e)\}}.$$

Then, for every $z \in \mathbb{R}^d$,

$$F''(x)[z, z] \leq (1 + \sigma_x(h))^2 F''(x + h)[z, z]$$

and

$$F''(x)[z, z] \geq (1 - \sigma_x(-h))^2 F''(x + h)[z, z].$$

Proof. Fix a direction $z \in \mathbb{R}^d$. Using the Helton-Vinnikov theorem, write

$$p(x + \alpha z + \beta h) = K \det(I + \alpha A + \beta B).$$

Then

$$F''(x)[z, z] = \operatorname{Tr}(A^{-2}).$$

Compute

$$F''(x + h)[z, z] = \operatorname{Tr}\left(((I + B)A^{-1})^2\right) \geq (1 + \lambda_{min}(B))^2 \operatorname{Tr}(A^{-2})$$

and

$$F''(x + h)[z, z] = \operatorname{Tr}\left(((I + B)A^{-1})^2\right) \leq (1 - \lambda_{min}(-B))^2 \operatorname{Tr}(A^{-2}).$$

Note that $\lambda_{min}(B) = \sigma_x(h)$ and $\lambda_{min}(-B) = \sigma_x(-h)$. The result follows. □

The following theorem shows that hyperbolic barrier Hessians map the primal cone *into* its dual.

Theorem 3.4.9 (Güler [29])**.** *Let $p \in \mathbb{R}[x_1, \ldots, x_d]$ be hyperbolic in direction e of degree n. Define $F : \Lambda_{++}(p, e) \to \mathbb{R}$ by $F(x) = -\log p(x)$. Let x and y lie in $\Lambda_{++}(p, e)$. Then $F''(x)[y]$ lies in the interior $\operatorname{int} \Lambda_{++}(p, e)^*$ of the dual cone.*

Proof. If K is a pointed convex cone with interior, then $s \in \operatorname{int} K^*$ if and only if $\langle x, s\rangle > 0$ for every $x \in K \setminus \{0\}$. Thus, fix $z \in \Lambda_{++}(p, e)$. Using the Helton-Vinnikov theorem, write

$$p(x + \alpha y + \beta z) = K \det(I + \alpha A + \beta B).$$

Both A and B are positive definite since $t \mapsto p(x + ty)$ and $t \mapsto p(x + tz)$ each have $\deg p$ roots, all of which are negative. Compute

$$F''(x)[y, z] = \left(\frac{\partial}{\partial \beta} \operatorname{Tr}((I + \beta B)^{-1} A)\right)_{\beta=0} = \operatorname{Tr} BA > 0.$$

The desired result follows. □

3.5 Approximating the primal integral scaling by numerical integration

There is occasion to compute, via numerical integration,

$$T_P^2 = \int_{-1}^{1} F''(x + ty)dt,$$

where $x - y$ and $x + y$ are points in $\Lambda_{++}(p, e)$. It is desirable to have a good bound on some operator norm of the approximation error in terms of the length of y in some norm. The length of y is a measure of centrality, so it will be bounded above by some constant less than 1. In exchange for bounding the length of y above, a constant approximation error bound is desired. The purpose of this section is to give, for various (constant) bounds on the measure of centrality and various numerical integration schemes, corresponding bounds on the worst-case approximation error.

Consider using a classical numerical integration formula of the form

$$\int_{-1}^{1} F''(x + ty)dt \approxeq \sum_{i=1}^{n} w_i F''(x + a_i y).$$

If the relative approximation error in every direction h is bounded—that is,

$$\left| \frac{\sum_{i=1}^{n} w_i F''(x + a_i y)[h, h]}{T_P^2[h, h]} - 1 \right| < \epsilon$$

for some $\epsilon > 0$ not depending on h, then the bound, for every $z \in \mathbb{R}^d$,

$$(1 - \epsilon)T_P^2[z, z] \leq \sum_{i=1}^{n} w_i F''(x + a_i y)[z, z] \leq (1 + \epsilon)T_P^2[z, z]$$

follows.

By the Helton-Vinnikov theorem, there are $n \times n$ matrices A and B such that

$$p(x + ty + sh) = p(x) \det (I - tA + sB)$$

and, for every $z \in \mathbb{R}^n \setminus \{0\}$, $|A[z, z]| \leq ||z||_2^2$. I may further assume via an orthogonal change of basis that A is diagonal; write a_i for the ith entry of A's diagonal. It happens that the eigenvalues of A are related to the error in approximating T_P^2. Then

$$F''(x + ty)[h, h] = \operatorname{Tr} \left(B(I - tA)^{-1} B(I - tA)^{-1} \right) = \sum_{i,j} \frac{B_{ij}^2}{(1 - ta_i)(1 - ta_j)}$$

Thus

$$\int_{-1}^{1} F''(x + ty)[h, h]dt = \sum_{i,j} B_{ij}^2 \int_{-1}^{1} \frac{1}{(1 - ta_i)(1 - ta_j)} dt.$$

Notice that each integrand is positive on $[-1, 1]$, hence that each summand is nonnegative, and therefore it is sufficient to show that the chosen quadrature formula is good at computing integrals of the form

$$\int_{-1}^{1} \frac{1}{(1 - ta_i)(1 - ta_j)} dt$$

$\max \lvert a_i \rvert = C$	Simpson error S	5-point error U
0.33	0.009	10^{-4}
0.5	0.056	0.002
0.67	0.244	0.006
0.75	0.483	0.009
0.9	2.31	0.114
0.95	6	0.591

Table 3.1: Worst-case approximation error for Simpson's rule and the 5-point Clenshaw-Curtis rule when integrating $(1 - a_i t)^{-1}(1 - a_j t)^{-1}$ for various bounds on $|a_i|$ and $|a_j|$.

where $|a_i| < 1 - \delta$ and $|a_j| < 1 - \delta$. If $a_i = a_j$, this integral is exactly

$$\frac{2}{1 - a_i^2}.$$

Otherwise, this integral is exactly

$$\frac{\log \frac{1+a_i}{1-a_i} - \log \frac{1+a_j}{1-a_j}}{a_i - a_j}.$$

Simpson's rule for approximating T_P^2 is

$$S_P^2 := \frac{1}{6}\left(F''(x) + 4F''(x + \delta_P/2) + F''(x + \delta_P)\right).$$

The 5-point Clenshaw-Curtis rule [14] for approximating T_P^2 is

$$U_P^2 := \frac{1}{30}\begin{pmatrix} F''(x) + 8F''(x + (1 - 1/\sqrt{2})\delta_P/2) + 12F''(x + \delta_P/2) \\ +8F''(x + (1 + 1/\sqrt{2})\delta_P/2 + F''(x + \delta_P) \end{pmatrix}.$$

Substituting $f(t) = \frac{1}{(1-xa_i)(1-xa_j)}$ and the expressions for these two quadrature rules, one can compute the following upper bounds on the relative error for Simpson's rule and the 5-point Clenshaw-Curtis rule. See Table 3.1.

In the context of approximating the primal integral scaling, the above discussion constitutes a proof of the following theorem. Note that the norm used to bound δ_P is the cone norm $||\cdot||_{K,x}$ rather than ellipsoidal norm $||\cdot||_x$.

Theorem 3.5.1. *Let p be a polynomial of degree ω hyperbolic in direction e. Let $K = \Lambda_+(p, e)$. Let x and $\tilde{x}$ be interior points of K. Let $\mu = -F'(\tilde{x})[x]/\omega$. Let $\delta_P = \mu\tilde{x} - x$. Let S_P^2, T_P^2, and U_P^2 be as above.*

Suppose $||\delta_P||_{K,x} \leq C$, where C appears in the first column of Table 3.1. Let S be the corresponding number in the second column and U the corresponding number in the third column. Then, for every $h \in \mathbb{R}^d$,

$$(1 - S)T_P^2[h, h] \leq S_P^2[h, h] \leq (1 + S)T_P^2[h, h]$$

and

$$(1-U)T_P^2[h,h] \leq U_P^2[h,h] \leq (1+U)T_P^2[h,h].$$

Having computed the necessary table, some remarks on Clenshaw-Curtis quadrature in general are in order.

First, as the name implies, the 5-point Clenshaw-Curtis rule is one of a family of quadrature rules due to Clenshaw and Curtis [14].

Second, the weights used in Clenshaw-Curtis quadrature are not necessarily optimal for the class of functions being integrated even if one fixes the nodes. I did not pursue this further since the weights given by both Simpson's rule and the 5-point rule give good enough approximation error.

Third, one can interpret Clenshaw-Curtis quadrature on the integrand $f(x)$ in terms of the Fourier coefficients of $f(\cos(\omega))$. Writing

$$\int_{-1}^{1} f(x)dx = \int_{0}^{\pi} f(\cos\omega)\sin\omega d\omega$$

and $f \circ \cos$ in terms of its cosine series as

$$f(\cos\omega) = \sum_{k=0}^{\infty} a_k \cos(k\omega),$$

one can compute

$$\int_{-1}^{1} f(x)dx = \sum_{k=0}^{\infty} \frac{2a_{2k}}{1-4k^2}.$$

Clenshaw-Curtis quadrature computes, when N is even,

$$\int_{-1}^{1} f(x)dx \approxeq \sum_{\ell=0}^{\infty} a_{2N\ell} + \frac{2}{1-N^2}a_{2N\ell+N} + \sum_{k=1}^{N/2-1} \frac{2}{1-4k^2}\left(a_{2N\ell+2k} + a_{2N\ell+2N-2k}\right).$$

For example, when $N = 4$, corresponding to the 5-point rule, the coefficients in front of a_{2k} are 1, $-1/3$, $-1/15$, $-1/3$, 1, $-1/3$, $-1/15, \cdots$ rather than the more desirable 1, $-1/3$, $-1/15$, $-1/35$, $-1/63$, $-1/99, \cdots$.

It is probably possible to work out good bounds on the Fourier coefficients of

$$\frac{1}{(1-a_i\cos\omega)(1-a_j\cos\omega)}$$

and thereby get a more general analysis of the dependence of the worst-case approximation error on the distance to centrality. I leave this to future work.

Chapter 4

Interior-point methods for hyperbolicity cone optimisation

This chapter describes a primal-dual interior-point method that may be effective for minimising a linear functional over the intersection of an affine space with a hyperbolicity cone. The implementation referenced is available at
`https://csclub.uwaterloo.ca/~tmyklebu/general.tar.gz`.

In this setting, in general, there is no explicit access to the dual barrier; algorithms are needed for approximating the dual barrier at some dual point s and for computing a dual step length. (However, the machinery of hyperbolicity cone optimisation may still be useful in cases where the dual barrier can be computed and represented.)

Other authors have proposed interior-point methods that work, at least in theory, in this setting. Nesterov and Nemirovskii [58] proposed and analysed the worst-case behaviour of several methods that apply to general convex optimisation, some of which do not need any access to the dual barrier.

Nesterov [57] proposed a nonsymmetric primal-dual interior-point method that makes use of the dual barrier only when trying to take a long step, and only requires low-precision computation of the value of the dual barrier. This method alternates between primal affine-scaling steps and "phases" of several centring steps. The centring phase computes a scaling point $w \in \operatorname{int} K$ such that $F''(w) \in \mathcal{T}_0^2(x, s)$. (There are examples of hyperbolicity cones where such a w need not exist if one asks for $F''(w) \in \mathcal{T}_1^2(x, s)$. The 5-dimensional Vinberg cone mentioned in Chapter 2 is one example.)

Skajaa, Jørgensen, and Hansen [82] implemented this method of Nesterov using a homogeneous self-dual formulation like that presented in Section 5.7. They report iteration counts and Cholesky counts on a variety of linear optimisation problems, entropy optimisation problems, geometric optimisation problems, and p-norm optimisation problems, compar-

ing against the commercial package MOSEK. Skajaa, Jørgensen, and Hansen incorporate a couple of interesting heuristics into their code. They use a sequence of BFGS updates to accelerate the centring phases and to compute a "second-order" search direction similar in spirit to Mehrotra's corrector.

Building on work by Renegar [73], Renegar and Sondjaja [74] give an affine-scaling method for hyperbolicity cone optimisation and analyse its worst-case iteration complexity. I am not aware of any computational experiments that used this algorithm.

In contrast to the work of Nesterov and of Renegar and Sondjaja, the method described in this chapter makes heavier use of the dual barrier. It is an infeasible-start variant of the Mizuno-Todd-Ye method.

4.1 Representing a hyperbolic polynomial p

It is necessary to represent the hyperbolic polynomial of interest in some fashion. My implementation represents it as a "straight-line program", which is a rooted directed acyclic graph in which each node corresponds either to a binary operation (addition or multiplication), negation, evaluation of a variable, or a constant.

This is a fairly natural way to represent polynomials on a computer. Each node of the graph represents either a number or the result of some computation. The name "straight-line program" comes from their natural expression as computer programs with no conditionals or control-flow constructs. One can order the nodes of the straight-line program such that no node "reads from" nodes ordered later than it, then one can assign each node a name, then one can translate the graph into a computer program.

Given a polynomial in this form, it is straightforward to compute the polynomial's value, gradient, and Hessian at any specific point.[1]

It is also straightforward to compute, given a point x and a direction d, the coefficients of the univariate (in t) polynomial given by $p(x+td)$. (Computing the coefficients in a numerically stable way is tricky, however, and my implementation does no such thing. Furthermore, getting the smallest positive root from the coefficient representation of a polynomial is a famously ill-conditioned problem. See [95].)

This is not the only possible representation, and there are many reasons to use representations other than this one. It is not at all clear that every hyperbolic polynomial that can be computed quickly—whatever that is taken to mean—can be represented compactly in this

[1] It is less straightforward to compute them rapidly, and my implementation does no such thing. I use the "forward mode of automatic differentiation" in a naive way. The "reverse mode of automatic differentiation" can yield gradients and Hessian-vector products I defer to Griewank and Walther's book [27] for definitions, descriptions, and analyses of the above terms related to automatic differentiation.

$\hat{x}_0 \leftarrow \hat{x}$
$H \leftarrow [F''(\hat{x}_0)]^{-1}$
loop
  $r \leftarrow F'(\hat{x}) + s$
  if $r^T H r < \epsilon^2$ **then return** $\hat{x}$
  end if
  $N \leftarrow -Hr$
  $\hat{x} \leftarrow \hat{x} + N$
end loop

Figure 4.1: An algorithm for refining an initial guess $\hat{x}$ at $-F'_*(s)$ until $F'(\hat{x})$ is within ϵ of s in the local norm at the initial $\hat{x}$.

form. For some of those that can be represented compactly—say, using polynomially many vertices—as straight-line programs, it appears that one might lose efficiency by doing so.

Determinants and Pfaffians (and their linear slices) can be represented "compactly" as straight-line programs. A survey paper by Rote [76] gives constructions of straight-line programs for the $n \times n$ determinant and $(2n) \times (2n)$ Pfaffian that have $O(n^4)$ nodes. Thus slices of the determinant polynomial can be represented "compactly" as straight-line programs if it is necessary to do so. Kaltofen [39] used asymptotically fast matrix multiplication algorithms to derive smaller straight-line programs for the $n \times n$ determinant—however, the number of nodes is still worse than cubic.

As mentioned in Remark 3.2.2, the permanent of a general $n \times n$ matrix is #P-hard to compute. The analogue for permanents of cofactor expansion yields a straight-line program with $(n-1)(2^n - 1)$ nodes for computing the permanent of an $n \times n$ matrix. I am not aware of any better algorithms for computing the permanent exactly.

4.2 Evaluating the dual barrier

In hyperbolicity cone optimisation, one may not be able to access the dual barrier directly. Depending on the relative costs of evaluating the gradient and Hessian of F, one can either evaluate the dual barrier arbitrarily closely or design an algorithm that does relatively few gradient evaluations. This section discusses how to approximate the dual barrier using the primal barrier should one choose to do so.

If one can compute F and F' much more quickly than the Hessian F'', the algorithm in Figure 4.1 computes an $\hat{x}$ such that $-F'(\hat{x})$ is close to s. "Close" is measured in the local norm at $\hat{x}_0$, and the algorithm terminates once $F'(\hat{x})$ is within ϵ of s in this norm.

It is not necessary to form an explicit inverse when computing H. All of the computations

in Figure 4.1 involving H can be done using a Cholesky factorisation of $F''(\hat{x}_0)$ instead.

4.3 Finding a primal step length

Given a primal iterate x and a primal search direction d_x, it is desirable to compute the largest t such that $x + td_x \in \Lambda_+(p, e)$—or, if such a t is substantially larger than 1, to know that a full step may be taken safely. This is equivalent to finding the smallest positive root of the univariate slice of p given by $t \mapsto p(x + td_x)$.

One could work out that univariate slice of p in some form and then compute its smallest positive root. I tried this, but (probably owing to my choice of the coefficient representation for the univariate polynomial) I got wildly inaccurate values for the smallest positive root.

Thus, in my implementation, I keep track of a t such that $x+td_x$ lies in $\Lambda_{++}(p, e)$. I initialise this t at zero, then I repeatedly update $t \leftarrow t+0.95/||d_x||_{x+td_x}$. By the Dikin ellipsoid bound, this update preserves the property that $x + td_x \in \Lambda_{++}(p, e)$.

J. Renegar suggested during my defence to exploit homogeneity here; one can find the smallest positive root of $p(x + td)$ by instead finding the largest positive root of $p(d + tx)$. Numerically, this is a much easier problem. I have not yet tried this.

4.4 Finding a dual step length

One can compute a lower bound on the longest admissible dual step using the primal Hessian at $\hat{x}$—a step almost to the boundary of the Dikin ellipsoid can never leave the dual cone. However, the short steps that result lead to slow convergence when close to the optimal solution.

By Theorem 3.4.9, the primal barrier Hessian anywhere in $\operatorname{int} K$ maps the primal cone into the dual cone. This also holds for nonnegative linear combinations of primal barrier Hessians. Therefore, if H is some nonnegative linear combination of primal barrier Hessians and $H^{-1}s + tH^{-1}d_s \in K$, then $s + td_s \in K^*$.

My implementation makes use of this fact, taking $H = F''(\hat{x})$ where $-F'(\hat{x}) = s$. An alternative would be to take H to be an approximation to the primal integral scaling T_P^2, but I have not evaluated this.

Interestingly, this algorithm is good enough to take near-unit steps when close to optimality on the randomly-generated problems reported on later in this chapter.

4.5 An algorithm

The algorithm in Figure 4.2 is an adaptation of the Mizuno-Todd-Ye algorithm to hyperbolicity cone optimisation. Modulo issues introduced by finite-precision arithmetic, it is a fairly faithful representation of how the implementation reported on in Section 4.6 works.

This is an *infinite* algorithm—it generates an infinite sequence of iterates and the convergence properties of this sequence are of interest. In practice, this algorithm is modified so that it does terminate. Termination criteria are typically designed around the desired properties of the solution to be returned and the quality of solution the algorithm, as implemented on a computer with inexact arithmetic, is capable of producing. In the experiments reported later, the algorithm was terminated as soon as either Cholesky factorisation failed or the algorithm of Figure 4.1 failed numerically.[2]

An added wrinkle in this algorithm is the line $s \leftarrow -F'(\hat{x})$. This will in general render the iterate dual-infeasible; it perturbs s by an amount bounded above by the error at termination of the algorithm in Figure 4.1.

In my implementation, the primal-dual scaling chosen is the Simpson's rule approximation to the primal integral scaling

$$\frac{1}{6}\left(F''(x) + F''(\hat{x}) + 4F''\left(\frac{x+\hat{x}}{2}\right)\right),$$

with the rank-four update of Theorem 2.6.1.

4.6 Computational results

It is interesting to examine the convergence of this method on specific hyperbolicity cone optimisation problems. To this end, I generated a number of strictly-feasible problems at random. I fixed the objective function c as the all-ones vector. I constructed a random matrix A with some number of rows, taking each entry to be uniformly distributed between 0 and 1. The right-hand side b was chosen so that $Ae = b$, where e is the all-ones vector.

In each case, I ran the code until an error occurred; typically, this happened because a negative diagonal entry was encountered during Cholesky factorisation. Each row except the first of each table represents an iteration. The first column reports what kind of step was taken; either "a" for an affine-scaling step or "c" for a corrector step. The second and

[2] In linear optimisation, sufficiently accurate approximately-optimal solutions can be used to generate exact optimal solutions. How to do this is not discussed in any detail in this thesis, but see Megiddo [48] for a "crossover" algorithm or Vavasis and Ye [90] for a modified interior-point method for linear optimisation that provably terminates in finitely many iterations.

$x \leftarrow \bar{e}$, $s \leftarrow \bar{e}$, $y \leftarrow 0$.
loop
  Use the algorithm in Figure 4.1 to compute $\hat{x} \approxeq -F'_*(s)$.
  $s \leftarrow -F'(\hat{x})$.
  $\delta_D \leftarrow s + \mu F'(x)$.
  $\mu \leftarrow s^T x/\theta$.
  $\beta \leftarrow (F''(\hat{x}))^{-1}[\delta_D, \delta_D]$.
  $T^2 \leftarrow$ a primal-dual scaling in $\mathcal{T}_1(x, s)$.
  if $\beta < \frac{1}{4}$ **then**
    Compute $(d_x^{aff}, d_y^{aff}, d_s^{aff})$ such that

$$\begin{array}{rrcl} Ad_x^{aff} & & = & b - Ax \\ A^T d_y^{aff} & +d_s^{aff} & = & c - A^T y - s \\ d_x^{aff} & +T^2 d_s^{aff} & = & -x. \end{array}$$

    Find the longest primal step length α_P such that $x + \alpha_P d_x^{aff}$ remains in K.
    Find an admissible dual step length α_D as in Section 4.4.
    $\alpha \leftarrow \min(\alpha_P, \alpha_D)$.
    do
      $\alpha \leftarrow 0.9\alpha$.
      $(x', y', s') \leftarrow (x, y, s) + \alpha(d_x^{aff}, d_y^{aff}, d_s^{aff})$.
      Compute $\beta(x', s')$.
    while $\beta(x', s') > \frac{1}{2}$
  else
    Compute $(d_x^{cor}, d_y^{cor}, d_s^{cor})$ such that

$$\begin{array}{rrcl} Ad_x^{cor} & & = & 0 \\ A^T d_y^{cor} & +d_s^{cor} & = & 0 \\ d_x^{cor} & +T^2 d_s^{cor} & = & -x + \mu\hat{x}. \end{array}$$

    $(x', y', s') \leftarrow (x, y, s) + (d_x^{cor}, d_y^{cor}, d_s^{cor})$.
  end if
  $(x, y, s) \leftarrow (x', y', s')$.
end loop

Figure 4.2: An infeasible-start predictor-corrector method adapted to hyperbolicity cone optimisation.

third columns are self-explanatory. α is the length of the step just taken. α_P^{max} is the length of the longest primal step possible in the direction just used. α_D^{max} is the length of the admissible dual step computed. The last three columns give the primal infeasibility, dual infeasibility, and duality gap.

The polynomials used were the elementary symmetric function E_k in n variables, which is the sum of all squarefree monomials of degree k made from the n variables, the Vamos polynomial, which is $E_4(a,b,c,d,e,f,g,h) - aebf - bfcg - cgdh - aedh - aecg$, and the Vamos-like quartic polynomials of Burton, Vinzant, and Youm [10] in more variables of the form

$$E_4(x_1,\ldots,x_n,y_1,\ldots,y_n) - \left(\sum_{i=2}^{n} x_1y_1x_iy_i\right) - \left(\sum_{i=2}^{n-1} x_{i-1}y_{i-1}x_iy_i\right).$$

Observe the algorithm's behaviour when μ is very small. In all cases, the algorithm is also able to take very long (i.e. very close to length-one) affine-scaling steps in both primal and dual.

step	$\log_{10}\mu$	β	α	α_P^{max}	α_D^{max}	$-\log_{10}\|\|r_P\|\|$	$-\log_{10}\|\|r_D\|\|$	$-\log_{10}\|r_G\|$
0	0.426	0.000	0.000	0.000	0.000	inf	inf	-1.6
a	0.106	0.392	0.522	2.000	0.773	26.3	16.6	-1.3
c	0.106	0.012	1.000	2.000	2.000	26.4	16.5	-1.3
a	-0.191	0.344	0.494	1.165	0.732	26.1	12.4	-1.0
c	-0.191	0.011	1.000	2.000	2.000	26.0	12.4	-1.0
a	-0.494	0.288	0.503	0.846	0.745	26.0	12.8	-0.7
c	-0.494	0.008	1.000	2.000	2.000	25.9	12.7	-0.7
a	-0.782	0.324	0.485	0.718	0.802	25.9	13.3	-0.4
c	-0.782	0.007	1.000	2.000	2.000	25.7	13.3	-0.4
a	-1.058	0.340	0.470	0.696	0.852	25.6	10.9	-0.1
c	-1.058	0.008	1.000	2.000	2.000	25.5	10.9	-0.1
a	-1.349	0.323	0.489	0.724	0.863	25.4	11.5	0.2
c	-1.349	0.007	1.000	2.000	2.000	25.3	10.1	0.2
a	-1.671	0.281	0.523	0.775	0.865	25.2	10.8	0.5
c	-1.671	0.006	1.000	2.000	2.000	25.1	10.2	0.5
a	-2.031	0.217	0.563	0.834	0.886	25.0	11.0	0.9
a	-2.431	0.399	0.603	0.893	0.924	24.9	11.8	1.3
c	-2.431	0.011	1.000	2.000	2.000	24.7	9.9	1.3
a	-3.262	0.415	0.852	0.947	0.959	24.9	11.6	2.1
c	-3.262	0.002	1.000	2.000	2.000	24.8	11.6	2.1
a	-4.227	0.077	0.892	0.991	0.992	24.8	13.5	3.1
a	-5.199	0.086	0.893	0.999	0.993	24.3	15.5	4.0
a	-6.171	0.088	0.893	1.000	0.993	21.5	17.5	5.0
a	-7.144	0.091	0.893	1.000	0.993	19.5	19.6	6.0
a	-8.115	0.098	0.893	1.000	0.993	18.2	17.2	6.9

Table 4.1: E_{15} in 40 variables, 15 random linear equality constraints

step	$\log_{10}\mu$	β	α	α_P^{max}	α_D^{max}	$-\log_{10}\|\|r_P\|\|$	$-\log_{10}\|\|r_D\|\|$	$-\log_{10}\|r_G\|$
0	0.125	0.000	0.000	0.000	0.000	inf	inf	-1.6
a	-0.214	0.490	0.542	2.000	0.803	26.8	21.2	-1.3
c	-0.214	0.015	1.000	2.000	2.000	26.7	18.7	-1.3
a	-0.473	0.443	0.448	1.397	0.700	26.4	17.6	-1.0
c	-0.473	0.017	1.000	2.000	2.000	26.4	16.0	-1.0
a	-0.708	0.442	0.418	1.043	0.653	26.2	16.3	-0.8
c	-0.708	0.017	1.000	2.000	2.000	26.1	13.3	-0.8
a	-0.941	0.396	0.416	0.897	0.682	26.0	13.5	-0.5
c	-0.941	0.011	1.000	2.000	2.000	25.9	13.3	-0.5
a	-1.174	0.347	0.416	0.835	0.728	25.6	13.8	-0.3
c	-1.174	0.009	1.000	2.000	2.000	25.5	13.8	-0.3
a	-1.417	0.362	0.428	0.809	0.737	25.2	14.2	-0.1
c	-1.417	0.009	1.000	2.000	2.000	24.9	14.2	-0.1
a	-1.651	0.332	0.417	0.786	0.752	24.6	14.7	0.2
c	-1.651	0.006	1.000	2.000	2.000	24.4	14.7	0.2
a	-1.882	0.307	0.413	0.797	0.775	24.2	15.2	0.4
c	-1.882	0.007	1.000	2.000	2.000	24.1	15.2	0.4
a	-2.170	0.438	0.484	0.833	0.755	23.8	15.7	0.7
c	-2.170	0.011	1.000	2.000	2.000	23.7	15.5	0.7
a	-2.489	0.442	0.521	0.812	0.823	23.4	16.1	1.0
c	-2.489	0.016	1.000	2.000	2.000	23.3	16.0	1.0
a	-2.869	0.464	0.583	0.873	0.864	23.2	16.6	1.4
c	-2.869	0.012	1.000	2.000	2.000	23.1	16.3	1.4
a	-3.278	0.385	0.610	0.905	0.904	23.0	17.1	1.8
c	-3.278	0.009	1.000	2.000	2.000	22.9	16.9	1.8
a	-3.717	0.238	0.636	0.949	0.942	22.8	16.9	2.2
a	-4.178	0.351	0.654	0.978	0.969	22.7	17.7	2.7
c	-4.178	0.001	1.000	2.000	2.000	22.7	17.7	2.7
a	-5.135	0.230	0.889	0.990	0.988	22.7	19.5	3.7
a	-6.109	0.258	0.894	0.993	0.993	22.7	21.2	4.6
c	-6.109	0.000	1.000	2.000	2.000	22.7	21.2	4.6
a	-7.083	0.003	0.894	0.993	0.993	22.7	23.0	5.6
a	-8.058	0.003	0.894	0.993	0.993	22.7	24.8	6.6

Table 4.2: E_{30} in 40 variables, 15 random linear equality constraints

step	$\log \mu$	β	α	α_P^{max}	α_D^{max}	$-\log_{10} \|\|r_P\|\|$	$\log_{10} \|\|r_D\|\|$	$\log_{10} \|r_G\|$
0	0.301	0.046	0.000	0.000	0.000	inf	inf	-0.9
a	-0.039	0.275	0.542	2.000	0.804	27.2	14.7	-0.6
c	-0.039	0.012	1.000	2.000	2.000	27.4	14.5	-0.6
a	-0.370	0.222	0.534	1.480	0.791	27.0	13.6	-0.2
a	-0.728	0.386	0.561	1.098	0.831	27.1	14.2	0.1
c	-0.728	0.014	1.000	2.000	2.000	27.3	14.2	0.1
a	-1.460	0.234	0.815	0.990	0.905	27.2	15.7	0.9
a	-2.161	0.378	0.801	0.975	0.976	26.9	16.9	1.6
c	-2.161	0.015	1.000	2.000	2.000	25.9	11.5	1.6
a	-3.138	0.048	0.895	0.995	0.994	26.9	13.5	2.5
a	-4.135	0.061	0.899	0.999	0.999	27.0	15.5	3.5
a	-5.134	0.081	0.900	1.000	1.000	24.5	17.5	4.5
a	-6.131	0.123	0.899	0.999	1.000	23.1	19.5	5.5
a	-7.124	0.202	0.898	0.998	0.999	20.4	18.9	6.5
a	-8.104	0.353	0.895	0.995	0.998	19.0	17.8	7.5
c	-8.105	0.014	1.000	2.000	2.000	18.6	17.5	7.5

Table 4.3: Vamos polynomial with 5 random linear equality constraints

step	$\log \mu$	β	α	α_P^{max}	α_D^{max}	$-\log_{10} \|\|r_P\|\|$	$\log_{10} \|\|r_D\|\|$	$\log_{10} \|r_G\|$
0	0.301	0.046	0.000	0.000	0.000	inf	inf	-0.9
a	0.010	0.257	0.488	1.152	0.724	27.4	19.8	-0.6
c	0.010	0.009	1.000	2.000	2.000	27.7	13.3	-0.6
a	-0.318	0.231	0.530	0.966	0.786	27.1	13.9	-0.3
a	-0.601	0.418	0.479	0.853	0.820	26.8	14.5	-0.0
c	-0.601	0.012	1.000	2.000	2.000	26.4	14.4	-0.0
a	-0.941	0.291	0.542	0.804	0.916	26.2	15.0	0.3
c	-0.941	0.009	1.000	2.000	2.000	26.0	14.8	0.3
a	-1.303	0.222	0.565	0.837	0.972	25.8	15.3	0.7
a	-1.697	0.423	0.597	0.885	0.948	25.7	16.0	1.1
c	-1.697	0.012	1.000	2.000	2.000	25.5	15.8	1.1
a	-2.402	0.335	0.803	0.939	0.954	25.7	17.2	1.8
c	-2.402	0.006	1.000	2.000	2.000	25.6	17.2	1.8
a	-3.327	0.177	0.881	0.980	0.979	25.8	19.0	2.7
a	-4.313	0.220	0.897	0.997	0.996	26.0	20.2	3.7
a	-5.311	0.263	0.899	0.999	0.999	25.3	20.3	4.7
c	-5.311	0.004	1.000	2.000	2.000	24.5	12.6	4.7
a	-6.309	0.006	0.900	1.000	1.000	22.9	14.5	5.7
a	-7.306	0.011	0.899	1.000	0.999	21.3	17.5	6.7
a	-8.314	0.029	0.902	1.002	1.003	18.3	15.8	7.7

Table 4.4: Vamos polynomial with 2 random linear equality constraints

step	$\log_{10}\mu$	β	α	α_P^{max}	α_D^{max}	$-\log_{10}\|\|r_P\|\|$	$-\log_{10}\|\|r_D\|\|$	$-\log_{10}\|r_G\|$
0	1.000	0.000	0.000	0.000	0.000	inf	inf	-1.6
a	0.508	0.389	0.678	1.103	0.753	25.5	15.3	-1.1
c	0.508	0.015	1.000	2.000	2.000	25.7	11.4	-1.1
a	0.113	0.144	0.597	0.884	0.891	25.7	12.2	-0.7
a	-0.296	0.291	0.611	0.905	0.940	25.9	12.3	-0.3
c	-0.296	0.006	1.000	2.000	2.000	25.8	12.3	-0.3
a	-1.141	0.394	0.857	0.952	0.974	26.0	13.9	0.5
c	-1.141	0.003	1.000	2.000	2.000	26.2	13.9	0.5
a	-2.110	0.075	0.893	0.992	0.996	25.8	15.8	1.5
a	-3.107	0.084	0.899	0.999	0.999	24.1	17.7	2.5
a	-4.106	0.087	0.900	1.000	1.000	22.6	19.6	3.5
a	-5.106	0.093	0.900	1.000	1.000	20.9	19.9	4.5
a	-6.105	0.108	0.900	1.000	1.000	18.3	18.2	5.5
a	-7.105	0.145	0.900	1.000	1.001	17.0	16.7	6.5

Table 4.5: Vamos-like polynomial in 40 variables with 10 random linear equality constraints

Chapter 5

Implementation of an interior-point method for linear optimisation

5.1 Introduction

The purpose of this chapter is to describe the techniques used in my implementation of an interior-point method for linear optimisation as well as some other techniques that may be interesting. My implementation can be found at `https://www.github.com/tmyklebu/lp`.

The heavy focus on implementation details, tricks, and heuristics in this chapter may seem to represent a significant break in philosophy from the earlier chapters. It does not. In my view, the purpose of studying algorithms for a particular class of problem is that someone might someday wish to solve problems from that class, and that someone might want to be as well-equipped to do so as possible. Tricks and heuristics are fair game in that context, and documentation of those implementation details that have shown themselves effective in practice is invaluable.

There are no available solvers tailored to hyperbolicity cone optimisation in general. There are few available solvers that use self-concordant barriers to handle general conic constraints. There are, however, successful packages, such as SeDuMi [83] and SDPT3 [87] for symmetric cone programming that make special use of self-scaled barriers. Linear optimisation is the best-understood class of optimisation problems for which interior-point methods have been successful. Several excellent software packages are available for solving linear optimisation problems using interior-point methods.

The technology for linear optimisation seems to be mature; these excellent software packages are all capable of solving broadly similar sets of problems in broadly similar amounts of time. The bulk of this chapter is spent discussing some of the techniques used in modern linear optimisation solvers.

If one is proposing, as I am in Sections 5.4.4 and 5.8, a new technique in linear optimisation, that technique must be evaluated with respect to a sensible baseline. A comparison showing a reduction in iteration count against a solver that is using the same step lengths in the primal and dual may not show anything interesting at all; the new technique when applied to a better solver might show an iteration count regression or no improvement. Likewise, a computational comparison showing a modest reduction in overall processing time against a solver whose Cholesky factorisation is very slow is not very hard to make.

This chapter studies algorithms for solving the linear optimisation problem

$$\begin{array}{rl} \min & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0. \end{array} \tag{5.1}$$

Here, A, b, and c are vectors of machine-representable numbers. In many applications, A is quite sparse.

The dual of this problem is

$$\begin{array}{rl} \max & b^T y \\ \text{subject to} & A^T y + s = c \\ & s \geq 0. \end{array} \tag{5.2}$$

A problem is considered solved when, for some notion of "close to", the following conditions hold:

- $x \geq 0$ and $s \geq 0$.
- Ax is close to b.
- $A^T y + s$ is close to c.
- $c^T x$ is close to $b^T y$.

The inequalities $x > 0$ and $s > 0$ will hold by construction, but the other three conditions make up a termination condition.

At each iteration, we solve linear systems of the form

$$\begin{array}{rrcl} Ad_x & & = & r_1 \\ A^T d_y & +d_s & = & r_2 \\ d_x & +T^2 d_s & = & r_3. \end{array} \tag{5.3}$$

The matrix T^2 will always be symmetric and positive definite. T^2 is traditionally chosen to be XS^{-1}. There is considerable variety in the choices of r_1, r_2, and r_3. One may solve several systems of this form with the same T^2 but different right-hand sides.

5.2 Solving the linear system

The survey paper [1] by Andersen, Gondzio, Mészáros and Xu on implementing interior-point methods for LP devotes a substantial amount of attention to linear algebra issues.

5.2.1 The Cholesky approach

The approach described in this subsection is frequently called the "normal equations" approach.

One approach to solving (5.3) is to find d_y first; one can solve $AT^2A^Td_y = AT^2r_2 - Ar_3 + r_1$ for d_y, then compute $d_s = r_2 - A^Td_y$, then compute $d_x = r_3 - T^2d_s$.

The matrix AT^2A^T is symmetric and positive semidefinite. If A has full row rank, it is positive definite. A Cholesky factorisation therefore exists; in principle, we can write $AT^2A^T = LL^T$ where L is lower-triangular. Provided $r_1 \in \operatorname{im} A$, one can then find some d_y that works, from which one can compute d_s and d_x.

If T^2 is chosen to be a diagonal matrix, as is traditionally done, and every column of A has few nonzero entries, AT^2A^T will often be sparse. It turns out that the number of nonzeros in the Cholesky factor L depends not only on the sparsity pattern of AT^2A^T, but also on the "variable ordering"—the ordering of the rows and columns of AT^2A^T; the number of nonzeros in a Cholesky factor of $PAT^2A^TP^T$ for some permutation matrix P can vary quite substantially with P. There is no numerical advantage to reordering rows and columns in Cholesky factorisation, so the variable ordering can be chosen heuristically to reduce the number of nonzeros in L.

Practically successful techniques for choosing a variable ordering (P in the last paragraph) exist and have been implemented. The book [20] by George and Liu provides an accessible introduction to the problem of solving large sparse positive definite systems by Cholesky factorisation. The book [15] by Davis is more recent and also covers methods for nonsymmetric systems. The module CHOLMOD [12] in the software package SuiteSparse is a state-of-the-art, freely-available and freely-usable implementation of sparse Cholesky factorisation.

Thanks to accumulated roundoff error, Cholesky factorisation may compute a negative or zero diagonal element of which it would like the square root. Wright [97] showed that zeroing every row of the Cholesky factor where a small or negative number is found on the diagonal and setting the corresponding entry on the right-hand side to zero when solving linear systems results in useful steps as long as the threshold chosen is small and the undisturbed part of AT^2A^T is reasonably-conditioned. More concretely, while the computed value of d_y in this approach may differ considerably from the true d_y, the components of the computed d_s corresponding to small components of s are accurate and the components of the computed

d_x corresponding to small components of x are accurate.

The following table should give some sense of the relative amounts of time required to compute a sparse Cholesky factorisation, to solve one linear system using that Cholesky factorisation, and to solve four linear systems with the same left-hand side simultaneously on a modern (as of 2015) computer.

This computer has an Intel Xeon E3-1245v2 (Sandy Bridge) processor with eight logical cores and four physical cores, though no experiment reported in this thesis used more than one core. This computer has 32GB of memory. Each physical core has a 32kB 8-way associative first-level data cache, a 32kB 8-way associative first-level instruction cache, and a 256kB 8-way associative second-level cache. There is also an 8MB 16-way associative third-level cache. Every cache line has 64 bytes. This computer runs Linux 3.16.0, and all code was compiled using gcc 4.9.2.

prob name	m	nnz	chol flops	bksv flops	chol time	bksv time	bksv4 time
G3_circuit	1585478	4.623e6	6.225e10	9.939e7	4.220	0.1019	0.2338
af_5_k101	503625	9.027e6	6.229e10	9.882e7	3.572	0.0822	0.1626
apache2	715176	2.767e6	1.865e11	1.378e8	9.092	0.1126	0.2349
bone010	986703	1.244e7	4.980e11	3.752e8	23.704	0.2838	0.5664
consph	83334	3.047e6	1.298e11	6.833e7	5.616	0.0469	0.0925
inline_1	503712	1.866e7	1.418e11	1.722e8	7.580	0.1335	0.2621
ldoor	952203	2.374e7	7.393e10	1.431e8	4.808	0.1230	0.2478
nd12k	36000	7.128e6	4.886e11	1.146e8	20.700	0.0760	0.1546
nd24k	72000	1.439e7	2.181e12	3.258e8	88.640	0.2199	0.4430
nd6k	18000	3.458e6	1.149e11	4.050e7	5.192	0.0270	0.0552
offshore	259789	2.251e6	1.034e11	8.335e7	5.116	0.0652	0.1293
ship_003	121728	4.104e6	8.195e10	6.090e7	3.800	0.0431	0.0875

Here, m is the number of rows (and columns) of the matrix, nnz is the number of nonzeros, chol means Cholesky factorisation, bksv means backsolve (i.e. solving $Lz = r$ for z given an m-vector r), and bksv4 means four parallel backsolves (i.e. solving $LZ = R$ for Z given an $m \times 4$ matrix R).

Notice that the ratio of "time" to "flops" is substantially—around a factor 15—higher for backsolves. Notice also that the time to do four parallel backsolves is substantially smaller than four times the time needed for a single backsolve. These numbers are machine-dependent, but the patterns noted above appear for several fundamental reasons, one of which is that the number of memory operations needed in a backsolve is larger than the number of floating-point operations, while it is typically substantially smaller for Cholesky factorisation. This is also one reason why the time to do four parallel backsolves is not four times the time needed to do a single backsolve.

For these reasons, I will refrain from simply counting flops, or counting matrix factorisations and backsolves, to estimate the costs and benefits of a technique—they cannot substitute for a measurement of the real computational cost involved.

The approach outlined above has some disadvantages. For one, if A has a single column without any zero entries, then AT^2A^T will be fully dense. A number of techniques have been proposed for handling this difficulty.

- Column splitting (Vanderbei [89]): If a variable x_i appears in a large number of rows, create several variables $x_i^1, \ldots, x_i^k$. In each row containing x_i, replace x_i with one of the variables $x_i^1, \ldots, x_i^k$, and add the equations $x_i^1 = x_i^2, \ldots, x_i^{k-1} = x_i^k$ to the system $Ax = b$.

- Partitioning columns: Partition the columns of A into "sparse" and "dense" columns. if S is the sparse part and D is the dense part, then $AT^2A^T = ST^2S^T + DT^2D^T$. Provided ST^2S^T is nonsingular, one can compute solutions to $AT^2A^Td_y = rhs$ using a Cholesky factorisation of ST^2S^T and the Sherman-Morrison-Woodbury formula. A variant is to add a small diagonal regulariser to ST^2S^T before factorisation and use iterative refinement.

- If ST^2S^T is singular, Andersen [2] proposed adding a diagonal regulariser EE^T to ST^2S^T "on the fly" during factorisation, adding a diagonal entry to EE^T whenever a small diagonal entry arises during Cholesky factorisation.

- Product-form Cholesky (Goldfarb and Scheinberg [21]): If one has an LDL^T factorisation of a positive definite matrix A (where L is unit lower-triangular and D is diagonal), then $A + vv^T = L(D + L^{-1}vv^TL^{-T})L^T$. One can get an LDL^T factorisation of $D + L^{-1}vv^TL^{-T}$; the triangular factor, say L', has the form

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ p_2\beta_1 & 1 & 0 & \cdots & 0 \\ p_3\beta_1 & p_3\beta_2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_n\beta_1 & p_n\beta_2 & p_n\beta_3 & \cdots & 1 \end{pmatrix}, \tag{5.4}$$

  where p and β are vectors of dimension n. Goldfarb and Scheinberg suggested partitioning A into sparse and dense columns, forming an LDL^T factorisation of $A_{dense}T^2A_{dense}^T$, working out $V := L^{-1}A_{sparse}$, and computing a Cholesky factorisation, as a product of matrices of the form (5.4), of $D + VV^T$. For formulae for computing the vectors p and β, see [21].

  I will note that both backsolve using matrices of the form (5.4) and multiplication in sequence by by several matrices of the form (5.4) are amenable to blocking and vectorisation.

A large number of practically successful linear optimisation problem solvers, such as CPLEX, BPMPD, and the `linprog` routine in MATLAB use the Cholesky factorisation approach together with some handling of dense columns. My code uses Cholesky factorisation, but it does not incorporate any tricks for handling dense columns.

5.2.2 Symmetric indefinite factorisation

Instead of finding d_y, then d_s, then d_x, one can instead solve a symmetric indefinite system of the form

$$\begin{array}{cccc} -T^{-2}d_x & A^T d_y & = & r_2 - T^{-2}r_3 \\ Ad_x & & = & r_1 \end{array}$$

using a factorisation of the form LDL^T, where L is unit lower-triangular and D is diagonal (but indefinite).

Note that this system has a number of nonzeros on the same order as the number of nonzeros in A.

Proceeding by elimination of the first n variables and equations leads to the Cholesky approach from the previous section and the dense column problem associated with it. In some sense, therefore, one can do at least as well using symmetric indefinite factorisation as with Cholesky.

Symmetric indefinite factorisation has notable complications. What follows is an illustration of this due to Bunch and Parlett [9]. In the example

$$M = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

there do not exist lower-triangular L and diagonal D such that $M = LDL^T$. If one replaces the 0's on the diagonal with some small δ, there is the factorisation

$$\begin{pmatrix} \delta & 1 \\ 1 & \delta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1/\delta & 1 \end{pmatrix} \begin{pmatrix} \delta & 0 \\ 0 & \delta - 1/\delta \end{pmatrix} \begin{pmatrix} 1 & 1/\delta \\ 0 & 1 \end{pmatrix}.$$

Consider solving systems using this factorisation in floating-point arithmetic. (In floating-point arithmetic, if one ignores underflow and overflow conditions as I will throughout this example, the difference between a computed addition, subtraction, multiplication, or division and the actual result r can be bounded above by r times the "machine epsilon" ϵ_{mach}.) For simplicity, consider a decimal floating-point system with six significant digits and take

$\delta = 10^{-4}$. Then we compute, for the right-hand side $(1, 0)^T$,

$$\begin{aligned}
& \begin{pmatrix} 0.0001 & 1 \\ 1 & 0.0001 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\
= & \begin{pmatrix} 1 & 10000 \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0.0001 & 0 \\ 0 & -10000 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 0 \\ 10000 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\
= & \begin{pmatrix} 1 & 10000 \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0.0001 & 0 \\ 0 & -10000 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ -10000 \end{pmatrix} \\
=_{FP} & \begin{pmatrix} 1 & 1000 \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 10000 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.
\end{aligned}$$

The correct result is roughly $(-10^{-4}, 1)^T$; solving the system using this factorisation lost about two digits of accuracy.

Bunch and Parlett [9] proposed a factorisation for dense symmetric indefinite matrices. This factorisation is of the form LDL^T, where L is lower triangular with unit diagonal and there exists a permutation matrix P such that PDP^T is block-diagonal with 1×1 and 2×2 blocks.

The 2×2 pivots in Bunch-Parlett factorisation mean that, unlike in the Cholesky approach, the variable ordering may need to change during factorisation for numerical reasons.

Sparse variants of Bunch-Parlett factorisation exist. These must balance the need to minimise fill-in during factorisation with the need to obtain a useful factorisation. The PARDISO project [78] develops and distributes a suite of direct symmetric indefinite solvers, and, to my knowledge, PARDISO represents the state of the art in symmetric indefinite factorisation. PARDISO's user manual, at the time of this writing, states

> The coefficient matrix is perturbed whenever numerically acceptable pivots cannot be found within a diagonal supernode block. One or two passes of iterative refinement may be required to correct the effect of the perturbations.

To my knowledge, there is currently no freely-available, freely-usable, general-purpose implementation of sparse symmetric indefinite factorisation of similar speed and quality to CHOLMOD. However, the BPMPD package of Mészáros [50] uses this approach for some LPs. A paper by Maros and Mészáros [47] describes the indefinite factorisation used in BPMPD.

5.2.3 Iterative methods

When confronted with a large, sparse, positive definite system such as $AT^2A^Td_y = rhs$, my first impulse is to try solving it by the conjugate gradient method. Conjugate gradient

needs a good preconditioner in order to converge quickly, and it is not clear what that preconditioner should be.

Gonzalez-Lima, Wei, and Wolkowicz [26] use a preconditioned LSQR iteration to find search directions, guessing the optimal partition using Tapia indicators in order to compute the preconditioner. They report an improvement in running time on some randomly-generated nondegenerate problems. Gondzio [24] uses a partial Cholesky decomposition as a preconditioner for the conjugate gradient method, reporting success in a comparison against HOPDM on some large linear and quadratic problems.

One idea that may be fruitful is to use a Cholesky factorisation of $AT_{old}^2 A^T$, where T_{old}^2 is from some past iteration, as a preconditioner. I have not yet tried this, and I hope to soon, but I speculate that it is advantageous on larger problems on which the Cholesky factorisation is substantially more expensive (say, a factor 50 or more) than a backsolve.

5.3 Controlling step length in theory

Mizuno, Todd, and Ye [52] developed a predictor-corrector algorithm that takes at most $O(\sqrt{n}\log(1/\epsilon))$ iterations, generating iterates in neighbourhoods of the form

$$N_2(\beta_{max}) = \left\{(x,s) \in \mathbb{R}^n_{++} \times \mathbb{R}^n_{++} : ||Xs - \mu e||_2 \leq \beta_{max}\right\}.$$

They also analysed the worst-case behaviour of similar algorithms that use the neighbourhood

$$N_\infty(\beta_{max}) = \left\{(x,s) \in \mathbb{R}^n_{++} \times \mathbb{R}^n_{++} : ||Xs - \mu e||_\infty \leq \beta_{max}\right\}$$

or the "wide neighbourhood"

$$N_\infty^-(\beta_{max}) = \left\{(x,s) \in \mathbb{R}^n_{++} \times \mathbb{R}^n_{++} : Xs \geq (1 - \beta_{max}\mu e\right\},$$

showing that the worst-case behaviour of their algorithm when adapted to N_∞ or N_∞^- is worse. Interestingly, they also provided a heuristic analysis of their algorithms; they show that, under a probabilistic assumption that may not be justified, their algorithm based on N_2 takes about $n^{1/4}\log(1/\epsilon)$ iterations, while their algorithm based on N_∞^- takes about $\log(n)\log(1/\epsilon)$ iterations.

5.4 Heuristic directions

5.4.1 Mehrotra's corrector

The following power series argument is heuristic. It is the $\nu = 2$ case of a more general argument due to Monteiro, Adler, and Resende [54].

Consider the ordinary differential equations defined by the affine scaling direction:

$$\begin{aligned} A\frac{dx}{dt} &= 0 \\ A^T\frac{dy}{dt} + \frac{ds}{dt} &= 0 \\ S(t)\frac{dx}{dt} + X(t)\frac{ds}{dt} &= -X(t)s(t). \end{aligned}$$

Differentiating each equation with respect to t, we find

$$\begin{aligned} A\frac{d^2x}{dt^2} &= 0 \\ A^T\frac{d^2y}{dt^2} + \frac{d^2s}{dt^2} &= 0 \\ S(t)\frac{d^2x}{dt^2} + X(t)\frac{ds}{dt^2} &= X(t)s(t) - 2\frac{dX}{dt}\frac{ds}{dt}. \end{aligned}$$

Note that the first two equations in this system are the same as those for computing the affine-scaling direction. A second-order approximation to the central path is then given by

$$(\hat{x}(t), \hat{s}(t)) = (x(0), s(0)) + t\left(\frac{dx}{dt}(0), \frac{ds}{dt}(0)\right) + \frac{t^2}{2}\left(\frac{d^2x}{dt^2}(0), \frac{d^2s}{dt^2}(0)\right).$$

The vector of complementarity products is then

$$\hat{X}(t)\hat{s}(t) = (1 - t + t^2/2)X(0)s(0) + O(t^3).$$

"$O(t^3)$" above is a somewhat abusive thing to write, since t is typically taken fairly large—close to 1 if possible. So the $O(t^3)$ term is not necessarily negligible.

Mehrotra [49] described a search direction that is a linear combination of the affine-scaling, constant-gap centring, and second-order corrector directions and reported on its computational effectiveness. Mehrotra's technique is very important in most, if not all, practical interior-point codes for linear optimisation.

Mehrotra first computes the affine scaling direction (d_x^{aff}, d_s^{aff}). Then he finds the longest primal and dual step lengths in the affine scaling direction, α_P^{aff} and α_D^{aff}, that do not leave the nonnegative orthant. Then he computes $\mu' = (x + \alpha_P^{aff} d_x^{aff})^T(s + \alpha_D^{aff} d_s^{aff})/n$. He takes $\gamma = (\mu'/\mu)^3$, guessing that a factor-γ reduction in μ is achievable, and looks for a search direction satisfying

$$Sd_x^{meh} + Xd_s^{meh} = -Xs + \gamma\mu e + D_x^{aff} d_s^{aff}.$$

If a unit step in the direction (d_x^{meh}, d_s^{meh}) is possible, the vector of complementarity products becomes

$$(X + D_x^{meh})(s + d_s^{meh}) = \gamma\mu e + D_x^{aff} d_s^{aff} + D_x^{meh} d_s^{meh}.$$

A variant of this procedure, implemented in Matlab's `linprog` and Zhang's LIPSOL, takes $\gamma = \min((\mu'/\mu)^3, 0.208^{3/2})$. The reason for the choice of 0.208 is obscure to me, but apparently it is due to Yin Zhang. (For what it's worth, the corresponding comment in the LIPSOL source code simply says "don't ask why.")

5.4.2 Higher-order Mehrotra-like correctors

The power series derivation given in the previous subsection can be extended to higher derivatives than just the second derivative. This was first done by Monteiro, Adler, and Resende [54]. Mehrotra's original paper [49] reports on experiments with these higher-order approximations to the central path, finding that they do not substantially improve iteration counts.

5.4.3 Gondzio's correctors

Gondzio [22] proposed another technique for improving search directions and reported on its practical effectiveness.

Given an iterate (x, s) and a search direction (d_x, d_s), Gondzio computes the largest $\alpha_P \leq 1$ and $\alpha_D \leq 1$ such that $x' := x + \alpha_P d_x \geq 0$ and $s' := s + \alpha_D d_s \geq 0$. Gondzio then computes $\mu' = s'^T x'/\theta$ and forms the vector $v' = X's'$. Gondzio then computes the vector w' to be the closest point on the hypercube $[\beta_{min}\mu, \beta_{max}\mu]^\theta$ and finds search directions such that $Sd'_x + Xd'_s = v' - w'$. These search directions are added to the current search directions. Gondzio iterates this procedure until an acceptability test is failed or too many Gondzio correctors have been computed.

There are several variants on this technique. In Gondzio's paper [22], Gondzio takes $\beta_{min} = 0.1$, $\beta_{max} = 10$, $\delta_\alpha = 0.1$, and $\gamma = 0.1$.

Gondzio's HOPDM code does something more elaborate. HOPDM modifies α_P and α_D taking $\alpha_P \leftarrow \min(1, 1.08\alpha_P + 0.08)$ and doing similarly for α_D. If the ith complementarity product $x'_i s'_i$ is less than 0.1 of the target μ, the ith entry of the right-hand side vector is $\mu - x'_i s'_i$. If $x'_i s'_i > 10\mu$, the ith entry of the right-hand side vector is -5μ. Otherwise, the ith entry of the right-hand side vector is 0.

Mészáros's BPMPD code does something very similar, but with 1.08 and 0.08 replaced by 1.09 and 0.09, respectively.

5.4.4 Rank-four updates to an old scaling

Suppose a factorisation of $AX_{old}S_{old}^{-1}A^T = LL^T$ is on hand. One can use a rank-four update as described in Section 2.6 to get a primal-dual scaling $T^2 = X_{old}S_{old}^{-1} + UDU^T$ where D is a 4×4 matrix. Then one has

$$AT^2A^T = LL^T + AUDU^TA^T.$$

Thus, one can solve linear systems whose left-hand side is AT^2A^T; if $AT^2A^Td_y = r$, then

$$L^Td_y = L^{-1}r - L^{-1}AU\left(D^{-1} + U^TA^TL^{-T}L^{-1}AU\right)^{-1}U^TA^TL^{-T}L^{-1}r.$$

To find d_y, compute the $m \times 4$ matrix $V := L^{-1}AU$ and the vector $s = L^{-1}r$. Then

$$L^Td_y = s - V\left(D^{-1} + V^TV\right)^{-1}V^Ts.$$

The matrix inverse here is of a 4×4 matrix. I have found that, when D and U represent two consecutive DFP updates in difference-of-squares form, this 4×4 system is typically well-behaved.

A single Richardson iteration sometimes improves this step. Explicitly, if T^2 is $X_{old}S_{old}^{-1}$ plus a rank-four update, in addition to considering the direction defined by

$$d_y^1 = (AT^2A^T)^{-1}r,$$

one can also consider the direction defined by

$$d_y^2 = d_y^1 + (AT^2A^T)^{-1}(r - AXS^{-1}A^Td_y^1)$$

where r is the desired right-hand side. Further Richardson iterations seem to hurt rather than help, but I am not sure why.

Thus far, I have not been able to make this technique, with or without the Richardson iteration, effective in computation. While it is easy to reduce the total number of Cholesky factorisations needed using this technique, it does not seem to reduce running time.

5.5 Numerical issues

The following two theorems are due to Demmel. They show that, in floating-point arithmetic, the computed Cholesky factorisation is the Cholesky factorisation of a nearby matrix, and, when this factorisation is used to solve a linear system in the usual way, one can bound the distance from the solution obtained to the solution desired. It is interesting and useful because the notion of "nearby" used is independent of scaling, the bounds are nearly tight, and Theorem 5.5.2 allows for accounting of error in computing the left-hand side.

Theorem 5.5.1 ([16], Lemma 2.1)**.** *Assume that the result of a binary operation in floating-point arithmetic differs from the true result by a factor at most $1 + \epsilon$. Let H be a positive definite matrix with floating-point entries. Let L be the Cholesky factor of H computed in floating-point arithmetic. Then there exists a matrix E such that $LL^T = H + E$ where, for every i and j,*

$$|E_{ij}| \leq \frac{(n+1)\epsilon}{1-(n+1)\epsilon}\sqrt{H_{ii}H_{jj}}.$$

Theorem 5.5.2 ([16], Theorem 2.1)**.** *Assume that the result of a binary operation in floating-point arithmetic differs from the true result by a factor at most $1 + \epsilon$. Let H be a positive definite matrix with floating-point entries. Let L be the Cholesky factor of H computed in floating-point arithmetic. Let δH be a perturbation satisfying $|\delta H_{ij}| \leq \eta\sqrt{H_{ii}H_{jj}}$ for some $\eta \geq 0$. Let D and A be such that D is diagonal and positive definite, A has unit diagonal, and $H = DAD$.*

Suppose one desires $x_{true} := (H + \delta H)^{-1}r$, and so one computes $x_{fp} := L^{-T}(L^{-1}r)$ by back-substitution and forward-substitution in floating-point arithmetic. Then

$$\frac{||D(x_{fp} - x_{true})||_2}{||Dx_{fp}||} \leq \left(n\eta + \frac{(3n^2 + n + n^3\epsilon)\epsilon}{1-(n-1)\epsilon}\right)\kappa(A),$$

where $\kappa(A)$ is the condition number of A, i.e. $||A||_2\,||A^{-1}||_2$.

5.6 Controlling step lengths in practice

We can prove convergence of interior-point methods based on explicit neighbourhoods. Instead of using an explicit neighbourhood, however, most practical LP codes instead take steps some large fraction of the way to the boundary in the current search direction. This large fraction is classically chosen to be a number like 0.99, 0.995, 0.999, or 0.9999. Mehrotra [49] uses a different choice of step length. By default, my implementation uses 0.9 for this large fraction for "fresh" steps and 0.5 here when a step is derived from a rank-four update to an old scaling.

One classically-known drawback of taking steps very close to the boundary is that the linear system $AXS^{-1}A^T$ becomes very ill-conditioned; it may become impossible to solve using double-precision arithmetic if one is too careless. Other implementations deal with this problem by omitting the ith column of A when forming $AXS^{-1}A^T$ if x_i is very small or treating x_i as a free variable if s_i is very small; my implementation does not do this.

Another classical practically-motivated modification of more theoretically sound interior-point methods discussed thus far is to take steps of different lengths in the primal and dual. My code does this as well.

On some problems, taking a step involving Mehrotra's corrector speeds convergence even when it decreases the barrier parameter μ by less than an affine-scaling step would. I am far from the first to notice this. One can observe this phenomenon in the wild on the NETLIB instance `dfl001` and Mészáros's instance `world`, among others.

If an implementation is using Gondzio correctors, it must select an appropriate number of Gondzio correctors. BPMPD and HOPDM use a heuristic based on the size of the Cholesky factor and the number of flops needed to compute it to compute a limit on the number of Gondzio correctors, together with a heuristic based on the improvement in centrality obtained by the most recent Gondzio corrector, to terminate correction early if insufficient progress is made. Gondzio's original paper on Gondzio correctors [23] experimented with varying the number of Gondzio correctors used between 1 and 10 and gave computational results. My implementation uses at most four Gondzio correctors by default.

The rank-four update scheme of Section 5.4.4 is new. I have been unable to find a rule that results in an overall computational savings. By default, my implementation takes up to 16 steps, stopping if the reduction in potential per unit time is worse than twice the average reduction from a "fresh" step. Limiting the number of steps based on the rank-four update scheme to 0, 1, or 2 seems to work better.

It should be noted that, when taking a step, one is nominally computing $x'_i \leftarrow x_i + \alpha dx_i$ for each i, and similarly for y and s. After doing so, thanks to roundoff, the equality $x'_i = x_i + \alpha dx_i$ will almost never hold exactly. Any interior-point method using approximate arithmetic must correct, rather than amplify, these errors.

5.7 Handling infeasibility and unboundedness via self-dual embedding

5.7.1 The Ye-Todd-Mizuno self-dual embedding

Ye, Todd, and Mizuno [99] proposed the following self-dual embedding of (5.1) and its dual:

$$\begin{array}{rccccccc} \min & & & & h_0\theta & & & \\ \text{subject to} & & -Ax & +b\tau & -b_0\theta & & = & 0 \\ & A^Ty & & -c\tau & +c_0\theta & +s & = & 0 \\ & -b^Ty & +c^Tx & & -g_0\theta & +\kappa & = & 0 \\ & b_0^Ty & -c_0^Tx & +g_0\tau & & & = & h_0 \\ & & x \geq 0, & \tau \geq 0, & & s \geq 0, \quad \kappa \geq 0. & & \end{array} \tag{5.5}$$

The optimal solution to this problem always has objective value 0, and it is attained when $\theta = 0$. The number h_0 will always be positive; it is $s_0^Tx_0 + \kappa_0\tau_0$.

By strict complementarity, there exists a solution such that $Xs = 0$, $x + s > 0$, $\kappa\tau = 0$, and $\kappa + \tau > 0$. A path-following interior-point method will provably converge to a strictly complementary solution in the absence of numerical errors [30], so either $\kappa = 0$ or $\tau = 0$ at the optimal solution found by a path-following IPM. Below, I expound on some properties this strictly complementary solution $(x, y, s, \kappa, \tau, \theta)$ has and how one can interpret the variables.

The variable τ represents a scaling of all of the primal and dual variables. If it is nonzero at the optimal solution, then the original problem (5.1) has an optimal solution at x/τ, and the original dual (5.2) has an optimal solution at $(y/\tau, s/\tau)$.

The third equation is "backward weak duality"; if x is feasible for (5.1) and (y, s) is feasible for (5.2), then $c^T x \geq b^T y$ with equality exactly when both x and (y, s) are optimal for their respective problems. This third equation is its reverse, $c^T x \leq b^T y$, with κ acting as a slack and g_0 being a perturbation. If κ at the strictly complementary optimal solution is nonzero, then we have positive $x \in \ker A$ and $A^T y + s = 0$ such that $b^T y > c^T x$. Either $c^T x < 0$, in which case the dual is infeasible, or $b^T y > 0$, in which case the primal is infeasible, or both. If $\kappa = 0$ at the optimal solution, then $c^T x = b^T y$ so both primal and dual are optimal.

In this formulation, an arbitrary pair of strictly positive x and s can be chosen and τ and κ may initially be set to arbitrary positive values; the parameters b_0, c_0, g_0, and h_0 can be chosen so that all of the above equations are approximately satisfied. This provides an attractive way to deal both with the difficulty of finding an initial strictly feasible point while working within the framework of feasible-start interior-point methods. Further, b_0, c_0, g_0, and h_0 can be recomputed from iteration to iteration in order to deal with (small) numerical error as it arises. One can interpret b_0 as the scaled primal residual and c_0 as the scaled dual residual.

The fourth equation in (5.5) actually has a purpose beyond simply making (5.5) self-dual. If the optimal face in either (5.1) is unbounded (for instance because a free variable was split into a positive-negative pair), the barrier problem (2.3) will be unbounded below for every $\mu > 0$. This can cause numerical problems—in the free variable splitting case, the positive and negative parts will drift away from zero at the same rate, reducing precision. For optimal x, y, s, and τ, the fourth equation can be rewritten as $s^T x_0 + x^T s_0 = K$ for some constant K. Since x_0 and s_0 are strictly positive vectors, this defines a bounded subset of the optimal face. This was known to Xu, Hung, and Ye [98], but it does not appear in [99].

Ye, Todd, and Mizuno also point out that search directions for (5.5) can be computed by solving three systems involving only x, y, and s and then working out the solution to a 2×2 linear system—the linear algebra approaches discussed before can be used within the context of this embedding with essentially no modification.

5.7.2 Xu, Hung, and Ye's modification

Xu, Hung, and Ye [98] proposed a simplification of Ye, Todd, and Mizuno's scheme. To find search directions, they solve the system

$$\begin{array}{rrrrrrrl} \min & & & & h_0\theta & & & \\ \text{subject to} & & -Ad_x & +bd_\tau & & & = & r_1 \\ & A^Td_y & & -cd_\tau & +d_s & & = & r_2 \\ & -b^Td_y & +c^Td_x & & & +d_\kappa & = & r_3 \\ & & Sd_x & & +Xd_s & & = & r_4 \\ & & & \kappa d_\tau & & +\tau d_\kappa & = & r_5 \\ & & d_x \geq 0, & d_\tau \geq 0, & d_s \geq 0, & d_\kappa \geq 0 & & \end{array} \tag{5.6}$$

for various choices of $r_{1\ldots5}$. It appears from their paper that the linear system (5.6) is solved using a direct method rather than by the trick used to solve the system in Ye, Todd, and Mizuno's formulation. My implementation essentially uses the linear system (5.6) to find search directions, but it solves the linear system (5.6) by taking an appropriate linear combination of directions that arise from solving simpler systems of the form

$$\begin{array}{rrrl} & Ad_x & & = r'_1 \\ A^Td_y & & +d_s & = r'_2 \\ & Sd_x & +Xd_s & = r'_3 \end{array}$$

The variables τ and κ destroy the separation between primal and dual. One therefore needs to adapt any desired trick to this setting.

Xu, Hung, and Ye still take separate steps in the primal and dual in some cases. If α_P is the primal step and α_D is the dual step, both are capped at a multiple of the longest admissible step of τ and of κ, and these variables are updated by $\tau' = \tau + \min(\alpha_P d_\tau, \alpha_D d_\tau)$ and either $\kappa' = \kappa + \alpha_D d_\kappa$ if $\tau' = \tau + \alpha_P d_\tau$, or $\kappa' = \kappa + \alpha_P d_\kappa$ otherwise. The choice of τ' means either the primal variables x or the dual variables y and s must be scaled.

Mehrotra's corrector can be adapted to the Xu-Hung-Ye setup by ignoring κ and τ when finding the Mehrotra corrector direction, then taking steps as described above. It may also be possible to modify Mehrotra's corrector to account for κ and τ; I have not tried any such technique.

Gondzio's correctors can also be adapted to the Xu-Hung-Ye setup. Gondzio's correctors make use of a "trial point"—if a step were to be taken using the current direction, the trial point indicates where the step would take the iterate. I compute this trial point, including a trial κ and τ, using Xu, Hung, and Ye's step selection technique. I also threshold the complementarity product $\kappa\tau$ before computing a Gondzio corrector as is done for all other complementarity products.

Other schemes for dealing with infeasibility and accumulated roundoff. Infeasible-start interior-point methods, pioneered by Lustig, Marsten, and Shanno [44], are used by commercially successful codes such as Gurobi and the symmetric cone optimisation problem solver SDPT3 [87].

Another elegant self-dual embedding scheme is due to Nesterov, Todd, and Ye [60]. Mizuno and Todd [51] investigated the (nontrivial) relationship between this Nesterov-Todd-Ye embedding and the Ye-Todd-Mizuno embedding given above.

5.8 Sparse Cholesky factorisation

Sparse Cholesky factorisation can be implemented to take advantage of the fast matrix multiplication provided by a platform-specific BLAS. "Supernodal" Cholesky factorisation was introduced by Ashcraft, Grimes, Lewis, Peyton, and Simon [3]. Rothberg and Gupta [77] give a detailed discussion of their implementation of supernodal Cholesky factorisation, importantly discussing the effects of the (then new) processor cache on performance. Ng and Peyton [66] discuss a parallel sparse Cholesky implementation. Where relevant, I will describe the supernodal data structure used in the successful CHOLMOD package [12].

In a supernodal factorisation, the lower-triangular factor L (of $A = LL^T$) is stored as a sequence of "supernodes." Each supernode represents a contiguous sequence of columns of L. A supernode is equipped with beginning and ending columns, an array of row indices, and a rectangular matrix representing the entries of L in the specified rows and columns. In CHOLMOD, these rectangular matrices are stored in column-major order.

When coalescing adjacent columns into a supernode, some supernodal Cholesky packages, CHOLMOD among them, will represent explicitly some matrix entries that are structurally zero in the interest of reducing the number of supernodes. (This trick does not appear in [3], but it does appear in CHOLMOD. I am not certain of its origin.) This can make factorisation faster, but the added "nonzeros" can make the factor larger and backsolves slower.

A remedy for the increased memory traffic associated with supernodal Cholesky backsolves is to convert the factor to "simplicial form"—list, for each column, the row of every nonzero in that column and the value of that nonzero. In this way, one can guarantee that no zero entries in the factor take any memory. This helps; however, backsolves using this data structure do essentially random access to the right-hand side vector and make poor use of the processor caches.

A hybrid data structure can be designed. Rothberg and Gupta [77] discussed a "blocked supernodal" data structure and a Cholesky factorisation routine that produces it as output. Borrowing heavily from their ideas, I postprocess a supernodal factorisation produced by CHOLMOD, converting it to an array of blocks. In my implementation, there are four types

of blocks:

- "Diagonal triangle": A dense lower-triangular matrix together with starting and ending column indices.
- "Subdiagonal rectangle": A dense rectangular matrix together with starting and ending column indices and an array of row indices.
- "Packed subdiagonal rectangle": A dense rectangular matrix together with starting and ending column indices and starting and ending row indices.
- "Sparse triangle": Starting and ending column indices, and, for each represented column, a list of row indices of nonzero entries and the values of those nonzeros.

Lower-triangular backsolves pass over this array of blocks in forward order, asking each block to do whatever it needs to make a lower-triangular solve happen. Upper-triangular backsolves pass over this array in reverse order.

A heuristic is employed to convert a supernodal factorisation into this new data structure. This heuristic is something of a mess. I will now give an overview. This overview is slightly inaccurate in the interest of simplicity of presentation. Refer to the code for details.

I keep an array of "pending sparse entries," each of which is a row-column-entry triple. I process the supernodes in order. For each supernode, I decompose the diagonal triangular part and I decompose the subdiagonal rectangular part.

Decomposing a diagonal triangle means that, if it is small and dense, I generate a diagonal triangle block. Otherwise, if it is very sparse, I append its nonzeros to the array of pending sparse entries. Otherwise, I break it into a top-left triangle, a bottom-left rectangle, and a bottom-right triangle, recursively decomposing those.

Decomposing a subdiagonal rectangle means that, if it is small and dense, I generate a (possibly packed) subdiagonal rectangle block. Otherwise, if it is very sparse, I append its nonzeros to the array of pending sparse entries. Otherwise, I break it into either top and bottom halves or left and right halves and recursively decompose those.

This heuristic can probably be improved.

Table 5.1 was generated from a number of matrices from the University of Florida sparse matrix collection. The columns have the following meanings:

- "prob": The name of the matrix being factored.
- "rows": The number of rows in the matrix.
- "nnz(A)": The number of nonzeros in the matrix.

- "nnz(L)": The number of nonzeros in the lower-triangular factor L.
- "fac time": The time, in seconds, needed to do a single numerical Cholesky factorisation of the given matrix. This does not include symbolic factorisation time, which is not reported here.
- "rebuild time": The time, in seconds, needed to convert CHOLMOD's supernodal factorisation to the new blocked data structure.
- "simp bksv": The average time, in seconds, over 50 trials, needed to solve $Lx = b$, where L is the lower-triangular factor, for the vector x using CHOLMOD's simplicial factorisation.
- "simp bksv4": The average time, in seconds, over 25 trials, needed to solve $LX = B$, where L is the lower-triangular factor, for the $m \times 4$ matrix X stored in column-major order using CHOLMOD's simplicial factorisation.
- "new bksv": The average time, in seconds, over 50 trials, needed to solve $Lx = b$, where L is the lower-triangular factor, for the vector x using the new blocked data structure.
- "new bksv4": The average time, in seconds, over 25 trials, needed to solve $LX = B$, where L is the lower-triangular factor, for the $m \times 4$ matrix X stored in column-major order using the new blocked data structure.

In the CHOLMOD tests, a workspace was allocated up-front to avoid counting memory allocation and deallocation overhead and the backsolve was done in-place using `cholmod_super_lsolve`.

The table 5.2 gives average Cholesky factorisation, rebuild, and backsolve times for three different factorisations on a variety of linear optimisation problems. I tested both CHOLMOD's supernodal and simplicial data structures here. For the simplicial factorisation tests, a supernodal factorisation was performed which was then converted to simplicial. I ran my LP solver to completion with at least the arguments `--no-stale`; for the simplicial test, `--simplicial-factor` was added, and for the new factorisation test, `--rebuild-factor` was added.

All of the times reported in Table 5.2 correspond to the average time needed to do a certain thing over an entire run of the solver on an input. The run represented in the "fac time" and "rebuild time" columns is the one with the new data structure; in the performance profile printed at exit, these are "dfp::update_root" and "supernodal_factorisation::rebuild", respectively. (Notably, "fac time" includes the time needed to form AA^T.) The "change time" column is the average time needed to convert a supernodal factorisation to simplicial. "bksv" means a lower-triangular backsolve with a single right-hand side; it is the "lower_half_solve"

prob	rows	nnz(A)	nnz(L)	fac time	rebuild time	simp bksv	simp bksv4	new bksv	new bksv4
af_5_k101	503625	9.03e6	9.88e7	3.57	0.308	0.082	0.163	0.070	0.122
ship_003	121728	4.10e6	6.09e7	3.80	0.192	0.043	0.088	0.038	0.068
G3_circuit	1585478	4.62e6	9.94e7	4.22	0.480	0.102	0.234	0.082	0.142
ldoor	952203	2.37e7	1.43e8	4.81	0.440	0.123	0.248	0.106	0.180
offshore	259789	2.25e6	8.34e7	5.12	0.276	0.065	0.129	0.057	0.101
nd6k	18000	3.46e6	4.05e7	5.19	0.136	0.027	0.055	0.025	0.044
consph	83334	3.05e6	6.83e7	5.62	0.208	0.047	0.092	0.043	0.077
inline_1	503712	1.87e7	1.72e8	7.58	0.472	0.134	0.262	0.117	0.206
apache2	715176	2.77e6	1.38e8	9.09	0.476	0.113	0.235	0.095	0.168
nd12k	36000	7.13e6	1.15e8	20.70	0.388	0.076	0.155	0.070	0.128
bone010	986703	1.24e7	3.75e8	23.70	1.16	0.284	0.566	0.249	0.443
nd24k	72000	1.44e7	3.26e8	88.64	1.13	0.220	0.443	0.201	0.362

Table 5.1: CHOLMOD's simplicial factorisation structure versus the blocked data structure on several matrices from the University of Florida sparse matrix collection. Times are in seconds.

line in the performance profile. "bksv3" means a lower-triangular backsolve with three right-hand sides stored consecutively; it is the "lower_half_solve_3" line in the performance profile.

Total times required for the LP solver to run on various inputs using various data structures for the Cholesky factorisation are reported in Table 5.4. Doing backsolves with the supernodal factorisation is always slowest. On these seven linear optimisation problems, the simplicial factorisation is faster on two cases, rail2586 and nug15, by 0.5% and 0.9%, the two run at very close to the same speed on one case, fome12, and the new data structure is faster on the remaining four cases by at least 2%. (On pds-40, this 2% difference can be entirely attributed to the solver doing one fewer iteration.)

Times per iteration for the LP solver to run on a large testbed of LP problems is given in Table 5.3. One can compute the geometric mean of the ratios in the fourth column; an average 1.1% speedup is observed over using the simplicial factorisation.

This is not a huge difference. However, I believe the implementation of "rebuild" described above can be improved both to speed it up and to enhance locality during backsolves. Furthermore, I believe that, with some careful programming that I have not yet done, one can bypass CHOLMOD's factorisation and the "rebuild" step entirely, factoring matrices directly into the new data structure for a savings both in factorisation time and in backsolve time.

prob	new time	simp time	ratio
80bau3b	11408	11677	0.976963

prob	new time	simp time	ratio
aa01	47748	49710	0.960531
aa03	49744	48784	1.019679
aa3	24424	25546	0.956079
aa4	12794	12915	0.990631
aa5	22219	23723	0.936602
aa6	16857	17630	0.956154
acc-tight5	11877	11912	0.997062
air02	21508	21323	1.008676
air03	27205	28075	0.969012
air04	47319	51763	0.914147
air05	29940	29940	1.000000
air06	46520	48851	0.952283
aircraft	124759	126014	0.990041
bab5	29948	30594	0.978885
bas1lp	628613	615752	1.020887
baxter	301528	304810	0.989233
biella1	25531	26644	0.958227
bnatt350	95056	98528	0.964761
car4	17482	17678	0.988913
ch	10560	10541	1.001802
complex	76490	76188	1.003964
core2536-691	70952	74370	0.954041
cov1075	18993	19585	0.969773
cq5	17697	18176	0.973647
cq9	38321	39516	0.969759
cr42	11792	12195	0.966954
cre-b	47479	48943	0.970088
cre-d	38143	39004	0.977925
crew1	20673	21046	0.982277
d2q06c	11236	11452	0.981139
dano3mip	148638	159695	0.930762
dbic1	420196	419762	1.001034
dbir1	489189	505919	0.966931
dbir2	576757	585680	0.984765
degen3	13343	13818	0.965625
dfl001	62629	65852	0.951057
ex3sta1	747308	750274	0.996047
fit1p	21259	21350	0.995738
fit2d	54544	51973	1.049468
fome11	130501	138597	0.941586
fome12	291649	291311	1.001160

prob	new time	simp time	ratio
fome13	633380	643766	0.983867
fome20	175049	178567	0.980299
fome21	394436	397333	0.992709
ge	20415	21260	0.960254
iis-100-0-cov	1007954	1000377	1.007574
iis-bupa-cov	1882628	1900653	0.990516
iis-pima-cov	4139796	4169138	0.992962
israel	13357	1890	7.067196
karted	6069745	6088916	0.996851
ken-11	29133	32149	0.906187
ken-13	73360	98999	0.741018
ken-18	351385	385766	0.910876
kl02	54943	52983	1.036993
lp22	67135	69624	0.964251
lpl1	117401	124923	0.939787
lpl3	30678	34512	0.888908
map18	3312067	3262373	1.015232
map20	3293550	3260612	1.010102
maros-r7	38914	41023	0.948590
mine-166-5	234982	233697	1.005499
mine-90-10	46881	49001	0.956736
mod2	123489	129516	0.953465
model10	35071	36325	0.965478
model11	28976	26446	1.095667
model4	12090	12074	1.001325
model5	22550	22360	1.008497
model8	19009	19536	0.973024
model9	12338	12155	1.015056
msc98-ip	712657	726034	0.981575
mzzv11	134805	142763	0.944257
n3700	20489	19294	1.061936
n3701	18554	19099	0.971464
n3702	18216	19590	0.929862
n3703	19325	20937	0.923007
n3704	18733	19452	0.963037
n3705	18224	18932	0.962603
n3706	19514	22657	0.861279
n3707	18715	19095	0.980100
n3708	18439	19225	0.959116
n3709	18556	19265	0.963198
n370a	18606	18993	0.979624

prob	new time	simp time	ratio
n370b	20144	22025	0.914597
n370c	18458	19155	0.963613
n370d	19132	19338	0.989347
n370e	19148	19573	0.978286
n3div36	40226	40410	0.995447
n3seq24	1534052	1509982	1.015941
n4-3	9705	10441	0.929509
nemswrld	60050	65491	0.916920
neos-1109824	564071	557486	1.011812
neos-1337307	22328	22591	0.988358
neos-1601936	123323	126191	0.977273
neos-476283	12339264	12320614	1.001514
neos-686190	404479	399035	1.013643
neos-916792	110378	109751	1.005713
neos-934278	33734	35813	0.941948
neos18	58514	60288	0.970575
net12	512301	518795	0.987483
netdiversion	738186	728275	1.013609
nl	28884	30472	0.947887
ns1208400	84674	90816	0.932369
ns1687037	13861319	13926621	0.995311
ns1688347	43994	45258	0.972071
ns1830653	43370	47719	0.908862
nsct1	962242	959097	1.003279
nsct2	1021798	1053340	0.970055
nsir1	59225	60531	0.978424
nsir2	66430	68278	0.972934
nug06	12546	12813	0.979162
nug12	151689	155618	0.974752
nug15	935704	930295	1.005814
nw14	400163	393688	1.016447
osa-07	15820	15814	1.000379
osa-14	41966	40949	1.024836
osa-30	84614	85232	0.992749
osa-60	229473	232131	0.988550
p010	21053	22291	0.944462
p05	10668	11016	0.968410
p6000	10706	11110	0.963636
pcb1000	10747	10868	0.988866
pcb3000	37861	37490	1.009896
pds-06	29375	31143	0.943230

prob	new time	simp time	ratio
pds-10	57910	61112	0.947604
pds-100	3024732	3076222	0.983262
pds-20	173215	180517	0.959550
pds-30	419310	433062	0.968245
pds-40	718851	719005	0.999786
pds-50	1130838	1158351	0.976248
pds-60	1580642	1615689	0.978308
pds-70	2044622	2076062	0.984856
pds-80	2537682	2556890	0.992488
pds-90	2813292	2895429	0.971632
pf2177	851991	831026	1.025228
r05	11328	11547	0.981034
rail2586	2436001	2538205	0.959734
rail4284	4378145	4392454	0.996742
rail507	104570	104108	1.004438
rail516	79460	78885	1.007289
rail582	103120	102825	1.002869
ran10x26	10280	10406	0.987892
ran12x21	9912	10009	0.990309
ran14x18	10163	9994	1.016910
ran16x16	10286	10107	1.017710
ran17x17	11990	12076	0.992878
rat5	24276	25852	0.939038
rat7a	91073	94172	0.967092
reblock67	9934	10294	0.965028
rlfddd	85124	88318	0.963835
rlfdual	74061	77471	0.955984
rmatr100-p10	815880	803813	1.015012
rmatr100-p5	1335361	1351451	0.988094
rmine6	68744	72199	0.952146
rocII-4-11	1320023	1327365	0.994469
roll3000	13596	13637	0.996993
satellites1-25	161276	169953	0.948945
seymourl	331912	321672	1.031834
sgpf5y6	1683642	1685697	0.998781
slptsk	627237	620637	1.010634
sp98ic	65663	64021	1.025648
sp98ir	23223	22211	1.045563
t0331-4l	127582	127116	1.003666
tanglegram2	2538469	2502351	1.014434
testbig	232782	246278	0.945200

prob	new time	simp time	ratio
triptim1	555807	559713	0.993021
ulevimin	47670	54749	0.870701
unitcal_7	146876	142973	1.027299
us04	49627	48475	1.023765
vpphard	593656	603071	0.984388
watson_1	346178	339463	1.019781
world	124969	127210	0.982383
zib54-UUE	11222	11872	0.945249

Table 5.3: Per-iteration running time of the LP solver on various inputs doing Cholesky backsolves with the simplicial and new data structures.

5.9 Other computational experiments

5.9.1 Comparison against CPLEX 12.4

Table 5.5 gives the results of a comparison against the commercial CPLEX 12.4 barrier algorithm. In this experiment, my solver was run with the arguments `--rebuild-factor --no-stale --meh-neigh=0.995`. The CPLEX 12.4 command-line tool was run, and the following commands were provided to it:

- `set threads 1` to run in single-threaded mode,
- `set pre pre n` to disable presolve,
- `read <problem>` to read the problem,
- `change problem lp` to discard any integrality constraints, and
- `baropt` to solve it.

Notably, this does not disable CPLEX's crossover.

Both solvers were timed using the command `time -f %U`, which reports user-mode seconds used.

The "dense column" column in Table 5.5 indicates whether CPLEX detected a dense column in the problem. (If it found a dense column, it solved the linear system at each iteration in

prob	fac time	rebuild time	change time	sup bksv	sup bksv3	simp bksv	simp bksv3	new bksv	new bksv3
dfl001	45230	3410	2043	815	2625	781	1424	628	1306
world	56920	7361	3006	2571	6264	1746	2929	1181	2554
ken-18	129713	15093	7330	7497	17327	4119	8080	1932	4595
fome12	210105	13235	9997	3957	11210	3762	6057	3094	5529
pds-40	458228	24821	15585	8113	21127	6432	10903	4452	8574
rail2586	604420	3871	2083	934	2956	912	1493	907	1501
nug15	851235	22606	15869	5108	15144	5189	8771	4770	8149

Table 5.2: CHOLMOD's stock factorisation structure versus the blocked data structure on AA^T for several linear optimisation problems. Times are in microseconds.

prob	sup iters	sup time	simp iters	simp time	new iters	new time
dfl001	34	2244742	34	2220240	34	2128297
world	55	8053134	56	7412580	56	7211394
ken-18	24	10521166	24	9320278	24	9007195
fome12	34	10175487	34	9870643	34	9861735
pds-40	51	39046029	50	37136633	49	36376561
rail2586	33	84619646	33	83838558	33	84193674
nug15	19	17805465	19	17626146	19	17783321

Table 5.4: Total running time of the LP solver on various inputs with various Cholesky data structures. Times are in microseconds.

a different way from my implementation.) The "dual formed" column in Table 5.5 indicates whether CPLEX solved the dual problem instead. (Again, this indicates that a fundamentally different computation was performed.) The remaining columns indicate the problem name, the amount of time needed by each solver, and the number of iterations performed by each solver.

The problems selected for this table are those for which I have a CPLEX-presolved problem, for which neither solver failed, and for which at least one solver took at least five seconds to complete.

The results of this experiment show that my solver is usually slower than CPLEX, but that it is roughly competitive with CPLEX on many problems.

prob	dense column?	dual formed?	my time	my iterations	CPLEX time	CPLEX iterations
bas1lp	-	-	9.82	9	3.95	10
baxter	D	D	11.36	37	1.99	63
dano3mip	D	-	5.40	33	3.73	26
dbic1	-	-	25.61	45	20.02	48
dbir1	-	-	109.99	215	14.56	45
dbir2	-	-	22.82	34	9.68	26
ex3sta1	-	D	7.83	10	0.70	10
fome12	-	-	10.49	34	9.24	28
fome13	-	-	22.59	34	22.31	31
fome20	-	-	9.34	44	7.69	42
fome21	-	-	24.19	55	18.41	50
iis-100-0-cov	-	D	8.29	8	0.04	17
iis-bupa-cov	-	D	25.87	14	0.07	19
iis-pima-cov	-	D	76.32	19	0.21	21
karted	-	-	122.66	19	288.94	20
ken-18	-	-	13.20	22	2.90	30
map18	-	D	89.10	27	8.94	36
map20	-	D	91.73	28	8.86	38
mod2	-	-	6.55	41	3.39	58
msc98-ip	-	D	21.77	30	3.76	28
mzzv11	-	-	7.16	46	4.55	32
n3seq24	-	-	38.43	18	24.21	30
neos-1109824	-	D	11.87	21	1.25	20
neos-476283	-	-	346.84	26	124.65	19
neos-686190	-	-	7.68	19	7.09	14
net12	-	-	14.74	26	14.24	21
netdiversion	-	-	19.36	23	20.04	20
nsct1	-	-	174.58	190	24.96	33
nsct2	-	-	40.78	39	20.60	27

prob	dense column?	dual formed?	my time	my iterations	CPLEX time	CPLEX iterations
nug15	-	-	16.73	18	20.87	17
nw14	-	-	79.27	45	1.52	33
osa-60	-	-	12.06	23	2.02	25
pds-100	-	-	306.62	99	196.48	61
pds-20	-	-	9.22	44	7.73	42
pds-30	-	-	23.99	53	17.30	45
pds-40	-	-	41.73	54	35.96	40
pds-50	-	-	74.10	62	57.38	45
pds-60	-	-	101.56	61	77.00	43
pds-70	-	-	145.24	68	96.29	43
pds-80	-	-	164.78	62	125.30	45
pds-90	-	-	188.42	64	137.80	45
pf2177	-	D	5.57	6	0.53	7
rail2586	-	-	95.36	32	28.31	44
rail4284	-	-	197.58	37	63.46	27
rmatr100-p10	-	D	22.55	28	0.52	36
rmatr100-p5	-	D	34.33	26	0.77	37
rocII-4-11	D	-	18.77	15	0.20	21
sgpf5y6	D	-	49.02	31	0.96	37
slptsk	-	D	9.49	16	0.72	19
t0331-4l	-	-	5.79	24	0.76	21
tanglegram2	-	D	26.04	11	0.07	9
triptim1	-	-	15.01	23	16.79	23
ulevimin	-	-	6.82	120	2.58	140
unitcal_7	-	D	8.37	44	2.51	38
vpphard	D	D	6.20	9	7.24	13
watson_1	-	-	28.26	67	9.62	71
world	-	-	9.02	59	4.32	70

Table 5.5: A comparison of my LP solver against CPLEX 12.4.

5.9.2 Experiments with rank-four updates to old scalings

Table 5.6 gives the results of a comparison among three configurations of my linear optimisation problem solver. These three configurations are:

- `--rebuild-factor--no-stale--meh-neigh=0.995`, as used in the comparison against CPLEX. This is the "plain" algorithm in Table 5.6.

- `--rebuild-factor --meh-neigh=0.95 --stale-neigh=0.5 --max-stale=8 --no-stale-corrector`. This is the "eight" algorithm in Table 5.6.
- `--rebuild-factor --meh-neigh=0.995 --stale-neigh=0.75 --max-stale=1 --no-stale-corrector`. This is the "one" algorithm in Table 5.6.

These conditions are as favourable to techniques that trade off Cholesky factorisations for extra backsolves as I know how to produce; apart from being the fastest factorisation structure overall, the `--rebuild-factor` option increases the cost of each Cholesky factorisation the most and decreases the cost of each backsolve the most.

All runs were timed using the command `time -f %U`, which reports user-mode seconds used.

Once again, the problems selected for this table are those for which I have a CPLEX-presolved problem, for which neither solver failed, and for which at least one solver took at least five seconds to complete.

These results show that rank-four updates can reduce the number of Cholesky factorisations required; the "eight" algorithm shows an average 9% reduction in Cholesky factorisations. However, this 9% reduction in Cholesky factorisations corresponds to an average 65% increase in running time. A more conservative approach, represented by the "one" algorithm, shows a 6.7% average decrease in the number of Cholesky factorisations but a 4.5% average increase in running time.

I have experimented with other configurations involving rank-four updates but I have found none that improve the running time on average. It may be possible to eke out a small gain by selectively disabling the technique on problems where Cholesky factorisations are relatively cheap.

problem	plain chols	plain time	eight chols	eight time	one chols	one time
bas1lp	8	9.820	9	13.380	8	10.100
baxter	37	11.360	29	15.630	35	11.990
cre-b	35	2.280	31	5.440	34	2.620
dano3mip	33	5.400	28	7.740	30	5.310
dbic1	45	25.610	32	51.080	40	28.430
dbir1	215	109.990	149	167.440	184	107.860
dbir2	34	22.820	23	30.970	36	26.820
ex3sta1	10	7.830	8	8.880	8	6.620
fome11	34	4.950	28	7.620	31	5.120
fome12	34	10.490	28	16.630	32	11.080
fome13	34	22.590	28	35.390	32	24.160
fome20	44	9.340	36	20.120	38	10.020
fome21	55	24.190	54	62.030	45	24.300

problem	plain chols	plain time	eight chols	eight time	one chols	one time
iis-100-0-cov	8	8.290	8	9.620	8	8.460
iis-bupa-cov	14	25.870	12	25.730	14	26.240
iis-pima-cov	19	76.320	14	64.360	16	65.760
karted	19	122.660	16	139.270	18	121.610
ken-13	17	2.380	17	5.250	18	2.800
ken-18	22	13.200	22	31.950	22	15.850
kl02	31	3.020	18	5.100	21	2.710
lpl1	17	3.840	17	9.520	15	4.230
map18	27	89.100	30	123.280	26	87.760
map20	28	91.730	32	130.980	26	87.760
mine-166-5	17	4.220	14	5.700	17	4.480
mod2	41	6.550	49	18.690	41	7.650
msc98-ip	30	21.770	26	27.940	28	21.350
mzzv11	46	7.160	36	10.610	40	7.110
n3seq24	18	38.430	19	69.670	17	39.800
nemswrld	26	2.120	29	5.540	26	2.580
neos-1109824	21	11.870	17	13.470	19	11.380
neos-476283	26	346.840	23	358.100	23	315.900
neos-686190	19	7.680	16	8.520	17	7.210
net12	26	14.740	23	21.300	23	14.360
netdiversion	23	19.360	20	40.980	24	23.250
ns1687037	23	346.460	11	198.920	22	337.690
nsct1	190	174.580	173	239.420	228	219.650
nsct2	39	40.780	22	35.270	29	33.090
nug15	18	16.730	15	17.930	18	17.260
nw14	45	79.270	63	144.910	68	95.610
osa-30	23	3.400	18	7.700	26	4.590
osa-60	23	12.060	22	28.400	22	13.990
pds-10	31	2.360	31	6.010	31	2.860
pds-100	99	306.620	84	524.010	95	333.300
pds-20	44	9.220	36	20.040	38	9.980
pds-30	53	23.990	48	50.490	49	25.960
pds-40	54	41.730	50	85.700	44	41.740
pds-50	62	74.100	56	141.120	49	71.020
pds-60	61	101.560	57	197.140	59	113.100
pds-70	68	145.240	64	275.770	64	155.940
pds-80	62	164.780	64	327.060	62	181.120
pds-90	64	188.420	65	378.760	63	209.320
pf2177	6	5.570	5	5.790	5	4.860
rail2586	32	95.360	40	352.800	33	122.590
rail4284	37	197.580	47	665.440	36	235.600

problem	plain chols	plain time	eight chols	eight time	one chols	one time
rail507	20	2.730	21	8.990	19	3.300
rail582	22	2.920	21	8.680	19	3.310
rmatr100-p10	28	22.550	22	23.300	24	20.090
rmatr100-p5	26	34.330	23	38.320	25	33.800
rocII-4-11	15	18.770	20	29.810	14	17.860
seymourl	13	4.770	11	5.440	12	4.610
sgpf5y6	31	49.020	22	43.910	27	44.070
slptsk	16	9.490	14	9.960	16	9.700
t0331-4l	24	5.790	24	11.510	24	6.440
tanglegram2	11	26.040	12	32.790	9	21.410
triptim1	23	15.010	20	22.310	21	15.210
ulevimin	120	6.820	95	17.740	118	8.650
unitcal_7	44	8.370	37	19.320	35	8.340
vpphard	9	6.200	10	10.880	8	5.990
watson_1	67	28.260	62	76.250	64	33.390
world	59	9.020	48	18.840	49	9.180

Table 5.6: A comparison of the algorithm with no rank-four updates, with up to eight rank-four updates, and with at most one rank-four update.

Chapter 6

Concluding remarks

This thesis presented a number of new results related in various ways to interior-point methods for convex optimisation based on primal-dual scalings:

- Theorem 2.5.1 states that a primal-dual scaling (called the dual integral scaling) can be constructed by integrating the Hessian of the barrier over a line segment.
- Theorem 2.6.1 states that two consecutive "orthogonal" quasi-Newton updates can change an arbitrary positive definite matrix into a primal-dual scaling.
- Theorem 2.7.3 states that, within a neighbourhood of the central path, an approximation to the dual integral scaling can be corrected using quasi-Newton updates to obtain a primal-dual scaling that does not badly disturb either the primal Hessian or the dual Hessian.
- Corollary 2.9.4 states that a variant of the Mizuno-Todd-Ye method takes $O(\sqrt{\theta}\log(1/\epsilon))$ iterations on general convex optimisation problems. This variant leaves open the choice of primal-dual scaling; if that primal-dual scaling is chosen in a primal-dual symmetric way, the resulting algorithm generates the same sequence of iterates after interchange of primal and dual problems.
- Theorem 3.5.1 gives explicit bounds on the error in approximating the primal integral scaling for a hyperbolic barrier by Simpson's rule and the 5-point Clenshaw-Curtis quadrature rule.
- Chapter 4 discusses an implementation of a primal-dual interior-point method for optimisation over hyperbolicity cones based on the Simpson's rule approximation to the primal integral scaling.

- Chapter 5 experimentally compares a classical interior-point method for linear optimisation with the quasi-Newton update technique of Theorem 2.6.1 as applied to linear optimisation.
- Section 5.8 discusses a data structure for storing a sparse Cholesky factor that results, experimentally in single-threaded computation, in faster backsolves.

Several basic questions about hyperbolicity cone optimisation remain.

What problems of practical interest can be formulated naturally as hyperbolicity cone optimisation problems but not as symmetric cone optimisation problems? I am not currently aware of any. Given the wealth of applications hyperbolic polynomials have found within mathematics, I continue to find this surprising.

There is, as yet, no fast implementation of a primal-dual interior-point method for hyperbolicity cone optimisation based on Güler's results on hyperbolic barriers.

The popular search directions for semidefinite optimisation do linear algebra on $n \times n$ and $m \times m$ matrices when the problem has m linear equality constraints and involves an $n \times n$ matrix of variables. However, directly finding search directions using any of the primal-dual scalings in this thesis would seem to involve linear algebra with matrices with $n(n+1)/2$ rows and columns. It would be interesting to see—at least, I would be interested in seeing—how quickly search directions for semidefinite optimisation can be computed in practice using the scalings in this thesis.

The computation logs in Chapter 4 show that steps of very close to unit length are possible when close to optimality. Does this happen in all cases? If not, does this happen in all cases that are somehow nondegenerate? Nesterov and Tunçel [61] gave an algorithm where it does, but that algorithm differs from the one in this thesis.

It may be possible to speed up present sparse Cholesky packages, at least in the case where many linear systems are solved in series with the same left-hand side. This usage pattern is not unique to interior-point methods; preconditioned iterative methods for linear algebra do the same.

References

[1] Erling D. Andersen, Jacek Gondzio, Csaba Meszaros, and Xiaojie Xu. Implementation of interior point methods for large scale linear programming. Working papers, Ecole des Hautes Etudes Commerciales, Universite de Geneve-, 1996.

[2] Knud D. Andersen. A modified Schur-complement method for handling dense columns in interior-point methods for linear programming. *ACM Trans. Math. Softw.*, 22(3):348–356, September 1996.

[3] C. Cleveland Ashcraft, Roger G. Grimes, John G. Lewis, Barry W. Peyton, Horst D. Simon, and Petter E. Bjørstad. Progress in sparse matrix methods for large linear systems on vector supercomputers. *International Journal of High Performance Computing Applications*, 1(4):10–30, 1987.

[4] Michael F. Atiyah, Raoul Bott, and Lars Gårding. Lacunas for hyperbolic differential operators with constant coefficients I. *Acta mathematica*, 124(1):109–189, 1970.

[5] Jonathan M. Borwein and Adrian S. Lewis. *Convex analysis and nonlinear optimization: theory and examples*. Springer Science & Business Media, 2010.

[6] Petter Brändén. Polynomials with the half-plane property and matroid theory. *Advances in Mathematics*, 216(1):302–320, 2007.

[7] Petter Brändén. Obstructions to determinantal representability. *ArXiv e-prints*, April 2010.

[8] Petter Brändén. Lecture notes for interlacing families. *Royal Institute of Technology, Stockholm, Sweden*, 2013.

[9] James R. Bunch and Beresford N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 8(4):639–655, 1971.

[10] Sam Burton, Cynthia Vinzant, and Yewon Youm. A real stable extension of the Vamos matroid polynomial. *arXiv preprint arXiv:1411.2038*, 2014.

[11] Coralia Cartis. Some disadvantages of a Mehrotra-type primal-dual corrector interior point algorithm for linear programming. *Appl. Numer. Math.*, 59(5):1110–1119, May 2009.

[12] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3):22:1–22:14, October 2008.

[13] Young-Bin Choe, James G. Oxley, Alan D. Sokal, and David G. Wagner. Homogeneous multivariate polynomials with the half-plane property. *Advances in Applied Mathematics*, 32(1.2):88 – 187, 2004. Special Issue on the Tutte Polynomial.

[14] Charles W. Clenshaw and Alan R. Curtis. A method for numerical integration on an automatic computer. *Numerische Mathematik*, 2(1):197–205, 1960.

[15] Timothy A. Davis. *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

[16] James W. Demmel. On floating point errors in Cholesky. Technical Report 14, LAPACK Working Note, October 1989.

[17] John E. Dennis, Jr. and Robert B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice Hall Series in Computational Mathematics. Prentice Hall, Inc., Englewood Cliffs, NJ, 1983.

[18] Jacques Faraut and Adam Korányi. *Analysis on Symmetric Cones*. Oxford mathematical monographs. Clarendon Press, 1994.

[19] Lars Gårding. An inequality for hyperbolic polynomials. *Journal of Mathematics and Mechanics*, 8:957–965, 1959.

[20] Alan George and Joseph W. Liu. *Computer Solution of Large Sparse Positive Definite*. Prentice Hall Professional Technical Reference, 1981.

[21] David Goldfarb and Katya Scheinberg. A product-form Cholesky factorization method for handling dense columns in interior point methods for linear programming. *Mathematical Programming*, 99(1):1–34, 2004.

[22] Jacek Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Comput. Optim. Appl.*, 6(2):137–156, September 1996.

[23] Jacek Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, 6(2):137–156, 1996.

[24] Jacek Gondzio. Matrix-free interior point method. *Computational Optimization and Applications*, 51(2):457–480, 2012.

[25] Jacek Gondzio and Marek Makowski. HOPDM - modular solver for LP problems: User's guide to version 2.12. In *Working Paper WP-95-50, International Institute for Applied Systems Analysis, Laxenburg*, 1995.

[26] Maria Gonzalez-Lima, Henry Wolkowicz, and Hua Wei. *A stable iterative method for linear programming.* Faculty of Mathematics, University of Waterloo, 2004.

[27] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation.* Siam, 2008.

[28] Osman Güler. Barrier functions in interior point methods. *Mathematics of Operations Research*, 21(4):860–885, 1996.

[29] Osman Güler. Hyperbolic polynomials and interior point methods for convex programming. *Mathematics of Operations Research*, 22:350–377, 1996.

[30] Osman Güler and Yinyu Ye. Convergence behavior of interior-point algorithms. *Mathematical Programming*, 60(1-3):215–228, 1993.

[31] Leonid Gurvits. Combinatorial and algorithmic aspects of hyperbolic polynomials. *arXiv preprint math/0404474*, 2004.

[32] Leonid Gurvits. Van der Waerden/Schrijver-Valiant like conjectures and stable (aka hyperbolic) homogeneous polynomials: one theorem for all. *Electron. J. Combin*, 15(1), 2008.

[33] Raphael A. Hauser and Osman Güler. Self-scaled barrier functions on symmetric cones and their classification. *Foundations of Computational Mathematics*, 2(2):121–143, 2002.

[34] Raphael A. Hauser and Yongdo Lim. Self-scaled barriers for irreducible symmetric cones. *SIAM Journal on Optimization*, 12(3):715–723, 2002.

[35] J. William Helton and Jiawang Nie. Semidefinite representation of convex sets. Technical report, 2007.

[36] J. William Helton and Victor Vinnikov. Linear matrix inequality representation of sets. *Communications on pure and applied mathematics*, 60(5):654–674, 2007.

[37] Lars Hörmander. *The Analysis of Linear Partial Differential Operators II: Differential Operators with Constant Coefficients.* Classics in Mathematics. Springer, 2004.

[38] Pascual Jordan, John von Neumann, and Eugene Wigner. On an algebraic generalization of the quantum mechanical formalism. *Annals of Mathematics*, 35(1):pp. 29–64, 1934.

[39] Erich Kaltofen. On computing determinants of matrices without divisions. In *Papers from the International Symposium on Symbolic and Algebraic Computation*, ISSAC '92, pages 342–349, New York, NY, USA, 1992. ACM.

[40] Igor Klep and Markus Schweighofer. An exact duality theory for semidefinite programming based on sums of squares. *Mathematics of Operations Research*, 38(3):569–590, 2013.

[41] Max Koecher. Positivitätsbereiche im R^n. *American Journal of Mathematics*, pages 575–596, 1957.

[42] Peter D. Lax. Differential equations, difference equations and matrix theory. *Communications on pure and applied mathematics*, 11(2):175–194, 1958.

[43] Adrian Lewis, Pablo Parrilo, and Motakuri Ramana. The Lax conjecture is true. *Proceedings of the American Mathematical Society*, 133(9):2495–2499, 2005.

[44] Irvin J. Lustig, Roy E. Marsten, and David F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing*, 6(1):1–14, 1994.

[45] Adam Marcus, Daniel Spielman, and Nikhil Srivastava. Interlacing families I: Bipartite Ramanujan graphs of all degrees. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 529–537. IEEE, 2013.

[46] Adam Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison-Singer problem. *arXiv preprint arXiv:1306.3969*, 2013.

[47] István Maros and Csaba Mészáros. The role of the augmented system in interior point methods. *European Journal of Operational Research*, 107(3):720–736, 1998.

[48] Nimrod Megiddo. On finding primal-and dual-optimal bases. *ORSA Journal on Computing*, 3(1):63–65, 1991.

[49] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[50] Csaba Mészáros. The BPMPD interior point solver for convex quadratically constrained quadratic programming problems. In *Proceedings of the 7th International Conference on Large-Scale Scientific Computing*, LSSC'09, pages 813–820, Berlin, Heidelberg, 2010. Springer-Verlag.

[51] Shinji Mizuno and Michael J. Todd. On two homogeneous self-dual approaches to linear programming and its extensions. *Mathematical Programming*, 89(3):517–534, 2001.

[52] Shinji Mizuno, Michael J. Todd, and Yinyu Ye. On adaptive-step primal-dual interior-point algorithms for linear programming. *Math. Oper. Res.*, 18(4):964–981, 1993.

[53] Lajos Molnár. Maps preserving the geometric mean of positive operators. *Proc. Amer. Math. Soc.*, 137(5):1763–1770, 2009.

[54] Renato D. C. Monteiro, Ilan Adler, and Mauricio G. C. Resende. A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension. *Mathematics of Operations Research*, 15:191–214, 1990.

[55] Tor G. J. Myklebust. Geometry of convex sets arising from hyperbolic polynomials. Master's thesis, University of Waterloo, 2007.

[56] Tor G. J. Myklebust and Levent Tunçel. Interior-point algorithms for convex optimization based on primal-dual metrics. *ArXiv e-prints*, November 2014.

[57] Yurii Nesterov. Towards nonsymmetric conic optimization. *Available at SSRN 921788*, 2006.

[58] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13 of *SIAM Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994.

[59] Yurii Nesterov and Michael J. Todd. Self-scaled barriers and interior-point methods for convex programming. *Math. Oper. Res.*, 22(1):1–42, 1997.

[60] Yurii Nesterov, Michael J. Todd, and Yinyu Ye. Infeasible-start primal-dual methods and infeasibility detectors for nonlinear programming problems. *Math. Program.*, 84:227–267, 1999.

[61] Yurii Nesterov and Levent Tunçel. Local superlinear convergence of polynomial-time interior-point methods for hyperbolic cone optimization problems. *ArXiv e-prints*, December 2014.

[62] Yurii E. Nesterov and Michael J. Todd. Primal-dual interior-point methods for self-scaled cones. *SIAM J. on Optimization*, 8(2):324–364, February 1998.

[63] Yurii E. Nesterov and Michael J. Todd. On the Riemannian geometry defined by self-concordant barriers and interior-point methods. *Foundations of Computational Mathematics*, 2(4):333–361, 2002.

[64] Tim Netzer and Raman Sanyal. Smooth hyperbolicity cones are spectrahedral shadows. *ArXiv e-prints*, August 2012.

[65] Tim Netzer and Andreas Thom. Hyperbolic polynomials and generalized clifford algebras. *ArXiv e-prints*, July 2012.

[66] Esmond Ng and Barry W. Peyton. A supernodal Cholesky factorization algorithm for shared-memory multiprocessors. *SIAM Journal on Scientific Computing*, 14(4):761–769, 1993.

[67] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[68] Wim Nuij. A note on hyperbolic polynomials. *Mathematica Scandinavica*, 23:69–72, 1968.

[69] Motakuri V. Ramana. An exact duality theory for semidefinite programming and its complexity implications. *Mathematical Programming*, 77(1):129–162, 1997.

[70] Motakuri V Ramana, Levent Tunçel, and Henry Wolkowicz. Strong duality for semidefinite programming. *SIAM Journal on Optimization*, 7(3):641–662, 1997.

[71] James Renegar. A polynomial-time algorithm, based on Newton's method, for linear programming. *Mathematical Programming*, 40(1-3):59–93, 1988.

[72] James Renegar. Hyperbolic programs, and their derivative relaxations. *Foundations of Computational Mathematics*, 6(1):59–79, 2006.

[73] James Renegar. Central swaths. *Foundations of Computational Mathematics*, 13(3):405–454, 2013.

[74] James Renegar and Mutiara Sondjaja. A polynomial-time affine-scaling method for semidefinite and hyperbolic programming. *arXiv preprint arXiv:1410.6734*, 2014.

[75] R. Tyrrell Rockafellar. *Convex analysis*. Princeton Mathematical Series, No. 28. Princeton University Press, Princeton, N.J., 1970.

[76] Günter Rote. Division-free algorithms for the determinant and the Pfaffian: Algebraic and combinatorial approaches. In *Computational Discrete Mathematics*, pages 119–135. Springer, 2001.

[77] Edward Rothberg and Anoop Gupta. Efficient sparse matrix factorization on high performance workstations—exploiting the memory hierarchy. *ACM Trans. Math. Softw.*, 17(3):313–334, September 1991.

[78] Olaf Schenk, Klaus Gärtner, Wolfgang Fichtner, and Andreas Stricker. PARDISO: a high-performance serial and parallel sparse linear solver in semiconductor device simulation. *Future Generation Computer Systems*, 18(1):69–78, 2001.

[79] Stefan H. Schmieta. Complete classification of self-scaled barrier functions. Technical report, Technical Report, Department of IEOR, Columbia University, New York, NY 10027, 2000.

[80] Rolf Schneider. *Convex bodies: the Brunn–Minkowski theory*. Number 151. Cambridge University Press, 2013.

[81] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[82] Anders Skajaa, John Bagterp Jørgensen, and Per Christian Hansen. On implementing a homogeneous interior-point algorithm for nonsymmetric conic optimization. Technical report, Technical University of Denmark, DTU Informatics, Building 321, 2011.

[83] Jos F. Sturm. Implementation of interior point methods for mixed semidefinite and second order cone optimization problems. *Optimization Methods and Software*, 17(6):1105–1154, 2002.

[84] Levent Tuncel. Potential reduction and primal-dual methods. In Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe, editors, *Handbook of Semidefinite Programming*, volume 27 of *International Series in Operations Research & Management Science*, pages 235–265. Springer US, 2000.

[85] Levent Tunçel. Generalization of primal-dual interior-point methods to convex optimization problems in conic form. *Found. Comput. Math.*, 1(3):229–254, 2001.

[86] Levent Tunçel and Henry Wolkowicz. Strong duality and minimal representations for cone optimization. *Computational optimization and applications*, 53(2):619–648, 2012.

[87] Reha H. Tütüncü, Kim Chuan Toh, and Michael J. Todd. SDPT3: a Matlab software package for semidefinite-quadratic-linear programming, version 3.0. 2001.

[88] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189 – 201, 1979.

[89] Robert J. Vanderbei. Splitting dense columns in sparse linear systems. *Linear Algebra and its Applications*, 152(0):107 – 117, 1991.

[90] Stephen A Vavasis and Yinyu Ye. A primal-dual interior point method whose running time depends only on the constraint matrix. *Mathematical Programming*, 74(1):79–120, 1996.

[91] Èrnest Borisovich Vinberg. The theory of convex homogeneous cones. *Trans. Moscow Math. Soc*, 12:340–403, 1963.

[92] David G. Wagner and Yehua Wei. A criterion for the half-plane property. *DISCRETE MATH*, pages 1385–1390, 2007.

[93] William C. Waterhouse. Linear transformations preserving symmetric rank one matrices. *J. Algebra*, 125(2):502–518, 1989.

[94] Hua Wei. *Convergence analysis of generalized primal-dual interior-point algorithms for linear optimization*. Department of Combinatorics and Optimization, Faculty of Mathematics, Waterloo, Ontario, Canada, 2002. Thesis (M.Math.)–University of Waterloo.

[95] James H. Wilkinson. The evaluation of the zeros of ill-conditioned polynomials. part i. *Numerische Mathematik*, 1(1):150–166, 1959.

[96] Stephen J. Wright. Stability of linear equations solvers in interior-point methods. *SIAM Journal on Matrix Analysis and Applications*, 16(4):1287–1307, 1995.

[97] Stephen J. Wright. Modified Cholesky factorizations in interior-point algorithms for linear programming. *SIAM Journal on Optimization*, 9(4):1159–1191, 1999.

[98] Xiaojie Xu, Pi-fang Hung, and Yinyu Ye. A simplified homogeneous and self-dual linear programming algorithm and its implementation. *Annals of Operations Research*, 1996.

[99] Yinyu Ye, Michael J. Todd, and Shinji Mizuno. An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm. *Math. Oper. Res.*, 19(1):53–67, 1994.

Appendices

Appendix A

Proof of Theorem 2.7.3

This appendix is devoted to a proof of Theorem 2.7.3, as well as a few other concrete lemmata needed in Section 2.9.

There are a number of explicit constants in this proof. All statements are (intended to be) correct as written. In particular, when a number such as 1.234567 is written, the exact rational number $\frac{1234567}{1000000}$ is intended, as opposed to some other real number close to 1.234567.

Let F be a θ-LHSCB for some convex cone K. Let $x \in \operatorname{int} K$ and let $s \in \operatorname{int} K^*$. Let $\tilde{x} := -F'_*(s)$ and $\tilde{s} := -F'(x)$.

Let $\mu = s^T x/\theta$. Let $\delta_P := x - \mu\tilde{x}$ and let $\delta_D := s - \mu\tilde{s}$. Write $\check{s} := (s + \mu\tilde{s})/2$.

A.1 The first update

Theorem A.1.1. *Assume* $||\delta_D||_s \leq 1/50$. *Then*

1. $||F''_*(s)\delta_D||^*_s = ||\delta_D||_s$;
2. $||F''_*(\bar{s})\delta_D||^*_s \leq 1.041233\,||\delta_D||_s$;
3. *for every* $v \in \mathbb{E}^*$,
$$|\langle v, \mu F''_*(\check{s})\delta_D - \delta_P\rangle| \leq 0.265621\mu\,||v||_s\,||\delta_D||^2_s;$$
4. $||\mu F''_*(\check{s})\delta_D - \delta_P||^*_s \leq 0.265621\mu\,||\delta_D||^2_s$.

Proof. 1. This follows straightforwardly from the definitions.

2. Recall that, for every $z \in \mathbb{R}^d$,

$$F_*''(\bar{s})[z,z] \leq \frac{1}{(1-||\delta_D||)^2} F_*''(s)[z,z].$$

Now we apply the previous part. Next, we notice that $\frac{1}{(1-||\delta_D||_s)^2} \leq 1.041233$.

3. Let $f(t) = \langle u, F_*'(\check{s}+t\delta_D)\rangle$. We consider an order-two Taylor expansion of f around zero; we see that, for every $t \in [-1/2, 1/2]$, there exists a $\bar{t} \in [\min(0,t), \max(0,t)]$ such that

$$f(t) = f(0) + tf'(0) + \frac{1}{2}t^2 f''(\bar{t}).$$

Notice that

$$f''(\bar{t}) = F_*'''(\check{s}+\bar{t}\delta_D)[u, \delta_D, \delta_D] \leq 2||u||||\delta_D||^2,$$

where both norms are the local $(\check{s}+\bar{t}\delta_D)$ norms (we used self-concordance property of F_*). We then apply the Dikin ellipsoid bound to these norms to relate

$$||u||_{\check{s}+\bar{t}\delta_D} \leq \frac{1}{1-||\delta_D||_s}||u||_s$$

and similarly for $||\delta_D||$. Consequently, using $1/(1-||\delta_D||_s) \leq 1.020409$, we see that

$$|f''(\bar{t})| \leq \frac{125000}{117649}||u||_s||\delta_D||_s^2.$$

Thus, for some $\bar{t}_1 \in [-1/2, 0]$ and $\bar{t}_2 \in [0, 1/2]$, we have

$$f(1/2) - f(-1/2) = f'(0) + \frac{1}{8}(f''(\bar{t}_1) - f''(\bar{t}_2)).$$

Consequently,

$$|f'(0) - f(1/2) + f(-1/2)| \leq \frac{31250}{117649}||u||_s||\delta_D||_s^2.$$

Notice that, by substitution and the chain rule,

- $f'(0) = \langle u, F_*''(\check{s})\delta_D\rangle$;
- $f(-1/2) = \langle u, F_*'(\mu\tilde{s})\rangle = -\langle u, x/\mu\rangle$;
- $f(1/2) = \langle u, F_*'(s)\rangle = -\langle u, \tilde{x}\rangle$.

The claimed bound now follows.

4. We use the definition of a dual norm:

$$\begin{aligned} ||F_*''(\check{s})\delta_D - x/\mu + \tilde{x}||_s^* &= \sup_{||u||_s=1} \langle u, F_*''(\check{s})\delta_D - x/\mu + \tilde{x}\rangle \\ &\leq \sup_{||u||_s=1} 0.265621||u||_s||\delta_D||_s^2 = 0.265621||\delta_D||_s^2. \end{aligned}$$

□

Lemma A.1.2. *Assume* $||\delta_D||_s \leq 1/50$. *Then, the zeroth-order low-rank update has small norm:*

1. *for every* $v \in \mathbb{E}^*$,
$$|\langle v, F_*''(\check{s})[\delta_D]\rangle| \leq 1.020305\,||v||_s\,||\delta_D||_s\,;$$
2. *for every* $v \in \mathbb{E}^*$ *and* $\bar{s} \in [s, \mu\tilde{s}]$,
$$|F_*'''(\bar{s})[v, \delta_D, \mu\tilde{s}]| \leq 2.124965\,||v||_s\,||\delta_D||_s\,\sqrt{\theta};$$
3. $||Hs - x||_s^* \leq 2.082788\sqrt{\theta}\mu\,||\delta_D||_s\,;$
4. $||x||_s^* \leq 1.020409\sqrt{\theta}\mu;$
5. $||Hs||_s^* \leq 1.020305\sqrt{\theta}\mu;$
6. $||Hs + x||_s^* \leq 2.040714\sqrt{\theta}\mu;$
7. $\langle s, Hs\rangle \geq 0.980100\theta\mu.$

Proof. 1. We compute, using Cauchy-Schwarz and the Dikin ellipsoid bound,
$$\begin{aligned} & |\langle v, F_*''(\check{s})[\delta_D]\rangle| \\ = \;& |F_*''(\check{s})[\delta_D, v]| \\ \leq \;& ||\delta_D||_{\check{s}}\,||v||_{\check{s}} \\ \leq \;& 1.020305\,||\delta_D||_s\,||v||_s\,. \end{aligned}$$

2. We compute
$$\begin{aligned} & |F_*'''(\bar{s})[v, \delta_D, \mu\tilde{s}]| \\ \leq \;& 2\,||v||_{\bar{s}}\,||\delta_D||_{\bar{s}}\,||\mu\tilde{s}||_{\bar{s}} \\ \leq \;& 2.124965\,||v||_s\,||\delta_D||_s\,||\mu\tilde{s}||_{\mu\tilde{s}}\,. \end{aligned}$$

3. We write
$$Hs = \mu F_*''(\check{s})(\mu\tilde{s}) + \mu F_*''(\check{s})\delta_D.$$
On the first term, we perform a Taylor expansion around $\mu\tilde{s}$; for every v there is a $\bar{s}$ on the line segment between s and $\mu\tilde{s}$ such that
$$\begin{aligned} F_*''(\check{s})[\mu\tilde{s}, v] &= F_*''(s - \delta_D)[\mu\tilde{s}, v] + \frac{1}{2}F_*'''(\bar{s})[\mu\tilde{s}, \delta_D, v] \\ &= \langle v, x\rangle/\mu + \frac{1}{2}F_*'''(\bar{s})[\mu\tilde{s}, \delta_D, v]. \\ &\leq \langle v, x\rangle/\mu + \frac{424993}{400000}\,||v||_s\,||\delta_D||_s\,\sqrt{\theta}. \end{aligned}$$

We also bound (using the Dikin ellipsoid bound first, followed by Lemma 2.7.1)

$$\langle v, F_*''(\check{s})\delta_D\rangle \leq 1.020305||v||_s||\delta_D||_s.$$

Adding these bounds and taking a supremum over all v such that $||v||_s = 1$, since $\theta \geq 1$, yields the bound

$$||Hs - x||_s^* \leq 2.082788\sqrt{\theta}\mu||\delta_D||_s,$$

as desired.

4. This follows directly from the Dikin ellipsoid bound.

5. Notice that

$$||Hs||_s^* = \mu\left\langle s, F_*''(\check{s})(F_*''(s))^{-1}F_*''(\check{s})s\right\rangle^{1/2} \leq \mu\frac{\langle s, F_*''(s)s\rangle^{1/2}}{(1-||\delta_D||/2)^2} \leq 1.020305\sqrt{\theta}\mu.$$

6. We apply the triangle inequality to the last two parts.

7. Note that $\langle s, Hs\rangle = \mu F''(\check{s})[s,s] \geq \mu(1-\delta_D/2)^2F''(s)[s,s] \geq 0.9801\theta\mu$.

□

Theorem A.1.3. *Assume* $||\delta_D||_s \leq 1/50$. *Then*

$$\left|\left|\frac{xx^\top}{\langle s,x\rangle} - \frac{Hss^\top H}{\langle s, Hs\rangle}\right|\right|_s \leq 6.462628\mu\,||\delta_D||_s\,.$$

Proof. We write the first low-rank update as

$$\frac{xx^\top - Hss^\top H}{\langle s,x\rangle} + \left(\frac{1}{\langle s,x\rangle} - \frac{1}{\langle s, Hs\rangle}\right)Hss^\top H.$$

Then, using the triangle inequality, Lemma 2.7.2, and the bound $||vv^\top||_s \leq ||v||_s^{*2}$ we bound its norm above by

$$\frac{1}{\langle s,x\rangle}\,||x - Hs||_s^*\,||x + Hs||_s^* + \left|\frac{1}{\langle s,x\rangle} - \frac{1}{\langle s, Hs\rangle}\right|||Hs||_s^{*2}\,.$$

The first term is bounded above by

$$\frac{531296828829}{125000000000}\mu||\delta_D||_s.$$

To bound the second term, note that

$$\left|\frac{1}{\langle s,x\rangle} - \frac{1}{\langle s, Hs\rangle}\right| = \left|\frac{\langle s, Hs - x\rangle}{\theta\mu\,\langle s, Hs\rangle}\right| \leq \frac{||s||_s||Hs - x||_s^*}{\theta\mu\,\langle s, Hs\rangle} \leq 2.082788\frac{||\delta_D||_s}{\langle s, Hs\rangle}.$$

Now, we bound $\langle s, Hs\rangle$ below by $0.980100\theta\mu$ to get a bound of

$$\frac{520697}{245025}\frac{||\delta_D||_s}{\theta\mu}.$$

The bound $||Hs||_s^* \leq 1.020305\sqrt{\theta}\mu$ then gives an overall bound on the second term of

$$\frac{179192457821897}{81000000000000}\mu||\delta_D||_s$$

Adding fractions gives (something slightly stronger than) the desired bound. □

A.2 The second update

Lemma A.2.1. *Assume* $||\delta_D||_s \leq 1/50$. *Let* $H_1 := H + \frac{xx^\top}{\langle s,x\rangle} - \frac{Hss^\top H}{\langle s,Hs\rangle}$. *Then,*

1. $||H\delta_D - \delta_P||_s^* \leq 0.265621\mu\,||\delta_D||_s^2$;
2. $\langle \delta_D, H\delta_D\rangle \geq 0.960400\mu\,||\delta_D||_s^2$;
3. $\langle \delta_D, \delta_P\rangle \geq 0.955087\mu\,||\delta_D||_s^2$;
4. $||H\delta_D||_s^* \leq 1.020305\mu\,||\delta_D||_s$;
5. $||\delta_P||_s^* \leq 1.025618\mu\,||\delta_D||_s$;
6. $||H\delta_D + \delta_P||_s^* \leq 2.045923\mu\,||\delta_D||_s$;
7. $||H_1\delta_D - H\delta_D||_s^* \leq 0.276518\mu\,||\delta_D||_s^2$;
8. $||H_1\delta_D - \delta_P||_s^* \leq 0.542139\mu||\delta_D||_s^2$;
9. $||H_1\delta_D||_s^* \leq 1.025836\mu||\delta_D||_s$;
10. $||H_1\delta_D + \delta_P||_s^* \leq 2.051454\mu||\delta_D||_s$;
11. $\langle \delta_D, H_1\delta_D\rangle \geq 0.944244\mu||\delta_D||_s^2$.

Proof. 1. This was proven in Theorem A.1.1, part (4).

2. This follows from the Dikin ellipsoid bound; $H \succeq 0.960400\mu F_*''(s)$.

3. Notice that
$$\langle \delta_D, \delta_P \rangle = \langle \delta_D, H\delta_D \rangle + \langle \delta_D, \delta_P - H\delta_D \rangle .$$
We bound the second term by Cauchy-Schwarz:
$$\langle \delta_D, \delta_P - H\delta_D \rangle \leq ||\delta_D||_s ||H\delta_D - \delta_P||_s^* \leq 0.265621\mu ||\delta_D||_s^3.$$
Using this with the bound from the previous part gives the advertised inequality.

4. We compute, using Lemma 2.7.1
$$\begin{aligned} ||H\delta_D||_s^* = \mu \left\langle \delta_D, F_*''(\check{s}) \left(F_*''(s)\right)^{-1} F_*''(\check{s})\delta_D \right\rangle^{1/2} &\leq 1.020305\mu \langle \delta_D, F_*''(s)\delta_D \rangle^{1/2} \\ &= 1.020305\mu ||\delta_D||_s, \end{aligned}$$
as advertised.

5. We use the triangle inequality followed by parts (1) and (4):
$$\begin{aligned} & ||\delta_P||_s^* \\ \leq\ & ||H\delta_D||_s^* + ||\delta_P - H\delta_D||_s^* \\ \leq\ & 1.020305\mu \, ||\delta_D||_s + 0.265621\mu \, ||\delta_D||_s^2 \\ \leq\ & 1.025618\mu \, ||\delta_D||_s . \end{aligned}$$

6. We use the triangle inequality, part (4) and the bound $||\delta_D||_s \leq 1/50$:
$$\begin{aligned} & ||H\delta_D + \delta_P||_s^* \\ \leq\ & 2||H\delta_D||_s^* + ||H\delta_D - \delta_P||_s^* \\ \leq\ & 2.040610\mu ||\delta_D||_s + 0.265621\mu ||\delta_D||_s^2 \\ \leq\ & 2.040610\mu ||\delta_D||_s + \frac{265621}{50000000}\mu ||\delta_D||_s, \end{aligned}$$
which is the claimed bound.

7. Recall that $\langle \delta_D, x \rangle = 0$, so
$$H\delta_D - H_1\delta_D = \frac{\langle s, H\delta_D \rangle}{\langle s, Hs \rangle} Hs.$$
Now, we bound using $\langle s, \delta_P \rangle = 0$, triangle inequality and part (1):
$$| \langle s, H\delta_D \rangle | = | \langle s, \delta_P \rangle + \langle s, H\delta_D - \delta_P \rangle | \leq 0 + ||s||_s ||H\delta_D - \delta_P||_s^* \leq 0.265621\sqrt{\theta}\mu ||\delta_D||_s^2$$
and recall (Lemma A.1.2 part (7))
$$\langle s, Hs \rangle \geq 0.980100\theta\mu$$

and (Lemma A.1.2 part (5))

$$||Hs||_s^* \leq 1.020305\sqrt{\theta}\mu.$$

Thus,

$$||H_1\delta_D - H\delta_D||_s^* \leq 0.276518\mu||\delta_D||_s^2.$$

8. We use the triangle inequality followed by parts (1) and (7).

9. We use the triangle inequality followed by parts (4), (7) and the fact that $||\delta_D||_s \leq 0.020000$.

10. We use the triangle inequality and parts (5) and (9).

11. We compute, using previous parts of this lemma,

$$\begin{aligned}\langle \delta_D, H_1\delta_D\rangle &= \langle \delta_D, \delta_P\rangle + \langle \delta_D, H_1\delta_D - \delta_P\rangle \\ &\geq 0.955087\mu\,||\delta_D||_s^2 - ||\delta_D||_s\,||H_1\delta_D - \delta_P||_s^* \\ &\geq 0.955087\mu||\delta_D||_s^2 - 0.542139\mu||\delta_D||_s^3 \\ &\geq 0.944244\mu||\delta_D||_s^2.\end{aligned}$$

□

Theorem A.2.2. *Assume* $||\delta_D||_s \leq 1/50$*, and take* $H := \mu F_*''(\check{s})$ *and* $H_1 := H + \frac{xx^\top}{\langle s,x\rangle} - \frac{Hss^\top H}{\langle s,Hs\rangle}$*. Then*

$$\left|\left|\frac{\delta_P\delta_P^\top}{\langle \delta_D, \delta_P\rangle} - \frac{H_1\delta_D\delta_D^\top H_1}{\langle \delta_D, H_1\delta_D\rangle}\right|\right|_s^* \leq 1.797089\mu||\delta_D||_s.$$

Proof. Write

$$\begin{aligned}&\frac{\delta_P\delta_P^\top}{\langle \delta_D, \delta_P\rangle} - \frac{H_1\delta_D\delta_D^\top H_1}{\langle \delta_D, H_1\delta_D\rangle} \\ =\ & \frac{1}{\langle \delta_D, \delta_P\rangle}\left(\delta_P\delta_P^\top - H_1\delta_D\delta_D^\top H_1\right) + \left(\frac{1}{\langle \delta_D, H_1\delta_D\rangle} - \frac{1}{\langle \delta_D, \delta_P\rangle}\right) H_1\delta_D\delta_D^\top H_1.\end{aligned}$$

Notice that

$$\begin{aligned}\left|\frac{1}{\langle \delta_D, H_1\delta_D\rangle} - \frac{1}{\langle \delta_D, \delta_P\rangle}\right| = \left|\frac{\langle \delta_D, H_1\delta_D - \delta_P\rangle}{\langle \delta_D, H_1\delta_D\rangle\,\langle \delta_D, \delta_P\rangle}\right| &\leq \frac{||\delta_D||_s||H_1\delta_D - \delta_P||_s^*}{|\,\langle \delta_D, H_1\delta_D\rangle\,\langle \delta_D, \delta_P\rangle\,|} \\ &\leq \frac{45178250000}{75152930769}\frac{1}{\mu||\delta_D||_s}.\end{aligned}$$

Further, recall that $||H_1\delta_D||_s^* \leq 1.025836\mu||\delta_D||_s$. Thus, the second term's norm is bounded above by $0.632615\mu||\delta_D||_s$. Using the lower bound on $\langle \delta_D, \delta_P\rangle$ and the upper bounds on $||H_1\delta_D - \delta_P||_s^*$ and $||H_1\delta_D + \delta_P||_s^*$, we get a bound on the first term's norm of

$$\frac{0.542139 \cdot 2.051454}{0.955087}\mu||\delta_D||_s \leq 1.164474\mu||\delta_D||_s.$$

Adding the bounds on the two terms together gives the advertised bound. □

Theorem A.2.3. *Assume* $||\delta_D||_s \leq 1/50$, *and take* $H := \mu F''_*(\check{s})$,

$$H_1 := H + \frac{xx^\top}{\langle s, x\rangle} - \frac{Hss^\top H}{\langle s, Hs\rangle}, \text{ and}$$

$$T^2 := H_1 + \frac{\delta_P \delta_P^\top}{\langle \delta_D, \delta_P\rangle} - \frac{H_1 \delta_D \delta_D^\top H_1}{\langle \delta_D, H_1 \delta_D\rangle}.$$

Then $||T^2 - H|| \leq 8.259717\mu \, ||\delta_D||_s \leq 0.165195\mu$.

Proof. We consider the two rank-two updates separately; Theorem A.1.3 controls the size of the first update and Theorem A.2.2 controls the size of the second update. We simply add the two bounds together. □

This implies Theorem 2.7.3, stated again below:

Corollary A.2.4. *If* $||\delta_D||_s \leq 1/50$, *then there exists a* $T \in \mathbb{S}^d$ *satisfying the following properties:*

- T *is positive definite;*
- $T^2 s = x$*;*
- $T^2 \tilde{s} = \tilde{x}$*;*
- *For every* $z \in \mathbb{R}^d$, $0.814905\mu F''_*(s)[z,z] \leq T^2[z,z] \leq 1.185500\mu F''_*(s)[z,z]$
- *For every* $z \in \mathbb{R}^d$, $\frac{0.808093}{\mu} \left(F''(x)\right)^{-1}[z,z] \leq T^2[z,z] \leq \frac{1.192311}{\mu} \left(F''(x)\right)^{-1}[z,z]$.

That is, $T \in \mathcal{T}_2(1.237483; x, s)$.

Note that in the above analysis, we did not utilize the additional flexibility provided by the term $(\mu\tilde{\mu} - 1)$. This establishes, in the language of [85], that ξ^* is $O(1)$ within a particular neighbourhood of the central path. Moreover, our specific choice T_H is in $\mathcal{T}_2(\eta; x, s)$ for $\eta = O(1)$, for every pair (x, s) that is in the same neighbourhood.

Therefore, Theorem 5.1 of [85] implies that a wide range of potential reduction algorithms (whose iterates are restricted in a neighbourhood of the central path) have the iteration complexity of $O\left(\sqrt{\theta} \ln(1/\epsilon)\right)$.

A.3 Bounds in v-space

The following lemma is useful to the convergence analysis in Section 2.9.

Lemma A.3.1. *Suppose $x \in \text{int}(K)$ and $s \in \text{int}(K^*)$ are such that $||\delta_D||_s < 1/50$. Take T^2 as in Theorem 2.7.3 and take T to be its self-adjoint positive-definite square root. Let $v := Ts = T^{-1}x$ and $\delta_v := T\delta_D$. Let z be an arbitrary vector in v-space. Let $x' \in \text{int}(K)$ and $s' \in \text{int}(K^*)$; define $\delta'_D := s' + \mu F'(x')$ and $\delta'_v := T\delta'_D$. Then,*

1. $||Tz|| \leq 1.088807\sqrt{\mu}\,||z||_s$;
2. $||z||_s \leq 1.107763\,||Tz||\,/\sqrt{\mu}$;
3. $||\delta_v|| \leq 0.021777\sqrt{\mu}$;
4. $||\delta'_D||_{s'} \leq \frac{||\delta'_D||_s}{1-||s-s'||_s}$;
5. *if* $||\delta'_v|| \leq 0.006527\sqrt{\mu}$ *and* $||s-s'||_s \leq 1/25$, *then* $||\delta'_D||_{s'} \leq 0.007533$;
6. *if* $||\delta'_v|| \leq 0.017330\sqrt{\mu}$ *and* $||s-s'||_s \leq 1/25$, *then* $||\delta'_D||_{s'} \leq 1/50$.

Proof. Recall from Theorem 2.7.3 that

$$0.814905\mu||z||_s^2 \leq ||Tz||^2 \leq 1.185500\mu||z||_s^2. \tag{A.1}$$

1. This is the square root of $||Tz||^2 \leq 1.185500\mu\,||z||_s^2$ with a constant rounded up.
2. This is the square root of $\mu\,||z||_s^2 \leq \frac{200000}{162981}\,||Tz||^2$ with a constant rounded up.
3. Take $z = \delta_D$ in part (1) and then use $||\delta_D||_s \leq 1/50$; we see
$$||\delta_v|| < 1.088807\sqrt{\mu}||\delta_D||_s \leq 0.021777\sqrt{\mu},$$
as desired.
4. This is the Dikin ellipsoid bound for comparing the s'-norm with the s-norm.
5. If $||\delta'_v|| \leq 0.006527\sqrt{\mu}$, then by part (2) $||\delta'_D||_s \leq 0.007231$. By part (4), then, $||\delta'_D||_{s'} \leq 0.007379$, as desired.
6. If $||\delta'_v|| \leq 0.017330\sqrt{\mu}$, then by part (2) $||\delta'_D||_s \leq 0.019198$. By part (4), then, $||\delta'_D||_{s'} \leq 0.019998$, which implies the desired result.

□