Iterative Refinements and Zoom: Acceleration and Warmstarting of Interior Point Methods

Leo Tenenblat

June 25, 2008

Contents

1 Introduction **1**
- 1.1 The Problem . . . 1
- 1.2 Interior Point Methods vs. Active Set Methods . . . 2
- 1.3 Difficulties with IPMs . . . 2

2 An Overview of Interior Methods **4**
- 2.1 Basic elements . . . 4
  - 2.1.1 Duality . . . 4
  - 2.1.2 Optimality Conditions for Constrained Optimization . 6
  - 2.1.3 Newton's Method for Nonlinear Equations . . . 11
- 2.2 IP algorithms/strategies . . . 13
  - 2.2.1 Affine-scaling . . . 14
  - 2.2.2 Path-following . . . 16
  - 2.2.3 Potential-reduction . . . 20
- 2.3 Theory of Barrier-type Methods . . . 22
  - 2.3.1 Convexity and Self-Concordance . . . 22
  - 2.3.2 Convergence and Complexity of Self-Concordant Barrier Methods . . . 23
- 2.4 Implementation Issues . . . 26
  - 2.4.1 Path-Following Methods . . . 26
  - 2.4.2 Warmstarting Interior Methods . . . 29

3 PDCO **32**
- 3.1 Regularization . . . 33
- 3.2 The Barrier Approach . . . 33
- 3.3 Newton's Method . . . 35
- 3.4 Solving for $(\Delta x, \Delta y)$ . . . 35

3.5 Scaling 36

4 The Refine and Zoom Technique **38**
4.1 Motivation 38
4.2 Refinement 39
4.3 Revisions to PDCO 42
4.3.1 Changes to Newton's Method 42
4.3.2 Changes to Scaling 43
4.4 Scaling (outside PDCO) 44
4.5 Zoom 45
4.6 Convergence and Complexity 46

5 Accelerating IPMs **48**
5.1 Motivation 48
5.2 Why Zoom and Refine Works 50
5.3 Numerical Results 52

6 LP Warmstarting **55**
6.1 Motivation 55
6.2 LP Zoomstarting/Jumpstarting 57
6.3 Numerical Results 58

7 Appendix **76**

Chapter 1

Introduction

In this chapter we exhibit the class of problems this thesis covers and give an overview of the interior point method, its strengths and weaknesses. We also outline here the structure of the thesis.

1.1 The Problem

We concern ourselves throughout this thesis primarily with the following type of problem:

$$\begin{array}{|lll|}\hline \text{NP} & \underset{x}{\text{minimize}} & \phi(x) \\ & \text{subject to} & Ax = b, \quad \ell \leq x \leq u, \\ \hline \end{array}$$

where $\phi(x)$ is a convex function with known gradient $g(x)$ and Hessian $H(x)$, and $A \in \mathbb{R}^{m \times n}$. The format of NP is suitable for any linear constraints. For example, a double-sided constraint $\alpha \leq a^T\widetilde{x} \leq \beta$ $(\alpha < \beta)$ should be entered as $a^T\widetilde{x} - \xi = 0$, $\alpha \leq \xi \leq \beta$, where $\widetilde{x}$ and ξ are relevant parts of x.

Many real-world problems can be modeled through the NP framework. Electronic circuit design problems or automatic control systems, for example, can be structured in the form above. Also, linear programs (LP) are a subset of NP that have numerous classic examples derived from industry.

Theoretical results are often derived within a more generalized context where the linear constraints are replaced by convex ones. We explain those problems in detail where applicable.

1.2 Interior Point Methods vs. Active Set Methods

Interior Point Methods (IPMs) reached a peak of development (under a different name) with the classic work of Fiacco and McCormick [12]. Karmarkar's algorithm [30] is considered to be the first algorithm of this kind intended for large problems and has been enhanced and adapted over the years. They fell from favor for some 10 years, but were reinvented in disguised form as a response to the poor complexity properties of active-set methods such as the Simplex Method for linear programming. IPMs are nowadays considered essential to every optimization algorithms suite, better suited for solving certain LPs than Simplex, while also able to tackle nonlinear problems. The Simplex Method, on the other hand, has also been adapted for use with nonlinear problems.

The main idea behind IPMs is to specify a series of simpler perturbed problems whose solutions x^k are always in the interior of the feasible region ($l < x^k < u$). As the perturbation is decreased toward zero, the sequence x^k follows a path leading toward feasibility and optimality for the original problem. This differs from active-set methods, where iterates move between adjacent feasible solutions on intersections of a set of active constraints, with constraints entering or leaving the active set in order to improve the objective until no such improvement is possible. In this difference lies the essence of the advantages and disadvantages of each approach. We focus mainly here on the analysis of IPMs, noting that [54] has a more comprehensive comparison of the two classes of algorithms. While IPMs have better complexity properties, since they do not face the combinatorial issue of the choice of active constraints, their iterations are much more costly. Also, because of the interior quality of its iterates, IPMs can encounter numerical difficulties as the iterates approach the constraint boundaries near the solution, which normally resides on some of them.

1.3 Difficulties with IPMs

Typical IPM implementations exploit sparsity patterns and efficient matrix factorizations to speed up the linear algebra in each iteration. However, for problems with a dense constraint matrix or when the matrix is expressed through an operator, it becomes necessary for an IPM to employ iterative

solvers in the subalgorithm for computing search directions. Image restoration and basis pursuit are examples of such problems. In these situations, there can be a drastic increase in computational effort as the iterates approach the solution. This happens because as the distance to some of the boundaries approaches zero, the linear system defining each search direction becomes increasingly ill-conditioned, causing a drastic increase in the number of iterations required by the subalgorithm solver to achieve adequate accuracy.

Another well-known deficiency of IPMs is their stalling behavior in problem restarts. In real world scenarios, it is commonly required to solve a series of closely related problems. Industrial models with varying capacities, or economics problems with changing demands, are examples of such situations. Frequently the solutions of such problems are also closely related, so one would expect solvers to require less computational effort on one such problem if starting from a solution to another in the same group. *Warm-starting*, as it is called, is common practice with active-set methods such as Simplex but is notoriously problematic in IPMs, again because of numerical issues related to boundary proximity of the iterates.

These difficulties and possible solutions to them are the main focal points of this dissertation.

Chapter 2

An Overview of Interior Methods

In this chapter we provide a comprehensive overview of the theory of constrained optimization, interior point methods, and the numerical implementation of such methods. For didactical purposes we introduce several of these elements within the framework of linear programming, giving references wherever possible to more in-depth analysis of their nonlinear counterparts.

Given the scope of this subject, the several topics underneath it, and the flurry of new research published in recent years on many of these topics, detailed all-encompassing surveys as well as topic-specific thorough discussions are widely available in the literature, for example [17, 23, 20, 35]. Many linear and nonlinear programming books also contain chapters on interior methods that include many of the latest results and unified viewpoints on theory, presented in didactical fashion (e.g. [56, 4, 5].)

2.1 Basic elements

2.1.1 Duality

Duality plays an important role in the understanding of interior methods. From simple nomenclature to the underlying theory and implementation involved in specific algorithms, elements of this concept appear throughout this thesis.

In the context of linear programming, (NP) can be represented more

succinctly as

LP-Primal	$\underset{x\in\mathbb{R}^n}{\text{minimize}}$	$c^T x$
	subject to	$Ax = b, \quad x \geq 0,$

where $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$. To this primal problem, there is a closely related problem, using the same data, which is called the Dual:

LP-Dual	$\underset{y\in\mathbb{R}^m}{\text{maximize}}$	$b^T y$
	subject to	$A^T y \leq c.$

LP-Dual can also be represented in a similar form to LP-Primal, without loss of generality, as follows:

LP-Dual'	$\underset{y\in\mathbb{R}^m}{\text{maximize}}$	$b^T y$
	subject to	$A^T y + z = c, \quad z \geq 0,$

where z is commonly referred to as the dual *slack* vector. We can assume that A has full row rank, since linear dependence among the constraints would imply either an infeasible problem or a redundant constraint. This implies a 1-1 correspondence between y and z in the dual feasible set, so we refer to a dual point simply as y.

The following theorem establishes the relationship between these two problems.

Theorem 1 (*Fundamental Theorem of Duality*)**.** *Suppose x and y are feasible for LP-Primal and LP-Dual respectively.*

(i) Weak duality. *The objective values satisfy $c^T x \geq b^T y$. In other words, a feasible primal point provides an upper bound for the objective of the dual problem and, likewise, a feasible dual point provides a lower bound for the primal objective. This property has the important consequence that if either problem is unbounded, the other is infeasible. Note that since the implication is unidirectional, it is still possible for both to be infeasible.*

(ii) Strong duality. *x and y are optimal if and only if $c^T x = b^T y$. The non-negative quantity $x^T z = x^T(c - A^T y) = c^T x - b^T y$ is known as*

the duality gap. *From this property, if one problem has a finite optimal solution, so does the other, and their optimal objective values are equal. When this happens, clearly* $x^T z = 0$*, a condition known as* complementary slackness*.*

Though most of these properties can be extended in a straightforward manner to the nonlinear case, it is important to note that the duality gap is not always zero as it is for LPs. Such a guarantee in the nonlinear case requires certain convexity properties of the objective and constraint equations as we see below.

2.1.2 Optimality Conditions for Constrained Optimization

The theory of constrained optimization covers the very general class of problems represented by

NP	$\underset{x}{\text{minimize}}$	$\phi(x)$
	subject to	$c_i(x) \geq 0, \quad i = 1, \ldots, m,$

where we assume throughout that ϕ and c_i are twice-continuously differentiable and convex. We denote the gradient and Hessian of ϕ by $g(x)$ and $H(x)$ respectively, where

$$\begin{aligned} g(x) &= \nabla\phi(x) \\ H(x) &= \nabla^2\phi(x) \\ H_i(x) &= \nabla^2 c_i(x). \end{aligned}$$

Analogously, the gradient and Hessian of $c_i(x)$ are denoted by $\nabla c_i(x)$ and $H_i(x)$, where

$$H_i(x) = \nabla^2 c_i(x),$$

while $J(x)$ is used for the $m \times n$ Jacobian of the constraints, whose i-th row is $\nabla c_i(x)$.

Optimality conditions for (NP) are inextricably linked to what is widely known in literature as the KKT conditions (or KKT point). The term stands for "Karush-Kuhn-Tucker", who identified (Karush in 1939 and Kuhn-Tucker separately in 1951) the optimality properties of inequality constrained problems that we now state.

Definition 1. *(KKT point) x^* is a (first-order) KKT point for (NP) if there exists an m-vector λ^*, called a* Lagrange multiplier vector*, such that*

(i) $c(x^*) \geq 0$ *(feasibility)*

(ii) $g(x^*) = J(x^*)^T \lambda^*$ *(optimality)*

(iii) $\lambda^* \geq 0$ *(nonnegativity of the multipliers), and*

(iv) $c_i(x^*)\lambda_i^* = 0,\ i = 1, \ldots, m$ *(complementarity).*

It is worth noting that KKT points are also saddle points of the Lagrangian function $L(x) = L(x, \lambda) = \phi(x) + \sum_{i=1}^{m} \lambda_i c_i(x)$. That is, we have the following result.

Proposition 1.

$$(x^*, \lambda^*) \in S^* \iff L(x^*, \lambda) \leq L(x^*, \lambda^*) \leq L(x, \lambda^*), \quad \forall\ x \in F, \lambda \in \mathbb{R}^m_+, \tag{2.1}$$

where S^* is the set of points (x^*, λ^*) satisfying the KKT conditions and $F = \{x \in \mathbb{R}^m; c_i(x) \geq 0\ \ \forall\ i = 1, \ldots, m\}$ represents the feasible set.

The following notation will also be useful later:

$$\begin{aligned} g_L(x, \lambda) &= \nabla_x L(x, \lambda) \\ H_L(x, \lambda) &= \nabla^2_{xx} L(x, \lambda). \end{aligned}$$

The Lagrange multipliers correspond to the dual variables associated with the problem, and their existence depends on the duality gap being zero, as the result below states. Analogously to the linear case, the duality gap is zero for nonlinear problems when $\sup_{\lambda \geq 0}(\inf L(x)) = \inf \phi(x)$.

Proposition 2.

(i) If there is no duality gap, the set of Lagrange multipliers is equal to the set of optimal dual solutions.

(ii) If there is a duality gap, the set of Lagrange multipliers is empty.

We now formally introduce some definitions that are used throughout this thesis.

Definition 2. *(active, inactive and violated constraints) A constraint* $c_i(x) \geq 0$ *is said to be* active *at a point* $\bar{x}$ *if* $c_i(\bar{x}) = 0$. *We denote by* $A(\bar{x}) = \{i \in \{1, \ldots, m\}; c_i(\bar{x}) = 0\}$ *the set of indices of the active constraints at* $\bar{x}$. *Similarly,* $c_i(x) \geq 0$ *is said to be* inactive *if* $c_i(\bar{x}) \neq 0$ *(and we denote by* $I(\bar{x}) = \{1, \ldots, m\} \setminus A(\bar{x})$ *the set of indices of inactive constraints), and* violated *if, in particular,* $c_i(\bar{x}) < 0$.

From the complementarity property (iv) and the definition above, if a constraint $c_i(x)$ is inactive then its multiplier λ_i is zero. That has an immediate implication: $\lambda_I^* = 0$ and hence (ii) can be reduced to $g(x^*) = J_A(x^*)^T \lambda_A^*$. If $c_i(x)$ is active, however, nothing can be said of λ_i. The following property excludes the possibility of λ_i and $c_i(x)$ being simultaneously zero.

Definition 3. *(strict complementarity) Strict complementarity holds at a KKT point* x^* *if there exists a multiplier* $\lambda^* \in M_\lambda$ *such that* $\lambda_i^* > 0 \ \ \forall\, i \in A$,

where $M_\lambda = M_\lambda(x^*) = \{\lambda \in \mathbb{R}^m; g(x^*) = J(x^*)^T \lambda, \lambda \geq 0$, and $c_i(x^*)\lambda_i = 0, \ \ \forall\, i = 1 \ldots, m\}$ is the set of acceptable multipliers..

The Lagrange multipliers in Definition 1 are not guaranteed to exist (see [16].) For such a guarantee we need an additional condition known as a *constraint qualification* (CQ). We now state the most commonly encountered CQs in the literature.

Definition 4. *(LICQ) The linear independence constraint qualification (LICQ) holds at the feasible point* $\bar{x}$ *if the problem is linear or if the Jacobian of the active constraints at* $\bar{x}$ *has full row rank, i.e. if the gradients of the active constraints are linearly independent.*

Definition 5. *(MFCQ) The Mangasarian-Fromovitz constraint qualification (MFCQ) holds at the feasible point* $\bar{x}$ *if it is strictly feasible, i.e. if* $c_i(\bar{x}) > 0$ *for all* i, *or if there exists a vector* p *such that* $\nabla c_i(\bar{x})^T p > 0$ *for all* $i \in A(\bar{x})$, *i.e. if* $J_A(\bar{x})p > 0$.

Definition 6. *(SLCQ) The Slater constraint qualification (SLCQ) holds at the feasible point* $\bar{x}$ *if it is strictly feasible.*

While the MFCQ is a weaker condition than the other two, it is more difficult to verify in practice; doing so requires solving a linear program. The LICQ is therefore frequently checked for instead, by performing rank-revealing factorizations, which are still nontrivial but require less work than

fully solving an LP (see [16] for a recent discussion about constraint qualifications).

In order to introduce optimality conditions, we first define formally the notion of optimality.

Definition 7. *(local constrained minimizer) The point x^* is a local constrained minimizer of (NP) if it is feasible ($x^* \in F$) and there exists a compact set S such that*

$$x^* \in int(S) \text{ and } \phi(x^*) = \min \phi(x) \ \ \forall\ x \in S \cap F.$$

First-order necessary conditions for constrained minimization now follow.

Theorem 2 (*first-order necessary conditions for a local constrained minimizer*)**.** *Suppose that x^* is a local minimizer of (NP) at which the MFCQ holds. Then x^* must be a KKT point.*

Local maxima can also satisfy the KKT conditions above. However, second-order properties are frequently as important, when convexity of the problem at hand is not guaranteed.

The (first-order) conditions stated above are also sufficient only when (NP) is linear. Nonlinear optimization requires additional conditions on the behavior of the Hessians of the constraints around the minimizer.

Theorem 3 (*second-order necessary conditions for a constrained minimizer*)**.** *Suppose that x^* is a local constrained minimizer of (NP) at which the LICQ holds. Then there is a vector λ^* that satisfies $\lambda^* \geq 0$, $c^{*T}\lambda^* = 0$, and $g^* = J^{*T}\lambda^*$, and further, it must be true that*

$$p^T H(x^*, \lambda^*)p \geq 0 \text{ for all } p \text{ satisfying } J_A^* p = 0.$$

Without the use of constraint qualifications, the most that can be proved is that a point x^* satisfying first- and second-order sufficient conditions is a strict local constrained minimizer. Proof for the following result can be found in any of [13, 15, 43, 48].

Theorem 4 (*sufficient conditions for a strict constrained minimizer*)**.** *The point x^* is a strict constrained minimizer of (NP) if*

(i) x^* is a KKT point, i.e. $c^* \geq 0i$, and there exists a nonempty set M_λ of multipliers λ satisfying $\lambda \geq 0, c^{*\lambda} = 0$, and $g^* = J^{*T}p$;

(ii) for all $\lambda \in M_\lambda$ *and all nonzero* p *satisfying* $g^{*T}p = 0$ *and* $J_A^* p \geq 0$, *there exists* $\omega > 0$ *such that* $p^T H(x^*, \lambda)p \geq \omega \|p\|^2$.

Simple examples can be constructed (see [14]) where the notion of a strict minimizer does not seem adequate. Rapid oscillations of the objective in a neighborhood can yield multiple strict minimizers, producing the effect of a continuous set of solutions. Formalizing this notion, to ensure a minimizer x^* is *isolated*, i.e. there exists a neighborhood $N \ni x^*$ such that no other minimizers exist in $N \setminus x^*$, a constraint qualification is also required, as the next theorem states.

Theorem 5 (*sufficient conditions for an isolated constrained minimizer*)**.** *The point* x^* *is an isolated local constrained minimizer of (NP) if*

(i) x^* *is a KKT point, i.e.* $c^* \geq 0$ *and there exists a nonempty set* M_λ *of multipliers* λ *satisfying* $\lambda \geq 0, c^* \cdot \lambda = 0$, *and* $g^* = J^{*T}p$;

(ii) the MFCQ holds at x^*, *i.e. there is a vector* p *such that* $J_A^* p > 0$;

(iii) for all $\lambda \in M_\lambda$ *and all nonzero* p *satisfying* $g^{*T}p = 0$ *and* $J_A^* p \geq 0$, *there exists* $\omega > 0$ *such that* $p^T H(x^*, \lambda)p \geq \omega \|p\|^2$.

As mentioned above, since the MFCQ is sometimes not practical to verify, the following variant, more practical but also more restrictive, is often used instead.

Theorem 6 (*strong sufficient conditions for an isolated constrained minimizer*)**.** *The point* x^* *is an isolated local constrained minimizer of (NP) if*

(i) x^* *is feasible and the LICQ holds at* x^*, *i.e.* $J_A(x^*)$ *has full row rank;*

(ii) x^* *is a KKT point and strict complementarity holds, i.e. the (necessarily unique) multiplier* λ^* *has the property that* $\lambda_i^* > 0 \ \ \forall\, i \in A(x^*)$;

(iii) for all nonzero vectors p *satisfying* $J_A x^* p = 0$ *there exists* $\omega > 0$ *such that* $p^T H(x^*, \lambda)p \geq \omega \|p\|^2$.

In 2000, Bertsekas and Ozdaglar [3] portrayed the issue of optimality and existence of Lagrange multipliers under new light, through the use of an enhanced version of the Fritz-John optimality conditions and a new principle unifying the constraint qualifications described above.

Theorem 7 (*enhanced Fritz-John conditions*)**.** *If x^* is a local minimizer of (NP) then there exist $\mu_0 \geq 0$ and $\lambda_i^* \geq 0,\ i = 1, \ldots, m$ not all zero such that*

(i) $\mu_0 g(x^*) = J(x^*)^T \lambda^*$;

(ii) if $\lambda_j^* > 0$ for some j, then there exists a sequence x^k with $x^k \to x^*$ such that

(a) $\phi(x^k) < \phi(x^*)$ *for all* k;

(b) $c_i(x^k)\lambda_i^* > 0$ *for all* i *with* $\lambda_i^* > 0$ *and all* k.

While perhaps not as easy to deal with from a practical standpoint as LICQ for instance, but theoretically more elegant, the principle that unifies the constraint qualifications is that of pseudonormality, now defined.

Definition 8. *(pseudonormality) The point x^* is* pseudonormal *if there is no vector λ^* and sequence $\{x^k\}$ such that*

(i) $J(x^*)^T \lambda^* = 0$;

(ii) $\lambda_i^* \geq 0$ *for all* $i = 1, \ldots, m$ *and* $\lambda_I^* = 0$;

(iii) $x^k \to x^*$ *and* $\sum_{i=1}^m c_i(x^k)\lambda_i^* > 0$ *for all* k.

The consequence of the above definition is clear: if x^* is pseudonormal, then we can take $\mu_0 = 1$ in the Fritz-John conditions, thus guaranteeing the existence of some $\lambda_i^* > 0$ satisfying condition (ii)-(b) in Theorem 7. Such multipliers are called *informative* because they have inherent sensitivity information, indicating which constraints to violate in order to improve the objective value. The proof that all constraint qualifications described herein (and other generalizations of them) imply pseudonormality is also contained in [3].

2.1.3 Newton's Method for Nonlinear Equations

Newton's method is widely used to compute zeros of nonlinear equations. We describe the method here briefly and highlight its applicability to optimization.

Given a nonlinear, continuously differentiable function $F : \mathbb{R}^n \to \mathbb{R}^n$ and a starting point x^0, the equation $F(x) = 0$ can be solved by successively

computing the zero of an approximation of F at the current iterate. More explicitly, the *Newton step* p^k from an iterate x^k where $F'(x^k)$ is nonsingular is defined as the vector such that $x^{k+1} = x^k + p^k$ is a zero of the first-order (linear) approximation of F at x^k:

$$0 = F(x^{k+1}) = F(x^k) + F'(x^k)p^k.$$

From the discussion in the previous section, we know that given a twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, a necessary optimal condition is $\nabla f(x^*) = 0$. Applying Newton's method to this condition (that is, taking $F(x) = \nabla f(x)$ in the previous example) we can construct a method for computing potential unconstrained optimizers of $f(x)$. If $\nabla^2 f(x^k)$ is positive definite, then each step has a unique minimizer. In this case, our scheme finds zeros of the local quadratic Taylor-series model, giving

$$0 = \nabla f(x^{k+1}) = \nabla f(x^k) + \nabla^2 f(x^k)p^k.$$

While situations can be construed where Newton's method diverges, under suitable conditions, it converges quadratically. The formal convergence result for the pure Newton method now follows.

Theorem 8. *If $F : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable, Lipschitz-continuous in a neighborhood N of x^* where $F(x^*) = 0$, and, if $x^0 \in N$, where $\forall x \in N, \nabla F(x)$ is positive definite, then starting from x^0, Newton's method converges quadratically to x^*.*

Since the result above is rather restrictive with respect to the initial point x^0 and in practice a generic starting point is preferable, Newton's method is customarily implemented with a line search as a globalization strategy. Then, instead of the pure method described above, we have

$$x^{k+1} = x^k + t^k p^k,$$

where $t^k > 0$ is calculated in such a way that there is a decrease of some merit function (e.g. $\|\nabla f(x^{k+1})\| < \|\nabla f(x^k)\|$.)

One frequently used line search that is simple and effective is the *backtracking* line search. It is based on two constants $0 < \alpha < .5$ and $0 < \beta < 1$. Given a descent direction p^k, it starts off with $t = 1$ and, while $F(x^{k+1}) > F(x^k) + \alpha t \nabla F(x^k)^T p^k$, t is reset to $t := \beta t$.

A convergence result for this damped method follows.

Theorem 9. *We assume the use of Newton's method with backtracking line search, and that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and the starting point $x^0 \in S$, such that f is strongly convex with constant m in S, i.e. $\nabla^2 f(x) \geq mI, \forall x \in S$. In addition, we assume that the Hessian of f is Lipschitz-continuous in S with constant L. Then there exist numbers η, γ with $\gamma > 0$ and $0 < \eta \leq m^2/L$, such that the following hold.*

- *If $\|\nabla f(x^k)\|_2 \geq \eta$, then*

$$f(x^{k+1}) - f(x^k) \leq -\gamma.$$

- *If $\|\nabla f(x^k)\|_2 < \eta$, then the line search selects $t = 1$ and*

$$\frac{L}{2m^2}\|\nabla f(x^{k+1})\|_2 \leq (\frac{L}{2m^2}\|\nabla f(x^k)\|_2)^2.$$

Essentially what the result above states is that this Newton-type method has two phases: the first, called the *damped* phase, in which $t < 1$ is chosen, and the second pure phase, in which $t = 1$ and, as the previous result states, convergence is quadratic.

2.2 IP algorithms/strategies

Interior methods have several variants with differing strategies aimed at theoretical or practical improvements. All share the property that iterates remain in the interior of some space, typically $l < x < u$ and $c_i(x) > 0$ for inequality constrained problems. The variations among interior methods can be grouped into the following categories: iterate space, iterate type, step type, and algorithm type.

- Iterate space. Methods that have iterates satisfying primal constraints and that treat dual variables as dependent are called *primal methods*. Analogously, dual methods have iterates in the dual space. Finally, primal-dual methods treat both as independent and their iterates belong to the Cartesian product of the primal and dual spaces.

- Iterate type. Two variants are possible here: feasible and infeasible methods. Both methods' iterates must satisfy the inequality constraints at all times. However, while iterates in feasible methods must also satisfy equality constraints, the same is not required of infeasible methods.

- Step type. Theoretical results in polynomial complexity often require some methods to take short steps at each iteration. Though these *short-step methods* posess polynomial bounds on the number of iterations, the polynomial coefficients can be quite large, leading to potentially high iteration counts. In contrast, long-step methods can and most often do perform better in practice. Variants of the latter type are the only ones seen in current implementations of interior methods.

- Algorithm type. Lastly, but most importantly, the underlying strategy of each approach is what most characterizes each particular method. It is commonly accepted that three different algorithm categories exist: affine-scaling, potential-reduction, and path-following. Each of these is described in detail next, along with its subvariants based on the different step and iterate strategies outlined above.

2.2.1 Affine-scaling

In 1984, Karmarkar proposed a method based on projective transformations of gradient-based directional steps. This seminal article [30] led to a flurry of research that extended the development of interior methods to its extensive coverage today. In some of this earlier research, Karmarkar's method was simplified, the transformations were eliminated, and the new class of methods that surfaced was termed affine-scaling. It was later realized that Dikin had proposed, 17 years earlier, the first method of this kind, including a proof of convergence assuming only primal nondegeneracy.

The general strategy of this class of methods consists of solving a series of subproblems, substituting an ellipsoid into each feasible set, which can be more easily handled. Given a strictly feasible (interior) starting point, an ellipsoid is defined that is centered on the current iterate and inscribed within the feasible region. The next iterate is defined as the minimizer of the objective function over this ellipsoid.

In Karmarkar's original proposal, the variables were scaled so as to have no component too close to the feasibility boundaries. This way, the feasibility constraints would not cause the step at each iteration to become too small. Applying this idea to the LP-Primal problem, we can scale the variables according to $x = Dw$, where D is a diagonal matrix. Taking the specific case of $D = X^k = \text{diag}(x^k)$ we have at any iteration $w = e$ (i.e. x^k maps to e), which has the desired effect of staying relatively distant from the feasibility

boundaries. The problem LP-Primal becomes

ASLP1	$\underset{w}{\text{minimize}}$	$(X^k c)^T w$
	subject to	$AX^k w = b, \quad w \geq 0.$

We can then substitute the nonnegativity constraints defining the feasible region by a unit ball centered at e with radius 1, noting that $\{w; \|w - e\| \leq 1\} \subset \{w; w \geq 0\}$. The resulting problem,

ASLP2	$\underset{w}{\text{minimize}}$	$(X^k c)^T w$
	subject to	$AX^k w = b, \quad \|w - e\| \leq 1,$

can be solved analytically in a simple fashion. Performing the inverse scaling, we get the same problem in the original space:

LP'	$\underset{x}{\text{minimize}}$	$c^T x$
	subject to	$Ax = b, \quad \|X^{k^{-1}} x - e\| \leq 1,$

where the unit ball has been scaled into what is known as the Dikin ellipsoid. The optimal solution is given by $x^{k+1} = x^k + \Delta x^k$, where

$$\Delta x^k = -\frac{X^k P_{AX^k} X^k c}{\|P_{AX^k} X^k c\|},$$

and $P_{AX^k} = I - (AX^k)(AX^k(AX^k)^T)^{-1}(AX^k)^T$ represents the projection matrix onto the null space of AX^k.

With step sizes incorporated into the above algorithm, the scheme becomes $x^{k+1} = x^k + \alpha \Delta x^k$ (with $0 < \alpha \leq 1$), for which there exists a convergence proof when $\alpha = 1/8$. In this variant, the iterates are allowed to move outside the ellipsoid, while still retaining feasibility. The step then becomes

$$\Delta x^k = -\frac{X^k P_{AX^k} X^k c}{\|P_{AX^k} X^k c\|_\infty},$$

for which the following convergence results exist.

Theorem 10.

- *If the problem and its dual are nondegenerate, then for every $\alpha < 1$, the sequence generated by the algorithm converges to the optimal solution.*

- *For $\alpha < 2/3$, the sequence generated by the algorithm converges to an optimal solution, regardless of degeneracy.*
- *There exists an example (with $\alpha = 0.995$) for which the sequence generated by the algorithm converges to a nonoptimal solution.*

As far as complexity results go, unfortunately there are no formal results indicating a polynomial bound. In fact, while there are no formal results stating the contrary, some analysis has been done by Vanderbei [56] suggesting that in particular circumstances the steps taken by the affine-scaling method might mimic those of Simplex and, therefore, could theoretically have exponential complexity. Examples of other variants and a more in-depth analysis of affine-scaling methods can also be seen in [56] and Saigal's book on linear programming [51].

2.2.2 Path-following

The main idea behind this type of algorithm is to replace the original problem of minimizing a given objective subject to constraints by a sequence of simpler problems that gradually approximate the original problem. In each subproblem, the objective function is modified by adding to it a weighted term that guarantees feasibility is satisfied. This weight factor is reduced at each subproblem, therefore reducing the effect of the added term and hence obtaining the desired characteristic of approximating the original problem. In essence, given the LP-Primal problem, we have each subproblem defined as

LP(μ)	$\underset{x}{\text{minimize}}$	$B(x, \mu) = c^T x + \mu I(x)$
	subject to	$Ax = b,$

where $I(x)$ ideally has the following properties:

- [PF1] $I(x)$ depends only on the constraints $x \geq 0$;
- [PF2] $I(x)$ preserves the continuity properties of the objective function at all feasible points;
- [PF3] For any sequence of feasible points converging to a point on the boundary of the feasible region, $I(x) \to \infty$.

The last property effectively acts as a barrier, preventing iterates from moving outside the feasible region. For this reason, this type of algorithm is also commonly referred to as a *barrier* method.

Such methods can be seen as an extreme case of a similar approach, termed *penalty methods*, where a high cost ($< \infty$) is assigned to infeasibility, in contrast to property (PF3). A popular penalty method is the quadratic one, based on the *augmented Lagrangian* function, which for problem LP-Primal takes the form

$$L_c(x, y) = c^T x + y^T(Ax - b) + \frac{\rho}{2}\|Ax - b\|^2.$$

In this method, the incorporation of the constraints into the objective means they can be entirely dropped from the subproblems, leading to a succession of bound-constrained problems of the form

$\text{LP}(c^k)$	$\underset{x}{\text{minimize}}$	$L_{c^k}(x, y^k)$
	subject to	$x \geq 0.$

The approximation of $L_{c^k}(x, y^k)$ to the original objective function is strongly determined by the proximity of y^k to y^* and the size of the penalty parameter c^k. Strategies for this method therefore revolve around increasing c^k and/or finding good ways to approximate the dual solution y^k at each iteration. Some recent research has been devoted to particular kinds of methods based on *exact penalty* functions, whose solutions are also solutions to their related unpenalized problems. Because of the later use of barrier-type methods in the remainder of this dissertation, we focus the detailed discussion of the next few sections strictly on these methods, referring the reader interested in a thorough analysis of penalty methods to [12].

The interior function $I(x)$ can be interpreted as an approximation of the indicator function $\chi : \mathbb{R} \to \mathbb{R}$ defined as

$$\chi(u) = \begin{cases} 0 & \text{if } u \leq 0, \\ \infty & \text{otherwise.} \end{cases}$$

Given a barrier function $I(x)$, and assuming $\mu \in \mathbb{R}_+$, the set of minimizers $\{x(\mu); x(\mu) = \text{ argmin } (c^T x + \mu I(x))\}$ defines what is known as the *central trajectory* or *central path*, which leads from the feasible region's *analytic center*, the solution point for $\mu = \infty$, to the optimal solution on the

feasible region's boundary, when $\mu \to 0$. The strategy in this algorithm type is usually for iterates to follow this trajectory (perhaps only roughly); hence, the term *path-following* method is frequently used as well. Convergence properties for this strategy are formally presented below.

Two well known and frequently used functions satisfying the properties outlined above are

$$I(x) = \sum_{i=1}^{m} \frac{1}{x_j},$$

and

$$I(x) = -\sum_{i=1}^{m} \ln(x_j),$$

called the inverse and logarithmic interior functions, respectively. There has been considerable research on both, though the logarithmic function has surfaced as a notable favorite in both theory and practice because of its attractive twice-continuously differentiable and self-concordance properties (the latter to be discussed later). Its first appearance in the literature dates to 1955 with Frisch's use of a closely related logarithmic potential method for retaining feasibility and accelerating convergence [21].

The barrier function derived from the logarithmic interior function then becomes

$$B(x, \mu) = c^T x - \mu \sum_{j=1}^{m} \ln(x_j),$$

which is the de facto standard barrier function for interior methods. It is also noteworthy that it is very closely related to Karmarkar's potential reduction method, as described above; see [22]. Because of the interior property (PF3) and because $B(x, \mu) \simeq c^T x$ for small $\mu > 0$ and all strictly feasible x, it seems reasonable to proceed as in our initial strategy outlined in the beginning of this section. That is, solve a series of equality constrained subproblems while decreasing μ at each outer iteration. From a practical standpoint, however, it is important to note that solving each subproblem accurately would require a fair amount of computational work and so the strategy must be adapted so that proximity of the iterates to the central path suffices.

We outline now a short-step primal-dual feasible algorithm based on the strategy outlined above. Given a current iterate (x^k, y^k, z^k) close to the central path and a current duality gap measure μ^k we define $\mu^{k+1} = \sigma\mu^k (0 <$

$\sigma < 1$) and we have the following primal-dual conditions based on μ^{k+1}:

$$\begin{aligned} Ax^k &= b && (2.2)\\ A^T y^k + z^k &= c && (2.3)\\ x_j z_j &= \mu^{k+1} \ \ \forall\ j. && (2.4) \end{aligned}$$

Solving for the Newton step, we then have the following linear system

$$\begin{pmatrix} A & & \\ & A^T & I \\ Z^k & & X^k \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \equiv \begin{pmatrix} b - Ax^k \\ c - A^t y^k - z^k \\ \mu^{k+1} e - X^k z^k \end{pmatrix}, \qquad (2.5)$$

where Z^k and X^k are diagonal matrices based on the vectors z^k and x^k. Solving this system allows us to define the new iterate simply as $(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + (\Delta x^k, \Delta y^k, \Delta z^k)$ and proceed to the next loop. Note that a full step is taken. This requires (x^k, y^k, z^k) to be close to the central path, and μ^{k+1} to be sufficiently close to μ^k. In other words, the change in μ must be small enough for a pure Newton step to reduce the residual norm $\|(r_1, r_2, r_3)\|$. Hence the name "short-step" method.

One reason for the popularity of path-following interior methods is the extent of theory possible from the nature of the algorithm. These theoretical properties of the central path are presented in detail in the next section as well as a comprehensive convergence analysis including aspects tied to implementation. Yet complexity properties for this type of algorithm are most likely its main attractor. For the scheme outlined above, for example, it has been proven for feasible methods [50] that a solution with ε accuracy (for the duality gap) is reached within at most N iterations, where

$$N = O(\sqrt{n} \log \frac{n\mu^0}{\varepsilon}),$$

where n is the size of primal solution vector and $\mu^0 = x^{0T} z^0$. This polynomial complexity bound is the best attained so far for linear optimization.

As mentioned above, despite their attractive complexity bounds, short-step methods often converge too slowly to be useful in practice. In the above method, σ is actually very close to 1 most of the time, which means a large decrease of the duality gap requires an enormous number of iterations. In the long-step variant of the method above, the idea is to pick a smaller σ. Practical performance is superior, even though this change leads to worse

overall theoretical properties. The difficulty is that in system (2.5), the last equation is an approximation of the nonlinear equation (2.4) that becomes less accurate if μ decreases too quickly. As a consequence, the barrier term might cease to serve its interiorizing purpose and the Newton step might lead to an infeasible iterate. Counter-intuitively, in this long-step variant a *damped* Newton step is therefore taken,

$$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha^k(\Delta x^k, \Delta y^k, \Delta s^k),$$

where $\alpha^k \leq 1$ is chosen to ensure feasibility throughout. Unfortunately, damping the Newton step has the undesirable effect that we now aim at a duality measure of $(1-\alpha^k(1-\sigma))\mu^k$ instead of the original $\sigma\mu^k$. Furthermore, because of our worsened approximation of (2.4) in (2.5) once more, there is no guarantee that the new iterate remains close to the central path. A common tactic to circumvent this problem is to keep taking Newton steps with a fixed duality measure ($\mu^{k+1} = \mu^k$) until the current iterate is sufficiently close to the central path, at which point μ is decreased ($\mu^{k+1} = \sigma\mu^k$). In [50], the complexity result for this long-step algorithm is proven to be

$$N = O(n \log \frac{n\mu^0}{\varepsilon}). \tag{2.6}$$

2.2.3 Potential-reduction

While Affine-Scaling methods were derived from Karmarkar's method, the basic principle in his original algorithm was somewhat different. His famous method made use of a potential function in order to measure improvement at each iteration. In this section that specific method is explained and a class of methods based on similar strategies is introduced.

Karmarkar's method (and consequently other potential-reduction methods) deserves attention not only for its historical significance. Though not as simple as affine-scaling methods nor, as is seen in the next section, as enticing as path-following methods in terms of efficiency results, potential-reduction methods still possess some advantages. In path-following methods, by far the most popular type of interior method nowadays, the iterates ideally follow, as the name suggests, a path leading to the optimal solution. Because of the nonlinear nature of this path, the step sizes (along a computed linear direction) must be adequately controlled in order not to greatly overshoot the path and the region around it, in which good convergence properties can

be expected. Potential-reduction methods, in contrast, have no tie to any such path, therefore allowing the use of large steps, which can greatly accelerate convergence. In addition, these methods have a performance guarantee and, because dual information is readily available, in their implementation it becomes simple to allow the user to set an optimality tolerance.

In the context of linear programming, potential-reduction methods typically involve solving a problem of the form

$$\begin{array}{lll} \text{PRP} & \underset{x,y,z}{\text{minimize}} & P(x,z) = q\log(x^T z) + I(x,z) \\ & \text{subject to} & Ax = b \\ & & x > 0, \\ & & A^T y + z = c \\ & & z \geq 0, \end{array}$$

where q is a parameter of the potential function (Karmarkar's method used $q = m$) and $I(x, z)$ is again some form of interior function. The first part of the (objective) potential function acts to reduce the duality gap to zero, while the second part keeps iterates within the feasible region, as in the path-following methods. The (primal) logarithmic interior function

$$I(x) = -\sum_{i=1}^{m} \ln(x_j),$$

is again a popular choice. Another primal-dual variant, named after Tanabe-Todd-Ye, uses

$$I(x) = -\sum_{i=1}^{m} \ln(x_j z_j),$$

which aims at having well-centered iterates, bringing this method closer to being a path-following method.

Under mild assumptions, it can be easily shown that in the case of the primal logarithmic barrier, the duality gap is bounded by the potential function,

$$x^T z \leq C_1 e^{P(x,z)/q}, \tag{2.7}$$

where C_1 is a problem-specific constant. Assuming some reduction of $P(x, z)$ by at least δ at every iteration, we then have that the method converges in at most

$$N = \frac{q}{\delta}(\log(\frac{x^T z}{\epsilon}) + C_2)$$

iterations, where C_2 is a constant dependent upon the starting point and the problem data.

For a more comprehensive overview of potential-reducing methods, the reader is referred to [53, 27, 1].

2.3 Theory of Barrier-type Methods

Following Karmarkar's introduction of interior methods for linear optimization, the nonlinear aspect of the method suggested that an extension to convex programming was possible. Yet most of the subsequent research was dedicated to the extension of potential-reduction and path-following algorithms to more restricted classes of nonlinear problems. Then, in their monumental contribution of 1994, Nesterov and Nemirovskii presented a deep and unifying theory and analysis of interior point methods for general convex optimization problems [41]. In this section we give an overview of some of their work, presenting the main ideas of self-concordant barrier functions as well as the main associated theoretical results.

2.3.1 Convexity and Self-Concordance

Let us begin by defining the general convex problem as

CP	$\underset{x}{\text{minimize}}$	$c_0(x)$
	subject to	$c_i(x) \geq 0, \quad i = 1, \ldots, m,$

where the functions $c_0, c_1, \ldots, c_m : \mathbb{R}^n \to \mathbb{R}$ are convex. That is, they satisfy

$$c_i(\alpha x + \beta y) \leq \alpha c_i(x) + \beta c_i(y), \quad \forall x, y \in \mathbb{R}^n \text{ and } \alpha, \beta \in \mathbb{R}^+ \text{ with } \alpha + \beta = 1.$$

Clearly the linear problem LP-Primal discussed above is a special case of CP.

Applying the barrier method idea introduced in the previous section, we can think of solving CP by reducing it to the solution of a series of unconstrained problems defined as

CP(μ)	$\underset{x}{\text{minimize}}$	$c_0(x) + \mu I(c(x)).$

Nesterov and Nemirovskii introduced the notion of a self-concordant function and showed that if $I(x)$ belongs to this class of functions, then CP can be solved efficiently using Newton's method. Formally, a function $f : \mathbb{R} \to \mathbb{R}$ is *self-concordant* if it satisfies the following property:

$$|f'''(x)| \leq f''(x)^{3/2}, \quad \forall x \in \mathrm{dom} f. \tag{2.8}$$

More generally, a function $f : \mathbb{R}^n \to \mathbb{R}$ is self-concordant if it is self-concordant along every line of its domain. That is, if $\tilde{f}(t) = f(x+tv)$ is self-concordant of t for all $x \in \mathrm{dom} f$ and for all v. The inverse and logarithmic functions introduced above are examples from this class.

In addition to providing a relative bound for the third derivative, which is necessary to obtain complexity results based on Newton's method, property (2.8) has the added benefit of being invariant with respect to affine coordinate changes. In other words, $\tilde{f}(y) = f(ay + b)$ for $a, b \in \mathbb{R}$ is self-concordant if and only f is. Furthermore, this notion of properties under which self-concordancy is preserved can be extended to what is known as "barrier calculus", noting that (some) scaling and addition do not alter self-concordancy, i.e. if c_1, c_2 are self-concordant, so are ac_1 with $a \geq 1$ and $c_1 + c_2$. This remarkable framework allows for a great deal of flexibility in the development of interior methods under one simple and robust unifying theory. While self-concordant barriers are only known for a few important classes of programs, such as linear, convex quadratic, and semidefinite programs, [41] contains a proof that every open convex set in $\mathbb{R}^n$ posesses a self-concordant barrier, which at least in theory provides a great deal of flexibility for its use in interior methods.

2.3.2 Convergence and Complexity of Self-Concordant Barrier Methods

While classic results of optimization typically involve the gradient of the objective or constraints, here a particular term turns out to be more useful. The *Newton decrement*,

$$\lambda(x) = (g^T H^{-1} g)^T$$

which can also be written as

$$\lambda(x) = (\Delta x^T \nabla^2 f(x) \Delta x)^{1/2},$$

is not only useful in the analysis of Newton's method, but is also a practical stopping criterion, since

$$f(x) - \inf_y \hat{f}(y) = f(x) - \hat{f}(x + \Delta x) = \frac{1}{2}\lambda(x)^2,$$

where $\hat{f}(x)$ is the second-order approximation of f at x. The Newton decrement can also be used in a backtracking line search, given that

$$\nabla f(x)^T \Delta x = -\lambda(x)^2$$

is the directional derivative of f at x in the direction of the Newton step. In this type of line search, starting with $t = 1$, one takes $t = \beta t$ until

$$f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x = f(x) + \alpha t \lambda(x)^2,$$

where $\alpha \in (0, 1/2)$ and $\beta \in (0, 1)$. This guarantees the line search eventually terminates with t such that there is a decrease of the objective for a relatively large stepsize.

A particularly useful property of the Newton decrement is that, like self-concordant functions, it is invariant to affine changes in coordinates.

The main convergence result stems from the following result.

Theorem 11. *Using Newton's method with backtracking line search, and given a starting point x^0 and a strictly self-concordant barrier, if the sublevel set $S = \{x; f(x) \leq f(x^0)\}$ is closed and f is bounded below, then there exist numbers $\eta, \gamma > 0$, with $0 < \eta \leq 1/4$, dependent only on the line search parameters α, β, such that the following hold:*

- *If $\lambda(x^k) > \eta$, then*
$$f(x^{k+1}) - f(x^k) \leq -\gamma. \tag{2.9}$$
- *If $\lambda(x^k) \leq \eta$, then the line search selects $t = 1$ and*
$$2\lambda(x^{k+1}) \leq (2\lambda(x^k))^2. \tag{2.10}$$

The first inequality provides a picture of what occurs during the *damped phase* of the Newton method, where it is possible that $t < 1$. Since the objective decreases monotonically by at least γ every iteration, convergence is guaranteed (recall the assumption about f on the sublevel set S being

bounded below). In addition, applying (2.9) recursively we find that $\lambda(x^k) \leq \eta$ after at most

$$\frac{f(x^0) - p^*}{\gamma}$$

iterations. Once a point x^l in this region is reached, (2.10) indicates this is now a pure Newton phase ($t = 1$) and convergence is quadratic. In particular by applying (2.10) repeatedly we find that $f(x^l) - p^* \leq \epsilon$ after at most

$$\log_2 \log_2(\frac{1}{\epsilon})$$

iterations. We thus obtain a final bound on γ

$$\gamma = \alpha\beta\frac{\eta^2}{1+\eta},$$

and hence a bound on total iterations,

$$\frac{f(x^0) - p^*}{\gamma} + \log_2 \log_2(\frac{1}{\epsilon}) = \frac{20 - 8\alpha}{\alpha\beta(1-2\alpha)^2}(f(x^0) - p^*) + \log_2 \log_2(\frac{1}{\epsilon}), \quad (2.11)$$

which is dependent only on the line search parameters α, β and the final accuracy ϵ.

As mentioned earlier, one of the great advantages of using self-concordant functions is their invariance to affine transformations. This allows for a great number of possible self-concordant functions. A few examples follow:

- (Log barrier for linear inequalities) $I(x) = -\sum_{i=1}^m \log(b_i - a_i^T x)$, with $\operatorname{dom} I = \{a_i^T x < b_i, i = 1, \ldots, m\}$.
- (Log-determinant) $I(x) = -\log\det(X)$, with $\operatorname{dom} I = S_{++}^n$.
- (Log of concave quadratic) $I(x) = -\log(x^T P x + q^T x + r)$, where $P \in -S_+^n$ and $\operatorname{dom} I = \{x; x^T P x + q^T x + r > 0\}$.

It can also be shown that for any convex function $\phi : \mathbb{R} \to \mathbb{R}$ with $\operatorname{dom} \phi = \mathbb{R}_{++}$ satisfying

$$|\phi'''(x)| \leq 3\frac{\phi''(x)}{x} \text{ for all } x, \quad (2.12)$$

the function

$$I(x) = -\log(-\phi(x)) - \log(x)$$

is self-concordant. Examples for ϕ satisfying (2.12) include

$$\phi(x) = x^p \text{ for } -1 \leq p \leq 0; \qquad \phi(x) = (ax + b)^2/x.$$

2.4 Implementation Issues

2.4.1 Path-Following Methods

Many practical issues arise in the details of typical implementations of the algorithms described above that can greatly affect their efficiency. We focus in this section on the particular case of primal-dual path-following methods, and cover the basic aspects that constitute the framework for an implementation. These aspects are: formulation of the primal-dual equations, techniques for solving the equations efficiently, line search methods, stopping criteria, and strategies for handling the barrier parameter μ.

Primal-dual methods using the log barrier are by far the most common algorithm type used in practice. Applying this method to CP leads to the following system that defines a feasible solution $(x(\mu), y(\mu))$ for each positive μ:

$$F^{\mu}(x,y) = \begin{pmatrix} g(x) - J(x)^T y \\ C(x)y - \mu e \end{pmatrix} = 0,$$

where $J(x)$ is the Jacobian of the constraints, $C(x)$ the diagonal matrix of the constraint values, and e the vector of ones of appropriate dimension. The Newton direction $(\Delta x, \Delta y)$ is defined by the Newton equations

$$F^{\mu}(x,y)'(\Delta x, \Delta y) = -F^{\mu}(x,y).$$

After collecting terms on the right-hand side, we have what are known as the Newton primal-dual equations,

$$\begin{pmatrix} H(x,y) & -J(x)^T \\ YJ(x) & C(x) \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} g(x) - J(x)^T y \\ C(x)(y - \pi(x,\mu)) \end{pmatrix}, \tag{2.13}$$

where $H(x,y)$ is the Hessian of the Lagrangian evaluated at (x,y), Y the diagonal matrix of the dual values, $\pi_i = \mu / c_i(x)$ for $i = 1, \ldots, m$, and we assume $C \succ 0, Y \succ 0$.

Since most of the computational effort during an iteration of a primal-dual method is dedicated to solving the above system, we shall look more carefully at the different options for handling this task in an efficient and numerically reliable way. The system (2.13) can be solved as is using, for instance, an iterative method such as pre-conditioned conjugate-gradient. In the case where $J(x)$ is an operator, an iterative approach such as this is the

only option. In other circumstances, direct solvers based on factorizations can be more efficient, such as sparse LU factorizations.

Sometimes performing appropriate modifications to (2.13) allows for the use of specialized factorizations known to be even more efficient. One common approach is to obtain a more condensed version of (2.13) by performing block elimination. Proceeding this way leads to the system

$$H_c(x,y)\Delta x = -(g(x) - J(x)^T\pi(x,\mu)),$$

where

$$H_c(x,y) = H(x,y) + J(x)^T Y^{-1} C(x) J(x).$$

If $f(x)$ is convex, $H(x,y)$ is symmetric positive definite and this system can be solved using a Cholesky factorization, noting that $J(x)^T Y^{-1} C(x) J(x)$ takes the form of AD^2A^T and is hence also symmetric positive definite. The Cholesky factorization then takes the form of LL^T, which is twice as efficient for computation as the LU decomposition.

A disadvantage with the above approach is that during the block elimination, fill-in can occur, causing the system to become dense, which is inappropriate for Cholesky solvers dependent on sparsity. A different strategy is to symmetrize (2.13) and solve it using other factorization techniques. Various alternatives exist for symmetrizing (2.13), most of which include some rearrangement of blocks in the Newton system followed by an appropriate scaling of certain blocks. A particularly popular symmetrization that works for the system above, and preserves the eigenvalues of the system, is to premultiply the second block of equations in (2.13) by $Y^{1/2}$ while also scaling the dual variables by $Y^{-1/2}$. This results in the system

$$\begin{pmatrix} H(x,y) & J(x)^T Y^{1/2} \\ Y^{1/2} J(x) & -C(x) \end{pmatrix} \begin{pmatrix} \Delta x \\ Y^{-1/2}\Delta y \end{pmatrix} = - \begin{pmatrix} g(x) - yJ(x)^T \\ Y^{-1/2} C(x)(y-\pi) \end{pmatrix}. \quad (2.14)$$

In the convex case, the matrix M in (2.14) is quasi-definite, which means suitable Cholesky-type factorizations such as $PMP^T = LDL^T$ exist for arbitrary symmetric permutations P, with D a diagonal but indefinite matrix. Another type of factorization for symmetric indefinite systems is LBL^T, where B is block-diagonal with blocks of order 1 or 2. This last approach has been analyzed with some success by Fourer and Mehrotra [18] and Wright [58] for the LP case.

It is important to note that in all of the above situations (and most other ways of tackling (2.13)), the systems ultimately handled tend to be very large

and contain few nonzero entries, so the use of efficient sparse linear solvers is critical.

As mentioned in section 2.2.2, a popular implementation of the primal-dual method is the long-step variant, where once the Newton directions are calculated, the step is damped to ensure primal and dual feasibility. Typically this means α^k is chosen so that $(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha^k(\Delta x^k, \Delta y^k, \Delta z^k)$ has components positive in (x^k, z^k). Also common is the additional use of line search methods (e.g. backtracking) that ensure decrease of some merit function (see [11]).

The loop of outer iterations should stop when the primal and dual residuals are within a predetermined threshold. For example, in the case of a problem like (LP) one could use the following criteria,

$$\|r_p\| = \|Ax - b\| < \varepsilon \tag{2.15}$$

$$\|r_d\| = \|A^t y - c\| < \varepsilon \tag{2.16}$$

Given that typical machine precision is roughly 10^{-16}, values for ϵ in the range of 10^{-8} are generally adequate.

For large-scale problems, the Newton system is often expensive to solve by direct methods. In such cases, the computational effort required to solve this inner loop accurately is unjustified for what is ultimately only one iterate on the (outer loop) path to optimality. Inexact Newton methods can be used to speed up the overall solve time by making use of reasonable approximations at this sublevel. Iterative methods can be used instead, for example, with a natural stopping rule based on the residual $\|r^k\|/\|\nabla f(x^k)\|$, where, if p_k is the actual Newton step, and r^k is given by

$$r^k = \nabla^2 f(x^k) p_k + \nabla f(x^k).$$

At the other end of the spectrum, solving these subproblems too coarsely might lead to excessive inaccuracy and overall poor performance, so a key question is what trade-offs are reasonable at this level in terms of improving overall performance. This question is addressed in [10], where a general class of inexact Newton methods is handled using

$$\|r^k\|/\|\nabla f(x^k)\| \leq \eta_k.$$

Such inexact methods are proven to be locally convergent if the forcing sequence $\{\eta_k\}$ is uniformly less than 1; in other words, if

$$\eta_k \leq \eta_{\max} < t < 1. \tag{2.17}$$

The rate of convergence is superlinear if and only if $\|r_k\| = o(\|\nabla f(x^k)\|)$ as $k \to \infty$.

Another implementation issue deserving of special attention is that of the choice of barrier parameter μ. For small μ, outer iteration steps are small, but follow the central path closely, which in turn means fewer Newton steps (typically 1) are required to obtain the next outer iterate. This results in a small number of inner iterations while many outer ones are required. A choice of large μ has the opposite effect: large outer steps are taken, but as a consequence, little proximity of these outer iterates to the central path is likely, thus leading to longer time spent in the Newton steps. Reasonably good performance can be obtained by simply keeping $10 \leq \mu \leq 20$, but a dynamic parameter can yield better results without requiring much extra effort to implement.

One popular adaptive method for μ is to make it proportional to the current complementarity value, that is,

$$\mu^k = \sigma^k \frac{(x^k)^T z}{n}, \tag{2.18}$$

where σ^k is a *centering parameter* ($\sigma^k \in (0, 1)$)and n is the number of variables. Mehrotra's predictor-corrector [38] is a popular version of this adaptive approach, where σ^k is calculated using a preliminary (predictor) affine-scaling step (with $\mu = 0$). The primal-dual step is taken with this barrier parameter 2.18 and then a corrector step is added to better follow the central path. While successful in practice, Mehrotra's method is based on a heuristic and thus lacks global convergence properties. In [42], Nocedal, Wächter, and Waltz propose another probing (predictor-like step) method whereby σ^k is determined via minimization of so-called *quality functions* such as the residual error based on the Newton step as a function of σ^k.

2.4.2 Warmstarting Interior Methods

Often times in practice, it is necessary to solve a series of closely related problems. This situation occurs frequently during, for example, analysis of what-if scenarios or when tackling a series of problems in which part of the structure depends on the results of the previous solution. When very little of the problem actually changes from one instance to the next, consecutive solutions are also often not far from each other. When this happens, one would intuitively imagine that a solution to one problem could be used as an

advanced starting point for solving the other problem. *Warmstarting*, as this is called, is common practice with active-set methods for Linear Programming such as Simplex and is particularly useful when problems are large and have long solve time. *Coldstarting*, in contrast, is when no such advanced point is used at startup.

In the Simplex method the iterates step from one feasible solution (vertex of the feasible polyhedron) to the next, moving along the edge that most improves the objective, until no further improvement is possible, i.e. an optimal solution is reached. In this scenario, the advantages of warmstarting are immediate: reaching optimality might take only a few iterations. With interior methods, however, this is most often *not* the case. The main issue is that although the warmstart point is close to the solution, it is typically distant from the central path, on which the algorithm so strongly relies. This leads then to numerical difficulties and backtracking to the extent that warmstarting often requires *more* computational effort than coldstarting. An illustrative example presented by Benson and Shanno [2], detailed in section 6.1, makes the case clear.

In this same paper, Benson and Shanno provide an alternative to overcome the issues described above for LP problems. Starting with a perturbed problem

LP-Primal'	$\underset{x\in\mathbb{R}^n}{\text{minimize}}$	$c^T x$
	subject to	$Ax \leq b, \quad x \geq 0,$

their approach consists of solving, at the warmstart phase, a problem similar to the original perturbed one, yet with a primal-dual penalty,

LP-P	$\underset{x,w}{\text{minimize}}$	$c^T x - d_x^T \psi_x - d_w^T \psi_w$
	subject to	$Ax + w \leq b$
		$-\psi_x \leq x \leq u_z$
		$-\psi_w \leq w \leq u_y$
		$\psi_x, \psi_w \geq 0,$

where ψ_x and ψ_w are relaxation variables for the lower bounds on x and w, respectively, and d_x and d_w are their corresponding penalty parameters. Since the ℓ_1 penalty function is exact, for sufficiently large values of the penalty parameters, the solution to LP-P matches that of LP-Primal'. The

reduced KKT system of this penalized problem has its diagonal matrix and right-hand sides perturbed, in addition to allowing the slack variables of the original primal and dual problems to become negative. With appropriately chosen penalty parameters, these changes avert the typical warmstarting numerical issues highlighted above.

While this technique does improve performance on warmstarting, it relies on dynamic updates of the penalty parameters d_x, d_w, u_y, u_z that, in turn, require some modifications to the underlying algorithm.

Another approach by Roos [49] has good complexity properties for an infeasible interior method, making use of an arbitrary starting point.

Gondzio and Grothey have also made some progress with warmstarting for quadratic programming [26]. Their heuristic is based on performing sensitivity analysis on the Newton step from the warmstart point. Their rationale is based on the fact that blocking is a common phenomenon in warmstarting; that is, only a small fraction of the Newton step can be taken because of the proximity of the current iterate to the feasibility boundary. They note that blocking typically happens only to a small number of components of the Newton direction. Sensitivity analysis thus allows them to choose appropriate modifications for the components to permit larger steps with better centering properties.

A good overview of strategies for warmstarting LPs is also presented by John and Yildirim [29]. They also present several implementation issues and compare the effectiveness of the different approaches based on a variety of perturbations to data components.

Chapter 3

PDCO

The first implementation of a primal-dual long step algorithm is credited to McShane, Monma, and Shanno [36], based on developments by Kojima, Mizuno, and Yoshise [32, 33] and, independently, Megiddo [37]. This method, which uses the logarithmic barrier, was later extended by Lustig, Marsten, and Shanno [35] into the structure most existing implementations are closely related to.

In the remainder of this chapter, we constrain ourselves to the particular implementation used in later analysis, PDCO, developed by Saunders [52], noting only that there is no shortage of comprehensive surveys on the subject of IPM implementation. These cover everything from basic differences of the various existing algorithms to the nuances of the fine details of each implementation, as well as ancillary computational aspects applicable to all methods, such as the choice of an initial point, line searches and trust region methods, presolving, and higher order methods, e.g. [38, 34, 35, 25].

Likewise, a plethora of software codes implementing interior methods exist for linear, convex, and nonconvex programming under various licensing agreements: some are open-source, others free to research community, and yet others strictly commercial. Examples include CPLEX [7], XPRESS [9], LOQO [57, 55], PCx [8], HOPDM [24], BPMPD [39], LIPSOL [60], OSL [28].

3.1 Regularization

Nominally, PDCO [52] solves the optimization problem

NP	$\underset{x}{\text{minimize}}$	$\phi(x)$
	subject to	$Ax = b, \quad \ell \le x \le u,$

where $\phi(x)$ is a separable smooth convex function.

To allow for constrained least-squares problems, and to ensure unique primal and dual solutions (and improve solver stability), we regularize (NP) as

NP2	$\underset{x,r}{\text{minimize}}$	$\phi(x) + \frac{1}{2}\|D_1 x\|^2 + \frac{1}{2}\|r\|^2$
	subject to	$Ax + D_2 r = b, \quad \ell \le x \le u,$

where D_1, D_2 are positive-definite diagonal matrices specified by the user. The diagonals of D_1 are typically small (10^{-3} or 10^{-4}). Similarly for D_2 if the constraints in NP should be satisfied reasonably accurately. For least-squares applications, some or all of the diagonals of D_2 are 1. Note that some elements of ℓ and u may be $-\infty$ and $+\infty$ respectively, but we expect no large numbers in A, b, D_1, D_2. If $\|D_2\|$ is small, we would expect A to be under-determined ($m < n$). If $D_2 = I$, A may have any shape.

3.2 The Barrier Approach

First we introduce slack variables x_1, x_2 to convert the bounds to non-negativity constraints:

NP3	$\underset{x,r,x_1,x_2}{\text{minimize}}$	$\phi(x) + \frac{1}{2}\|D_1 x\|^2 + \frac{1}{2}\|r\|^2$
	subject to	$Ax + D_2 r = b$
		$x - x_1 = \ell$
		$x + x_2 = u$
		$x_1, x_2 \ge 0.$

Then we replace the non-negativity constraints by the log barrier function, obtaining a sequence of convex subproblems with decreasing values of μ ($\mu >$

0):

$$
\begin{array}{lll}
\text{NP}(\mu) & \underset{x,r,x_1,x_2}{\text{minimize}} & \phi(x) + \tfrac{1}{2}\|D_1 x\|^2 + \tfrac{1}{2}\|r\|^2 - \mu \sum_j \ln([x_1]_j [x_2]_j) \\
& \text{subject to} \quad Ax + D_2 r = b & : y \\
& \qquad\qquad\quad x - x_1 = \ell & : z_1 \\
& \qquad\qquad\quad -x - x_2 = -u, & : z_2
\end{array}
$$

where y, z_1, z_2 denote dual variables for the associated constraints. With $\mu > 0$, most variables are strictly positive: $x_1, x_2, z_1, z_2 > 0$.

The KKT conditions for the barrier subproblem involve the three *primal* equations of NP(μ), along with four *dual* equations stating that the gradient of the subproblem objective should be a linear combination of the gradients of the primal constraints:

$$
\begin{array}{rll}
Ax + D_2 r = & b & \\
x - x_1 = & \ell & \\
-x - x_2 = & -u & \\
A^T y + z_1 - z_2 = & g(x) + D_1^2 x & : x \\
D_2 y = & r & : r \\
X_1 z_1 = & \mu e & : x_1 \\
X_2 z_2 = & \mu e, & : x_2
\end{array}
\tag{3.1}
$$

where $X_1 = \text{diag}(x_1)$, $X_2 = \text{diag}(x_2)$, and similarly for Z_1, Z_2 later. Actually the last two equations arise in a different form. The dual equation for x_1 is really

$$
-z_1 = \nabla(-\mu \ln(x_1)) = -\mu X_1^{-1} e, \tag{3.2}
$$

where e is a vectors of 1's. Thus, $x_1 > 0$ implies $z_1 > 0$, and multiplying by $-X_1$ gives the equivalent equation $X_1 z_1 = \mu e$ as stated. In this form, the last two equations are commonly called (perturbed) *complementarity* conditions.

3.3 Newton's Method

We now eliminate $r = D_2 y$ and apply Newton's method:

$$\begin{aligned}
A(x+\Delta x) + D_2^2(y+\Delta y) &= b \\
(x+\Delta x) - (x_1+\Delta x_1) &= \ell \\
-(x+\Delta x) - (x_2+\Delta x_2) &= -u \\
A^T(y+\Delta y) + (z_1+\Delta z_1) - (z_2+\Delta z_2) &= g + H\Delta x + D_1^2(x+\Delta x) \\
X_1 z_1 + X_1\Delta z_1 + Z_1\Delta x_1 &= \mu e \\
X_2 z_2 + X_2\Delta z_2 + Z_2\Delta x_2 &= \mu e,
\end{aligned}$$

where g and H are the current objective gradient and Hessian. To solve this Newton system, we work with three sets of residuals:

$$\begin{pmatrix} \Delta x - \Delta x_1 \\ -\Delta x - \Delta x_2 \end{pmatrix} = \begin{pmatrix} r_\ell \\ r_u \end{pmatrix} \equiv \begin{pmatrix} \ell - x + x_1 \\ -u + x + x_2 \end{pmatrix}, \tag{3.3}$$

$$\begin{pmatrix} X_1\Delta z_1 + Z_1\Delta x_1 \\ X_2\Delta z_2 + Z_2\Delta x_2 \end{pmatrix} = \begin{pmatrix} c_\ell \\ c_u \end{pmatrix} \equiv \begin{pmatrix} \mu e - X_1 z_1 \\ \mu e - X_2 z_2 \end{pmatrix}, \tag{3.4}$$

$$\begin{pmatrix} A\Delta x + D_2^2\Delta y \\ -H_1\Delta x + A^T\Delta y + \Delta z_1 - \Delta z_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \equiv \begin{pmatrix} b - Ax - D_2^2 y \\ g + D_1^2 x - A^T y - z_1 + z_2 \end{pmatrix} \tag{3.5}$$

where $H_1 = H + D_1^2$. We use (3.3) and (3.4) to replace two sets of vectors in (3.5). With

$$\begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} = \begin{pmatrix} -r_\ell + \Delta x \\ -r_u - \Delta x \end{pmatrix}, \qquad \begin{pmatrix} \Delta z_1 \\ \Delta z_2 \end{pmatrix} = \begin{pmatrix} X_1^{-1}(c_\ell - Z_1\Delta x_1) \\ X_2^{-1}(c_u - Z_2\Delta x_2) \end{pmatrix}, \tag{3.6}$$

$$\begin{aligned}
H_2 &\equiv H + D_1^2 + X_1^{-1}Z_1 + X_2^{-1}Z_2 \\
w &\equiv r_2 - X_1^{-1}(c_\ell + Z_1 r_\ell) + X_2^{-1}(c_u + Z_2 r_u),
\end{aligned} \tag{3.7}$$

we find that

$$\begin{pmatrix} -H_2 & A^T \\ A & D_2^2 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} w \\ r_1 \end{pmatrix}. \tag{3.8}$$

3.4 Solving for $(\Delta x, \Delta y)$

If $\phi(x)$ is a general convex function with known Hessian H, system (3.8) may need to be treated directly by sparse or iterative methods.

Alternatively it may be possible to obtain a sparse Cholesky factorization

$$H_2 = LL^T \quad \text{or} \quad H_2 \approx LL^T, \tag{3.9}$$

where $H_2 = H +$ diagonal terms, and L is a nonsingular permuted triangle. This is trivial if $\phi(x)$ is a separable function, since H and H_2 in (3.7) are then diagonal. In other cases it may suffice to use $\text{diag}(H)$ or some other approximation to H in the definition of H_2 and L, thereby implementing a pseudo-Newton method for obtaining reasonable directions $(\Delta x, \Delta y)$.

System (3.8) may now be solved by eliminating either Δx or Δy:

$$(A^T D_2^{-2} A + H_2)\Delta x = A^T D_2^{-2} r_1 - w, \qquad D_2^2 \Delta y = r_1 - A\Delta x, \tag{3.10}$$

or

$$(AH_2^{-1}A^T + D_2^2)\Delta y = AH_2^{-1}w + r_1, \qquad H_2 \Delta x = A^T \Delta y - w. \tag{3.11}$$

Sparse Cholesky factorization may again be applicable to the left-hand parts of these systems, but for numerical reasons it is preferable to regard them as least squares problems:

$$\min_{\Delta x} \left\| \begin{pmatrix} D_2^{-1}A \\ L^T \end{pmatrix} \Delta x - \begin{pmatrix} D_2^{-1} r_1 \\ -L^{-T} w \end{pmatrix} \right\|^2, \qquad D_2 \Delta y = D_2^{-1}(r_1 - A\Delta x), \tag{3.12}$$

or

$$\min_{\Delta y} \left\| \begin{pmatrix} L^{-1}A^T \\ D_2 \end{pmatrix} \Delta y - \begin{pmatrix} L^{-1} w \\ D_2^{-1} r_1 \end{pmatrix} \right\|^2, \qquad L^T \Delta x = L^{-1}(A^T \Delta y - w). \tag{3.13}$$

The right-most vectors in (3.12)–(3.13) are part of the residual vectors for the least squares problems (and may be by-products from the least squares computation).

3.5 Scaling

PDCO assumes the problem has been sensibly scaled such that $\|A\| \approx 1$. One such scaling is described in section 4.4. PDCO then allows the user to define factors β and ζ to scale other problem data in order to improve

numerical characteristics of the problem without altering its solution. The guiding principle for the choice of these inputs is the following:

$$\begin{aligned} \beta &= \text{input estimate of} \|x\|_\infty \\ \zeta &= \text{input estimate of} \|z\|_\infty. \end{aligned}$$

Within PDCO, they are used to scale the problem data such that $\|\bar{x}\|_\infty \approx 1$ and $\|\bar{z}\|_\infty \approx 1$, where $\bar{x}$ and $\bar{z}$ represent the scaled versions of x and z, respectively.

Given the LP problem

LP3	$\underset{x,r,x_1,x_2}{\text{minimize}}$	$c^T x + \frac{1}{2}\|D_1 x\|^2 + \frac{1}{2}\|r\|^2$	
	subject to	$Ax + D_2 r = b$	$: y$
		$x - x_1 = \ell$	$: z_1$
		$-x - x_2 = -u$	$: z_2$
		$x_1,\ x_2 \geq 0,$	

a natural choice for the β scaling factor is $\beta = \|b\|_\infty$ which, given the primal feasibility equation, leads to $\bar{x} = x/\beta$. To keep the problem consistent, the objective equation in turn means that $\tilde{c} = \beta c$ and then a typical choice for the second scaling factor would be $\zeta = \|\tilde{c}\|_\infty = \|\beta c\|_\infty$. Adjusting the other terms throughout the problem for consistency gives us the following final scaling set:

$$\begin{array}{lll} \bar{A} = A, & \bar{b} = b/\beta, & \bar{c} = \beta c/\zeta, \\ \bar{l} = l/\beta, & \bar{u} = u/\beta, & \bar{x} = x/\beta, \\ \bar{y} = \beta y/\zeta, & \bar{z}_1 = \beta z_1/\zeta, & \bar{z}_2 = \beta z_2/\zeta, \\ \bar{D}_1 = \beta D_1/\sqrt{\zeta}, & \bar{D}_2 = \sqrt{\zeta} D_2/\beta, & \bar{r} = r/\sqrt{\zeta}. \end{array}$$

Chapter 4

The Refine and Zoom Technique for Linearly Constrained Optimization

We now introduce the technique that is the main focus of this dissertation, based on refinements and scaling of problems using Interior Methods. This technique is aimed at accelerating IPMs using iterative solvers for computing each search direction, and improving performance of general IPMs for warmstarting linearly constrained problems. The results from here on are, to the best of the author's knowledge, new.

4.1 Motivation

When the constraint matrix of an LP is dense or when it is expressed through an operator, it is necessary for an IPM to make use of iterative solvers. In such situations, as the Newton iterations proceed, the systems defining the search directions become increasingly ill-conditioned, and the number of calls to the underlying solver increases drastically. Since in such methods most of the work takes place inside the iterative solver, the overall computational effort is dramatically increased. The mechanics of this issue are detailed in the next section, and a variety of concrete examples exhibiting such behavior are given. This is related to the fact that the iterates in the initial stages of the solve are more distant from the constraints of the problem, a numerically more stable configuration for IPMs, coupled with the known result that

greater accuracy is required in solving the Newton equations, the closer one wishes to get to the true solution, as (2.17) states.

The situation described above serves as motivation for the development of a new technique aimed at reducing the effort in the final iterations of the solve. The idea is to solve the problem in two stages, placing a "magnifying glass" over the second stage, in an effort to emulate the interior quality of the first (cheap) step. If a final solution with accuracy 10^{-6} is sought, for instance, an approximate solution $(\tilde{x}, \tilde{y}, \tilde{z})$ is first obtained, with accuracy 10^{-3} in the normal (cold start) fashion. Given a well-conditioned problem, theory in [10] says this should always be a relatively effortless solve. Then a new problem is defined in terms of a correction (dx, dy, dz) such that $(\tilde{x} + dx, \tilde{y}+dy, \tilde{z}+dz)$ solves the original problem within the desired final accuracy of 10^{-6}. Adequate scaling must be applied to this second-stage problem because in particular the variables (dx, dy, dz) are scaled up by a factor of 10^3 to be of $O(1)$. The idea is for the conditions of this second problem to resemble a problem in its first stage. A cold start is again used, with the initial iterates well within the interior of the feasible region, where the solver is usually most efficient. With the scaling, the definition problem only has to be solved to an accuracy of 10^{-3} again: in theory another step requiring little computational effort. When the magnified (dx, dy, dz) is scaled down by a factor of 10^3 and added to the first-stage solution, its 3-digit accuracy should give 6-digit accuracy to the corrected solution.

In the next section the details of the method are given, along with the modifications necessary to the underlying interior-point method.

4.2 Refinement

Suppose $(\widetilde{x}, \widetilde{y}, \widetilde{z}_1, \widetilde{z}_2, \widetilde{x}_1, \widetilde{x}_2, \widetilde{r})$ is an approximate solution to problem P3:

$$
\begin{array}{lll}
\text{P3} & \underset{x,r,x_1,x_2}{\text{minimize}} & \phi(x) + \tfrac{1}{2}\|D_1 x\|^2 + d^T r + \tfrac{1}{2}\|r\|^2 + c_1^T x_1 + c_2^T x_2 + \kappa \\
& \text{subject to} \quad Ax + D_2 r = b & : y \\
& \qquad\qquad\quad x - x_1 = \ell & : z_1 \\
& \qquad\qquad -x - x_2 = -u & : z_2 \\
& \qquad\qquad\quad x_1,\ x_2 \geq 0, &
\end{array}
$$

where $d = c_1 = c_2 = \kappa = 0$. Then we redefine the problem with

$$\begin{aligned} x &= \widetilde{x} + dx \\ r &= \widetilde{r} + dr, \end{aligned}$$

which yields

$$\begin{array}{lll} \text{P3}' & \underset{dx,dr,x_1,x_2}{\text{minimize}} & \phi(\widetilde{x}+dx) + \frac{1}{2}\|D_1(\widetilde{x}+dx)\|^2 + d^T(\widetilde{r}+dr) \\ & & \quad + \frac{1}{2}\|\widetilde{r}+dr\|^2 + c_1^T x_1 + c_2^T x_2 + \kappa \\ & \text{subject to} & A(\widetilde{x}+dx) + D_2(\widetilde{r}+dr) = b \qquad : y \\ & & \widetilde{x} + dx - x_1 = \ell \qquad : z_1 \\ & & -\widetilde{x} - dx - x_2 = -u \qquad : z_2 \\ & & x_1,\ x_2 \geq 0, \end{array}$$

which simplifies to

$$\begin{array}{lll} \text{P3}' & \underset{dx,dr,x_1,x_2}{\text{minimize}} & \phi(\widetilde{x}+dx) + (D_1^2\widetilde{x})^T dx + \frac{1}{2}\|D_1 dx\|^2 \\ & & \quad + (d+\widetilde{r})^T dr + \frac{1}{2}\|dr\|^2 + \kappa \\ & & \quad + \frac{1}{2}\|D_1\widetilde{x}\|^2 + d^T\widetilde{r} + \frac{1}{2}\|\widetilde{r}\|^2 + c_1^T x_1 + c_2^T x_2 \\ & \text{subject to} & Adx + D_2 dr = \widetilde{b} \qquad : y \\ & & dx - x_1 = \tilde{\ell} \qquad : z_1 \\ & & -dx - x_2 = -\tilde{u} \qquad : z_2 \\ & & x_1,\ x_2 \geq 0, \end{array}$$

where

$$\begin{aligned} \widetilde{b} &= b - A\widetilde{x} - D_2\widetilde{r} \\ \tilde{\ell} &= \ell - \widetilde{x} \\ \tilde{u} &= u - \widetilde{x}. \end{aligned}$$

We now add the following Lagrangian terms to the objective:

$$\widetilde{y}^T(\widetilde{b} - Adx - D_2 dr), \qquad \widetilde{z}_1^T(\tilde{\ell} - dx + x_1), \qquad \widetilde{z}_2^T(-\tilde{u} + dx + x_2),$$

giving a problem with modified dual variables:

$$
\begin{array}{lll}
\text{P3}'' & \underset{dx,dr,x_1,x_2}{\text{minimize}} & \phi(\widetilde{x}+dx) + (D_1^2\widetilde{x} - A^T\widetilde{y} - \widetilde{z}_1 + \widetilde{z}_2)^T dx + \frac{1}{2}\|D_1 dx\|^2 \\
& & + (d + \widetilde{r} - D_2\widetilde{y})^T dr + \frac{1}{2}\|dr\|^2 + (c_1 + \widetilde{z}_1)^T x_1 \\
& & + (c_2 + \widetilde{z}_2)^T x_2 + \widetilde{b}^T\widetilde{y} + \frac{1}{2}\|D_1\widetilde{x}\|^2 + d^T\widetilde{r} \\
& & + \frac{1}{2}\|\widetilde{r}\|^2 + \widetilde{z}_1^T\tilde{\ell} - \widetilde{z}_2^T\tilde{u} + \kappa \\
& \text{subject to} & Adx + D_2 dr = \widetilde{b} \qquad : dy \\
& & dx - x_1 = \tilde{\ell} \qquad : dz_1 \\
& & -dx - x_2 = -\tilde{u} \qquad : dz_2 \\
& & x_1,\ x_2 \geq 0,
\end{array}
$$

It is now clear that if we define

$$
\begin{aligned}
\tilde{\phi}(dx) &= \phi(\widetilde{x}+dx) + (D_1^2\widetilde{x} - A^T\widetilde{y} - \widetilde{z}_1 + \widetilde{z}_2)^T dx \\
\widetilde{d} &= d + \widetilde{r} - D_2\widetilde{y} \\
\widetilde{c}_1 &= c_1 + \widetilde{z}_1 \\
\widetilde{c}_2 &= c_2 + \widetilde{z}_2 \\
\tilde{\kappa} &= \kappa + \widetilde{b}^T\widetilde{y} + \frac{1}{2}\|D_1\widetilde{x}\|^2 + d^T\widetilde{r} + \frac{1}{2}\|\widetilde{r}\|^2 + \widetilde{z}_1^T\tilde{\ell} - \widetilde{z}_2^T\tilde{u} \\
y &= \widetilde{y} + dy \\
z_1 &= \widetilde{z}_1 + dz_1 \\
z_2 &= \widetilde{z}_2 + dz_2,
\end{aligned}
$$

we return to the structure of problem P3. We therefore concern ourselves with only problem P3 in our analysis from now on.

Replacing again the non-negativity constraints in P3 by the log barrier function, we obtain the analogous sequence of subproblems:

$$
\begin{array}{lll}
\text{P3}(\mu) & \underset{x,r,x_1,x_2}{\text{minimize}} & \phi(x) + \frac{1}{2}\|D_1 x\|^2 + d^T r + \frac{1}{2}\|r\|^2 \\
& & + c_1^T x_1 + c_2^T x_2 + \kappa - \mu\sum_j \ln([x_1]_j[x_2]_j) \\
& \text{subject to} & Ax + D_2 r = b \qquad : y \\
& & x - x_1 = \ell \qquad : z_1 \\
& & -x - x_2 = -u \qquad : z_2.
\end{array}
$$

4.3 Revisions to PDCO

If we define

$$\begin{aligned} \bar{z}_1 &= z_1 + c_1 \\ \bar{z}_2 &= z_2 + c_2, \end{aligned}$$

then the KKT conditions for P3(μ) become

$$\begin{aligned} Ax + D_2 r &= b \\ x - x_1 &= \ell \\ -x - x_2 &= -u \\ A^T y + z_1 - z_2 &= g(x) + D_1^2 x \qquad : x \\ D_2 y &= d + r \qquad : r \\ X_1 \bar{z}_1 &= \mu e \qquad : x_1 \\ X_2 \bar{z}_2 &= \mu e, \qquad : x_2 \end{aligned} \tag{4.1}$$

where $X_1 = \text{diag}(x_1)$, $X_2 = \text{diag}(x_2)$, and similarly for $\overline{Z}_1$, $\overline{Z}_2$ later. As before, the last two equations arise in a different form. The dual equation for x_1 is really

$$-z_1 = c_1 + \nabla(-\mu \ln(\bar{x}_1)) = c_1 - \mu X_1^{-1} e, \tag{4.2}$$

where e is a vectors of 1's. Thus, $x_1 > 0$ implies $\bar{z}_1 > 0$, and multiplying by $-X_1$ gives the equivalent equation $X_1 \bar{z}_1 = \mu e$ as stated.

4.3.1 Changes to Newton's Method

We now eliminate $r = D_2 y - d$ and apply Newton's method:

$$\begin{aligned} A(x + \Delta x) + D_2^2(y + \Delta y) - D_2 d &= b \\ (x + \Delta x) - (x_1 + \Delta x_1) &= \ell \\ -(x + \Delta x) - (x_2 + \Delta x_2) &= -u \\ A^T(y + \Delta y) + (z_1 + \Delta z_1) - (z_2 + \Delta z_2) &= g + H\Delta x + D_1^2(x + \Delta x) \\ X_1 \bar{z}_1 + X_1 \Delta z_1 + \overline{Z}_1 \Delta x_1 &= \mu e \\ X_2 \bar{z}_2 + X_2 \Delta z_2 + \overline{Z}_2 \Delta x_2 &= \mu e, \end{aligned}$$

where g and H are the current objective gradient and Hessian. To solve this Newton system, we work with three sets of residuals:

$$\begin{pmatrix} \Delta x - \Delta x_1 \\ -\Delta x - \Delta x_2 \end{pmatrix} = \begin{pmatrix} r_\ell \\ r_u \end{pmatrix} \equiv \begin{pmatrix} \ell - x + x_1 \\ -u + x + x_2 \end{pmatrix}, \tag{4.3}$$

$$\begin{pmatrix} X_1\Delta z_1 + \overline{Z}_1\Delta x_1 \\ X_2\Delta z_2 + \overline{Z}_2\Delta x_2 \end{pmatrix} = \begin{pmatrix} c_\ell \\ c_u \end{pmatrix} \equiv \begin{pmatrix} \mu e - X_1\bar{z} \\ \mu e - X_2\bar{z}_2 \end{pmatrix}, \tag{4.4}$$

$$\begin{pmatrix} A\Delta x + D_2^2\Delta y \\ -H_1\Delta x + A^T\Delta y + \Delta z_1 - \Delta z_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \equiv \begin{pmatrix} b - Ax - D_2^2 y + D_2 d \\ g + D_1^2 x - A^T y - z_1 + z_2 \end{pmatrix} \tag{4.5}$$

where $H_1 = H + D_1^2$. We use (4.3) and (4.4) to replace two sets of vectors in (4.5). With

$$\begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} = \begin{pmatrix} -r_\ell + \Delta x \\ -r_u - \Delta x \end{pmatrix}, \qquad \begin{pmatrix} \Delta z_1 \\ \Delta z_2 \end{pmatrix} = \begin{pmatrix} X_1^{-1}(c_\ell - \overline{Z}_1\Delta x_1) \\ X_2^{-1}(c_u - \overline{Z}_2\Delta x_2) \end{pmatrix}, \tag{4.6}$$

$$\begin{aligned} H_2 &\equiv H + D_1^2 + X_1^{-1}\overline{Z}_1 + X_2^{-1}\overline{Z}_2 \\ w &\equiv r_2 - X_1^{-1}(c_\ell + \overline{Z}_1 r_\ell) + X_2^{-1}(c_u + \overline{Z}_2) r_u) \end{aligned} \tag{4.7}$$

we find that

$$\begin{pmatrix} -H_2 & A^T \\ A & D_2^2 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} w \\ r_1 \end{pmatrix} \tag{4.8}$$

as before.

4.3.2 Changes to Scaling

The additional data terms c_1, c_2, d, and κ also need to be scaled appropriately within PDCO. For the sake of completeness, below is the full revised scaling set:

$$\begin{array}{lll} \bar{A} = A, & \bar{b} = b/\beta, & \bar{c} = \beta c/\zeta, \\ \bar{c}_1 = \beta c_1/\zeta, & \bar{c}_2 = \beta c_2/\zeta, & \bar{l} = l/\beta, \\ \bar{u} = u/\beta, & \bar{x} = x/\beta, & \bar{y} = \beta y/\zeta, \\ \bar{z}_1 = \beta z_1/\zeta, & \bar{z}_2 = \beta z_2/\zeta, & \bar{D_1} = \beta D_1/\sqrt{\zeta}, \\ \bar{D_2} = \sqrt{\zeta} D_2/\beta, & \bar{r} = r/\sqrt{\zeta}, & \bar{d} = d/\sqrt{\zeta}, \\ \bar{\kappa} = \kappa/\zeta. & & \end{array} \tag{4.9}$$

4.4 Scaling (outside PDCO)

As mentioned in section 3.5, PDCO assumes that A is well scaled, which helps improve numerical characteristics (condition number) and speeds up solve time of problems. An example of one way to perform such a scaling is given below for the case of an LP problem, though its extension to nonlinear problems is straightforward. Note that this type of scaling, while used here in the context of refinement problems (second-stage), is also applicable to the first-stage and in fact for *any* LP problem, regardless of the solver being used.

Given the problem

$$\begin{array}{lll} \text{LP3'} & \underset{x,r,x_1,x_2}{\text{minimize}} & c^Tx + \frac{1}{2}\|D_1x\|^2 + d^Tr + \frac{1}{2}\|r\|^2 + c_1^Tx_1 + c_2^Tx_2 + \kappa \\ & \text{subject to} & Ax + D_2r = b \qquad : y \\ & & x - x_1 = \ell \qquad : z_1 \\ & & -x - x_2 = -u \qquad : z_2 \\ & & x_1,\ x_2 \geq 0, \end{array}$$

the idea is to find diagonal matrices R, C and scalars β, ζ such that $\hat{A} = R^{-1}AC^{-1}$, $\hat{c} = c/\zeta$, and $\hat{b} = b/\beta$ all have entries close to 1. To ensure consistency of the data however (that is, to make sure the problem is unchanged), other components of the LP must be scaled accordingly.

Let us begin with the scaling of the constraint matrix. Assuming $A = R\hat{A}C$, the primal feasibility equation becomes

$$R\hat{A}Cx + D_2r = b,$$

from which $\hat{x} = Cx$ (and similarly $\hat{l} = Cl$,$\hat{u} = Cu$). Premultiplying the above equation by R^{-1} we get

$$\hat{A}\hat{x} + R^{-1}D_2r = R^{-1}b,$$

from which $\hat{D}_2 = R^{-1}D_2$, $\hat{b} = R^{-1}b$. Performing a similar analysis on the dual problem and the objective equation (in order to keep objective values the same) yields the following scaling set:

$$\begin{array}{lll} \hat{A} = R^{-1}AC^{-1}, & \hat{b} = R^{-1}b, & \hat{c} = C^{-1}c, \\ \hat{c}_1 = C^{-1}c_1, & \hat{c}_2 = C^{-1}c_2, & \hat{l} = Cl, \\ \hat{u} = Cu, & \hat{x} = Cx, & \hat{y} = Ry, \\ \hat{z}_1 = C^{-1}z_1, & \hat{z}_2 = C^{-1}z_2, & \hat{D}_1 = C^{-1}D_1, \\ \hat{D}_2 = R^{-1}D_2, & & \end{array}$$

while d, r, κ remain unchanged.

For the first stage of the zoom and refine technique, the typical choices for β and ζ described in section 3.5 are appropriate, keeping in mind the scaling of A as above may alter the value of some of the data used (so A should be scaled *before* calculating β or ζ). In the following section we examine how the circumstances of the second-stage of the technique affect the chioces of these scaling input parameters.

4.5 Zoom

The second-stage problems that arise from the refinement technique described above intentionally contain data of small magnitude, therefore requiring re-scaling, which we call "zooming" to distinguish from the usual scaling. Let us assume that the initial problem is already well scaled according to the methodology described in the previous section. If at the end of the first-stage solve we obtain $\widetilde{x}, \widetilde{y}, \widetilde{z}$ with 3-digit accuracy, we have by definition that dx, dy, dz will all be $O(10^{-3})$.

In addition, the second-stage problem has $\widetilde{b} = O(10^{-3})$ and objective gradient of order $O(10^{-3})$, which has a significant impact on all the other scaling factors. Therefore we choose $\beta = 10^{-3}$ and, assuming that A is already well scaled (the refinement step would not materially alter A), per (4.9) we now have a problem in

$$\widehat{\Delta x} = \frac{dx}{\|\widetilde{b}\|} \simeq 10^3 dx. \tag{4.10}$$

In addition, ζ is picked so that

$$\widehat{dz} = \frac{\widetilde{b} dz}{\|\zeta\|} \simeq 10^3 dz \text{ and, analogously, } \widehat{dy} \simeq 10^3 dy. \tag{4.11}$$

In particular, we point out that this yields

$$\widehat{\Delta x_j}\widehat{dz_j} \simeq 10^6 dx_j dz_j. \tag{4.12}$$

Now once $\widehat{dx}, \widehat{dy}, \widehat{dz}$ are computed to accuracy of 3 digits, the solution is unscaled, resulting in $x + dx, y + dy, z + dz$ accurate to 6 digits.

4.6 Convergence and Complexity

Theorem 11 already described the convergence results for Newton's method applied to a nonlinear optimization problem with inequality constraints replaced by a self-concordant barrier. Similar results also exist for the variant of this problem with additional equality constraints, which are handled explicitly, i.e. without being substituted by a barrier term in the objective.

If we define

$$c_0(x,r) = \phi(x) + \tfrac{1}{2}\|D_1 x\|^2 + d^T r + \tfrac{1}{2}\|r\|^2 + c_1^T x_1 + c_2^T x_2 + \kappa,$$

it is clear that $c_0(x,r)$ is convex if $\phi(x)$ is, because the other terms are linear or quadratic with respect to x or r. Given that the barrier used in PDCO is the classic logarithmic one, which enjoys all the properties of self-concordance, we conclude that the PDCO algorithm applied to NP3 and each phase of the "refine and zoom" variant applied to P3 have the same theoretical convergence properties as those expressed in Theorem 11.

The complexity results require more careful attention and depend on the following two lemmas which provide the bound on outer and inner iterations, respectively. Their proofs can be found in [50].

Lemma 1. *If the barrier parameter μ has the initial value μ^0 and is updated as $\mu^{k+1} = \sigma\mu^k$, with $0 < \sigma < 1$, then after at most*

$$\frac{1}{1-\sigma}\log\left(\frac{n\mu^0}{\varepsilon}\right) \tag{4.13}$$

iterations we have $n\mu \leq \varepsilon$.

Lemma 2. *For given $\sigma(0 < \sigma < 1)$, the number of inner iterations between two successive updates of the barrier parameter μ is not larger than*

$$2\left(1 + \sqrt{\frac{(1-\sigma)\sqrt{n}}{\sigma}}\right)^4. \tag{4.14}$$

Theorem 12. *The following expression is an upper bound for the total number of iterations required by the logarithmic barrier algorithm with line searches using the 2-phase zoom and refine technique:*

$$O\left(n\log\left(\frac{n^2\mu_1^0\mu_2^0}{\varepsilon}\right)\right), \tag{4.15}$$

where μ_1^0 and μ_2^0 are the starting duality gap values for phase 1 and phase 2, respectively. Note that μ_2^0 is not *the same as the ending (unscaled) duality value for phase 1.*

Proof. We assume the zoom and refine technique uses a convergence threshold of $\sqrt{\varepsilon}$ in each of its 2 phases. By multiplying the bound on outer iterations (4.14) and the bound on inner iterations (4.13), then rounding the product, if not integral, to the smallest integer above it, we obtain the following bound on iterations for the first phase:

$$\frac{1}{1-\sigma}\left(2\left(1+\sqrt{\frac{(1-\sigma)\sqrt{n}}{\sigma}}\right)^4\right)\log\left(\frac{n\mu_1^0}{\sqrt{\varepsilon}}\right).$$

Analogously, for the second phase of the zoom and refine technique, we have the following bound:

$$\frac{1}{1-\sigma}\left(2\left(1+\sqrt{\frac{(1-\sigma)\sqrt{n}}{\sigma}}\right)^4\right)\log\left(\frac{n\mu_2^0}{\sqrt{\varepsilon}}\right).$$

Summing the two, we have a total bound for the full 2-phase method:

$$\frac{1}{1-\sigma}\left(2\left(1+\sqrt{\frac{(1-\sigma)\sqrt{n}}{\sigma}}\right)^4\right)\log\left(\frac{n^2\mu_1^0\mu_2^0}{\varepsilon}\right).$$

Now, by taking σ to be a fixed constant independent of n ($n = 1/2$ makes the math simple), the bound on total iterations becomes $O\left(n\log\left(\frac{n^2\mu_1^0\mu_2^0}{\varepsilon}\right)\right)$, as desired. □

Chapter 5

Accelerating IPMs With Iterative Solvers

In this chapter we illustrate some of the problems encountered with IPMs based on iterative solvers and show how the zoom technique can be used to alleviate these difficulties. Numerical results are also given, comparing IPM performance with and without the zoom technique under various parameter choices.

5.1 Motivation

As described previously, when IPMs make use of iterative solvers, as further precision is sought in the solution, the overall computational effort required increases dramatically.

The particular problem that triggered the research on the zoom technique was one of image reconstruction, first studied by Nagy and Strakoš [40] in 2000, and later analyzed by Byunggyoo Kim in his 2002 dissertation [31].

Given an observed light intensity image b of stars in our galaxy, and a blurring operator A, we wish to find a reconstruction of the true image x such that Ax approximates b in the least squares sense. This amounts to solving the following non-negative least squares problem:

$$\begin{array}{ll} \underset{x,r}{\text{minimize}} & \lambda e^T x + \frac{1}{2}\|r\|^2 \\ \text{subject to} & Ax + r = b, \quad x \geq 0, \end{array}$$

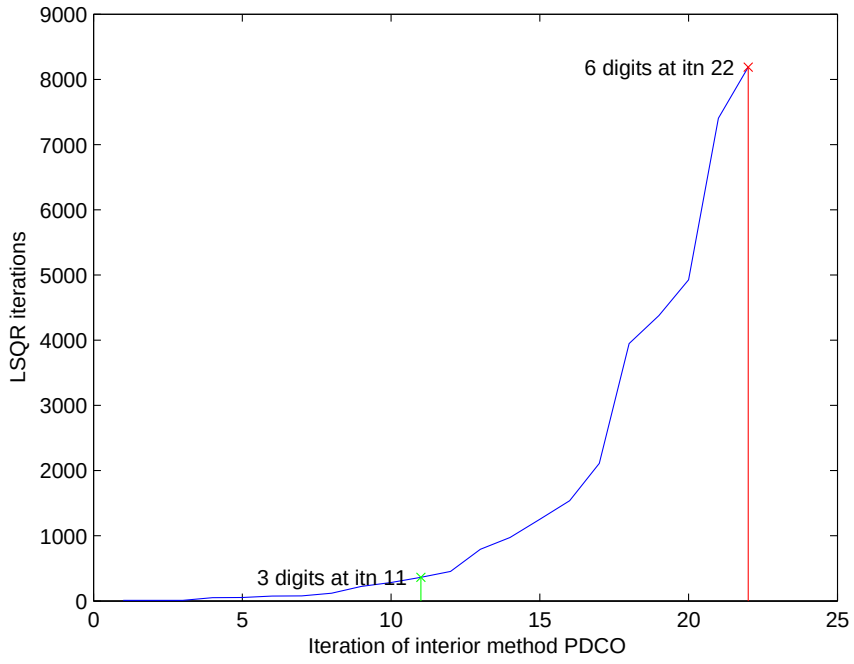

Figure 5.1: LSQR vs. PDCO iterations while solving the star problem

where r is the residual vector, and $\lambda = 10^{-4}$ helps redue the number of variables x_j that are significantly greater than zero. The true and blurred images are shown in figure 5.2.

PDCO has three algorithms available to calculate the search directions Δy:

- Cholesky factorization on

$$(AD^2A^T + \delta^2 I)\Delta y = AD^2 w + \delta r_1 \tag{5.1}$$

- Sparse QR factorization on

$$\min \left\| \begin{pmatrix} DA^T \\ \delta I \end{pmatrix} \Delta y - \begin{pmatrix} Dw \\ r_1 \end{pmatrix} \right\| \tag{5.2}$$

- an iterative least squares solver LSQR [45, 44] on (5.2).

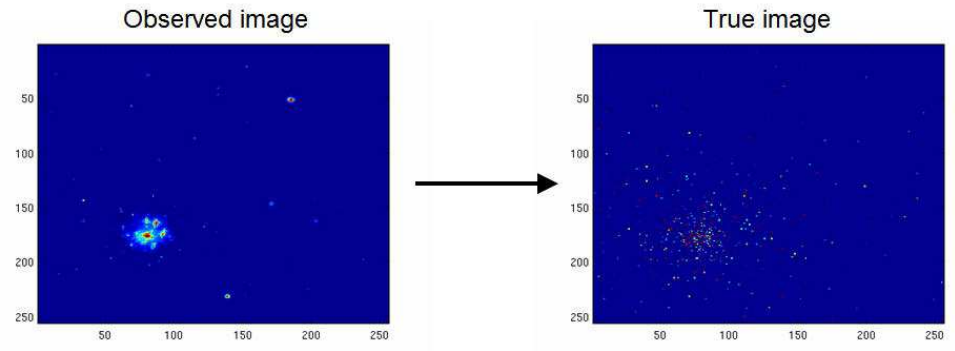

Figure 5.2: True and blurred image of star data

The operator A in this case turns out to be an expensive 2-dimensional (65,000 x 65,000) discretized 2-D convolution, so the only viable alternative is to make use of the LSQR routine. On a typical star problem solved using PDCO, however, the number of LSQR iterations grows exponentially with requested accuracy, as figure 5.1 demonstrates.

Since calculation of the search directions dominates runtime in such solves, it is clear that overall computational effort and runtime also grow exponentially.

PDCO will thus exhibit the behavior described above whenever LSQR is used, which is the only option when A is an operator as in the Star problem. Since Cholesky and QR factorizations also depend heavily on exploiting the sparsity of A for efficient performance, LSQR is again the method of choice when the constraint matrix or its factorizations are not sparse. Basis pursuit problems [6] and multi-commodity network LPs [47] are examples where this occurs.

The zoom technique is aimed at improving the performance on the classes of problems described above. While we here only analyze its effect on PDCO and the underlying LSQR, the potential benefit to similar algorithm packages is clear.

5.2 Why Zoom and Refine Works

The effects of ill-conditioning on interior methods have been studied extensively in the literature [10]. In the presence of degeneracy, it is well known

that as an interior method progresses, the linear system being solved becomes increasingly ill-conditioned. As a consequence, if an iterative method is being used for the linear system, the time to compute an accurate search direction increases steadily [31].

The benefit of the zoom method is that it attempts to reduce the ill-conditioning that typically plagues the later stages of interior methods. To understand the details of the inner workings of the zoom method, we revert to the simpler case of an interior method applied to an LP problem.

The standard LP problem

$$\boxed{\begin{array}{lll} \text{LP}(\mu) & \underset{x}{\text{minimize}} & c^T x + \mu \sum_j \ln(x_j) \\ & \text{subject to} & Ax = b, \quad x > 0 \end{array}}$$

has as its KKT conditions

$$\begin{aligned} Ax &= b, \\ A^T y + z &= c, \\ Xz &= \mu e. \end{aligned} \tag{5.3}$$

Applying Newton's method to (5.3) results in the system

$$\begin{pmatrix} A & & \\ & A^T & I \\ Z & & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \equiv \begin{pmatrix} b - Ax \\ c - A^T y - z \\ \mu e - Xz \end{pmatrix},$$

which can be reordered into

$$\begin{pmatrix} & I & A^T \\ Z & X & \\ A & & \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta z \\ \Delta y \end{pmatrix} = \begin{pmatrix} r_2 \\ r_3 \\ r_1 \end{pmatrix}.$$

We can then use I as a block pivot to eliminate Δz, yielding

$$\begin{pmatrix} Z & -XA^T \\ A & \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} r_3 - Xr_2 \\ r_1 \end{pmatrix} \tag{5.4}$$

and, multiplying the top of (5.4) by $-X^{-1}$ gives

$$\begin{pmatrix} -X^{-1}Z & A^T \\ A & \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} r_4 \\ r_1 \end{pmatrix}, \tag{5.5}$$

where $r_4 = r_2 - X^{-1}r_3$.

The condition number is defined as

$$\text{cond}(J) = \frac{\sigma_{\max}(J)}{\sigma_{\min}(J)},$$

where $\sigma_{\max}(J)$ and $\sigma_{\min}(J)$ are, respectively, the maximal and minimal singular values of J.

In [58], Wright shows that in the degenerate case, the matrix M in (5.5), has the property that its smallest singular value is $O(\mu)$ and thus

$$\text{cond}(M) \approx \frac{1}{\mu}$$

for all iterates near the central path, and μ sufficiently small.

Typical algorithms have stopping criteria based on the final value of μ so, considering the zoom method, at the end of the first stage with a target precision of 10^{-4} we have

$$\text{cond}(M_1) \approx \frac{1}{\mu} \simeq 10^4,$$

where M_1 represents the matrix M in (5.5) at the intermediate solution (end of first stage). However, as the next stage is started, the new problem is defined based on Δx and Δz instead, with appropriate scaling as shown in (4.9) for which the relationships (4.10)–(4.12) hold. Therefore, by design we have that $1 \simeq \mu_2 >> \mu_1$ and hence it is clear that $\text{cond}(M_2) \simeq 1$, from which we conclude that

$$\text{cond}(M_2) << \text{cond}(M_1),$$

where M_2 represents the matrix M in (5.5) at the starting point for the scaled new problem (at the start of second stage).

5.3 Numerical Results

The PDCO algorithm was used to solve the Star problem described above using two approaches: the usual single-phase solve and the two-phase method using the refine and zoom technique. The table below shows how the latter technique is approximately 30% faster than the usual approach.

problem	accuracy	LSQR itns.	time (s)
standard solve	10^{-6}	17,273	9,845
refine+zoom step 1	10^{-3}	382	235
refine+zoom step 2	10^{-3}	11,612	6,784
refine+zoom total	10^{-6}	11,994	7,019

Table 5.1: Solve time and LSQR iteration count for Star problem

Large problems such as Star, based on operators, are hard to come by. So the refine and zoom technique was also tested on a subset of the Netlib suite of LP problems. Even though the operator in this case is an explicit matrix, and therefore a direct method could be used on 5.1 or 5.2 for obtaining search directions, the use of the iterative LSQR routine to the 5.2 formulation was forced to simulate a non-explicit operator case. The table below shows the numerical results for these tests. Since the LSQR routine dominates the CPU time, the LSQR iteration count displayed can be also interpreted as a measure for comparing the computational solve time of the two approaches.

In the majority of cases where the standard solve method had a comparatively low LSQR iteration count to begin with, the refine and zoom approach had generally comparable iteration counts, with a few instances slighly above or below the standard approach. However, in the more complex problems, the refine and zoom approach outperformed the standard approach every time. Of the problems where the standard solve took over 15,000 LSQR iterations, refine and zoom technique had a performance improvement ranging between 2 and 83%, with an average of 28%, in line with the results obtained with the Star problem. It is worth noting that the overall size of the constraint matrices in the Netlib problems (the details of which can be found in the Appendix) is smaller than that of Star by a factor of 100,000.

problem name	standard solve LSQR itns	refine+zoom total LSQR itns
adlittle	1579	1761
afiro	395	394
agg	8136	7793
bandm	25867	13837
beaconfd	612	684
blend	3776	3679
bore3d	4832	5276
brandy	27874	22228
capri	303546	261985
degen2	461399	311783
e226	12496	14463
etamacro	7283	7011
finnis	16549	11973
gfrd_pnc	6271	6268
grow7	5068	6413
israel	6721	7756
kb2	2775	3730
lotfi	9451	8263
qap8	246762	40353
recipe	3901	2959
sc105	2514	2846
sc205	6776	5374
sc50a	1138	1131
sc50b	856	907
scagr25	24347	20858
scagr7	4236	3938
scfxm1	46965	30867
scorpion	33758	20909
scrs8	17913	15458
scsd1	855	922
sctap1	13754	10406
share1b	18438	18012
share2b	11197	9983
shell	12856	12768
stair	46746	41689
standata	3938	3872
standgub	4073	3653
standmps	8956	9222
stocfor1	4527	3369
vtp_base	4613	4714

Table 5.2: LSQR iteration count for standard solve vs. refine and zoom applied to Netlib LP problems

Chapter 6

LP Warmstarting for IPMs with Direct Solvers

The novel technique presented here differs from previous ones found in the literature in that it requires few changes to existing solvers, being primarily a modeling approach rather than an algorithmic one. Given two problems, LP1, LP2, and a solution (x, y, z) to LP1, the strategy is to define a new problem LP3' in terms of a correction solution (dx, dy, dz) so that $(x+dx, y+dy, z+dz)$ solves LP2.

6.1 Motivation

The difficulties with warmstarting in interior point methods are well known [19, 59, 46, 2, 35]. We exhibit here only a summary of what happens inside a typical solver. The example below was first presented by Benson and Shanno [2].

Consider the following LP problem:

LP-x	$\underset{x_1,x_2}{\text{minimize}}$	$x_1 + 3x_2$
	subject to	$x_1 + x_2 \leq 3$
		$2x_1 + x_2 \leq 2$
		$x_1,\ x_2 \geq 0.$

The optimal primal and dual solutions for this problem are

$$\begin{aligned} x &= (0, 2), \qquad w = (1, 0), \\ y &= (0, 3), \qquad z = (5, 0). \end{aligned} \tag{6.1}$$

Therefore, we have that the first constraint is inactive and the second constraint is active at the optimum.

Next, if the first constraint is altered to be

$$x_1 + x_2 \leq 1,$$

then the optimal primal and dual solutions for the modified problem become

$$\begin{array}{ll} x = (0,1), & w = (0,1), \\ y = (3,0), & z = (2,0). \end{array}$$

Hence, the first constraint has now become active while the second has become inactive.

The reduced KKT system for this problem is

$$\begin{pmatrix} -E & A \\ A^T & F \end{pmatrix} \begin{pmatrix} \Delta y \\ \Delta x \end{pmatrix} = \begin{pmatrix} \rho - E\gamma_w \\ \sigma + \gamma_z \end{pmatrix}, \tag{6.2}$$

where

$$\begin{aligned} E &= Y^{-1}W \\ F &= X^{-1}Z \\ \rho &= b - Ax - w \\ \sigma &= c - A^T y + z \\ \gamma_w &= \mu W^{-1}e - y \\ \gamma_z &= \mu X^{-1}e - z, \end{aligned}$$

and the following were block eliminated in the process:

$$\begin{aligned} \Delta z &= \gamma_z - F\Delta x \\ \Delta w &= E(\gamma_w - \Delta y). \end{aligned}$$

Now when we solve the modified problem with the warmstart solutions (6.1), the only change to the reduced KKT system is to primal infeasibility, ρ, for the inactive constraint. The corresponding entry of the diagonal matrix, E, is of order 10^8 (theoretically it is $1/0$). Therefore, a very small value for this constraint's Δy allows the system to be satisfied. In fact, in the first several iterations, Δx and Δy are both no more than 10^{-8} for all components.

Second, the barrier parameter, μ, starts at 0, as its computation,

$$\mu = r\frac{w^T y + x^T z}{m+n}$$

is not affected by the perturbation to the data. Hence, the step direction formulas for the slack variables result in Δz also being close to 0. This means that primal feasibility also can be restored without disturbing the primal variables, and by simply moving the primal slacks. In the first iteration, we have $\Delta w_1 = -2$ to obtain feasibility, but the steplength α_p is cut to prevent $w_1 + \alpha_p \Delta w_1$ from becoming negative.

The numerical trouble occurs when the short steps decrease w_1 to a value sufficiently close to 0. Since y_1 has not moved away from 0, we are now at a degenerate pair. AD^2A becomes almost singular. Cholesky factorization and the subsequent backsolve start producing numerical errors, and iterative refinement measures are required. Such measures, however, provide step directions that move the iterates away from the warmstart solution. In this case they have the effect of moving the dual variables to values around 1.5 and the dual slack z_1 to around 3.05.

Continuing from this point, the algorithm finds the optimal solution in another 12 iterations, but after losing all the advantage of a warmstart. In fact, the coldstart solution of the same problem takes 12 iterations while the warmstart takes a total of 15.

We show below how the zoom technique (for accelerating coldstarts) can also be applied to alleviate the difficulties just described.

6.2 LP Zoomstarting/Jumpstarting

In the context of interior point methods, LP problems that differ by oly a small perturbation in their data have solutions that are typically not too far apart. The challenges described in the previous section show, however, that the solution to one problem can be a poorly centered starting point for the perturbed problem, thus leading to solve times that could be longer than a normal coldstart solve. Though the numerical challenges are somewhat different, the technique described in section 4.2 for accelerating interior methods based on iterative solvers is also applicable here.

Unlike typical warmstart methods where the perturbed problem is solved using the original solution as a starting point, the key to the strategy here,

which we'll call "zoomstarting", is to define a new problem based on the difference vector between the two solutions, and appropriately scale the problem so we are in essence performing a simple, short, coldstart solve. In this sense, the formulation of problem LP3' can be adopted, making use of the data (A, b, c, etc.) from the perturbed problem and the variables ($\widetilde{x}, \widetilde{y}, \widetilde{z}$, etc.) from the solution to the original problem.

In this case, the theory for convergence and complexity results for a "zoomstart" are the same as a normal coldstart solve for any typical LP problem, as described in sections 2.2.2 and 2.3.2. Analogously to the case of acceleration of IPMs, scaling is key to why the "zoomstarting" method works, since its effect is in essence to ensure the starting point is well removed from the bounds on each variable.

The advantage of "zoomstarting" over coldstart is that in the latter one must solve the problem with the usual precision threshold of, say, 10^{-6}, whereas in the former, less precision is needed (10^{-3} or 10^{-4} say), depending on the proximity between the original and perturbed solutions. Given the dependency of the complexity results (2.6) on the precision threshold ε, the benefit here is clear.

6.3 Numerical Results

The Netlib suite used by Benson and Shanno for their numerical results [2] is the set we use here. While they have confined themselves to a small subset of the smallest problems in the set, where the sum of the dimensions of the constraint matrix A is less than $1,000$, we have expanded to all problems where the problem file is smaller than 100Kb in size. The perturbations performed on the problems are identical to those found in [2], though the formulation below is slightly different, for clarification purposes. The perturbations are based on two uniformly distributed random variables, $e_1 \in (0,1)^m$, $e_2 \in (-1,1)^m$:

$$\begin{aligned} b_i \quad & \text{is perturbed if } e_{1i} > \max(.9, 1 - \frac{20}{m}) \\ \widetilde{b_i} \quad & = \begin{cases} \delta e_{2i} & \text{if } b_i = 0; \\ b_i(1 + \delta e_{2i}) & \text{otherwise.} \end{cases} \end{aligned}$$

This means that no more than 10% (if small m) or 20 constraints (if large m) are perturbed, on average. In addition, the magnitude of the perturba-

tions is small: $\delta \in \{0.001, 0.01, 0.1\}$. In practice, perturbations are likely to be made to other data in the problem. To mimic such conditions, analogous perturbations are made for A and c, which are tested separately. Finally, for each value of δ and a given data component to be perturbed, 5 sets of random variables are generated, for a total 45 runs per problem. The results below represent the averages over all 5 random variable sets.

Within PDCO, the LP problems are regularized to the NP2 form and the routine based on Cholesky factorizations discussed in section 3.4 is used for calculating the search directions.

Table 6.1 shows the summary statistics of average performance over 41 Netlib problems. The complete tables of results can be found at the end of this chapter.

	$\delta = 0.1$	$\delta = 0.01$	$\delta = 0.001$
A pert.	0.91	0.77	0.78
b pert.	0.80	0.50	0.43
c pert.	0.86	0.74	0.66

Table 6.1: Average ratio of PDCO iterations (warmstart/coldstart)

Figure 6.1 shows the ratio of warmstart to coldstart iterations vs. the distance of solutions for perturbations applied to b, where each point on the graph represents a problem from the Netlib suite for each of the three values of δ. The graph is restricted to the subset of problems that have average solution distance $\Delta \leq 1$ less than or equal to one where, given the solutions to the original and perturbed problems x^O and x^P respectively,

$$\Delta = \frac{\|x^O - x^P\|}{1 + \|x^O\|}.$$

The clustering of points around the 0.6 mark on the vertical axis of Figure 6.1 indicates that warmstarting using the zoom technique had an efficiency gain between 57% and 9% compared to coldstart runs, with an improvement of 40%. One would expect the benefit of the zoom technique to be reduced as Δ increases, but the figure shows clearly that this is not quite the case. The summary results in Table 6.1 indeed show the reduced benefit effect, but it only happens when the distance between the original and perturbed solutions is very large. On the other hand, some problems exhibited *increased* iteration count under the zoom technique despite a small average Δ.

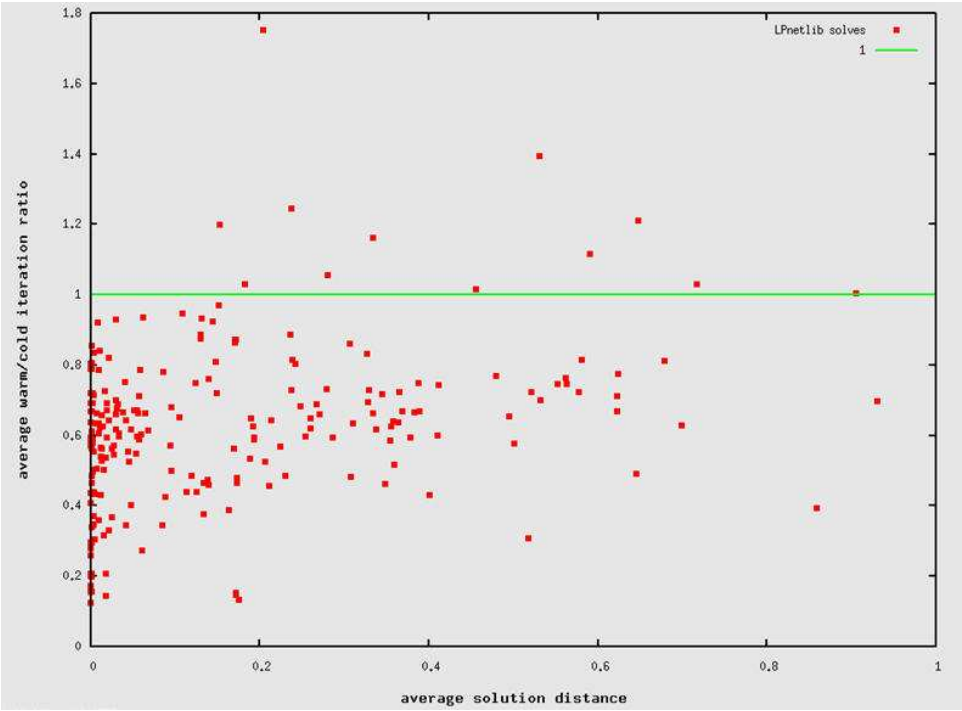

Figure 6.1: PDCO iterations (warmstart/coldstart) vs. perturbation to x, y

The tables below present the numerical results, by problem, for each perturbation to A, b, c and for each $\delta \in \{0.001, 0.01, 0.1\}$. Basic information about the Netlib LP problems used can be found in the Appendix and the full set of Netlib LP problems and more information about them can be found at http://www.netlib.org/lp/data/. It is worth noting that since the CPU time is directly proportional to the number of PDCO (outer) iterations in this case, the average iteration ratio displayed in the tables below can also be interpreted as an average CPU time ratio.

A perturbed	$\delta = 0.1$	
problem name	avg soln dist	avg ratio
adlittle	5.52e−01	0.75
afiro	6.24e−01	0.77
agg	2.84e+03	1.17
bandm	9.15e+04	0.78
beaconfd	1.57e+02	0.83
blend	2.42e−01	0.8
bore3d	1.77e+06	1.16
brandy	1.45e+04	1.47
capri	1.00e+01	1.29
degen2	3.07e+01	1.05
e226	1.48e+01	0.93
etamacro	1.76e+00	0.74
finnis	1.45e+02	0.69
gfrd_pnc	1.84e+01	0.86
grow7	1.28e+00	0.47
israel	2.24e−01	0.57
kb2	5.65e+00	1.41
lotfi	3.69e−01	0.67
qap8	1.91e+01	0.89
recipe	4.88e+02	1.35
sc105	2.80e−01	1.05
sc205	8.58e−02	0.78
sc50a	6.44e−02	0.66
sc50b	4.27e+00	0.8
scagr25	6.48e−01	1.21
scagr7	1.13e+00	0.97
scfxm1	1.14e+03	1
scorpion	6.67e+03	1.15
scrs8	3.64e−01	0.64
scsd1	1.09e−01	0.95
sctap1	2.37e−01	0.89
share1b	6.23e−01	0.71
share2b	5.91e−01	1.11
shell	6.28e+01	0.72
stair	8.28e+00	1.04
standata	6.45e−01	0.49
standgub	6.79e−01	0.81
standmps	5.63e+03	0.73
stocfor1	4.21e+01	1.32
vtp_base	7.43e+02	1.48

Table 6.2: Warmstart/coldstart ratios with perturbations to A, $\delta = 0.1$

A perturbed	$\delta = 0.01$	
problem name	avg soln dist	avg ratio
adlittle	3.56e−01	0.63
afiro	6.23e−01	0.67
agg	4.45e+02	0.87
bandm	1.24e+00	0.65
beaconfd	1.85e+02	0.88
blend	3.03e−02	0.66
bore3d	4.39e+06	1.24
brandy	1.26e+05	1.47
capri	1.05e+00	0.84
degen2	8.62e+00	0.79
e226	3.58e−01	0.64
etamacro	9.90e+00	0.8
finnis	3.24e+01	0.66
gfrd_pnc	1.96e+02	0.82
grow7	1.19e−01	0.48
israel	4.40e−02	0.55
kb2	1.20e+00	1.03
lotfi	3.38e−01	0.61
qap8	3.24e+01	0.93
recipe	2.23e+02	1.12
sc105	4.83e−02	0.62
sc205	5.70e−02	0.71
sc50a	3.00e−02	0.7
sc50b	4.99e+01	0.74
scagr25	5.65e−02	0.66
scagr7	3.78e−01	0.59
scfxm1	6.95e+00	0.73
scorpion	2.38e+03	0.79
scrs8	3.11e−01	0.63
scsd1	1.45e−01	0.92
sctap1	1.50e−01	0.72
share1b	5.33e−01	0.7
share2b	2.79e−01	0.73
shell	1.89e+00	0.63
stair	4.11e−01	0.6
standata	2.70e+00	0.54
standgub	9.06e−01	1
standmps	2.46e+02	0.79
stocfor1	4.56e−01	1.01
vtp_base	1.18e+02	1.15

Table 6.3: Warmstart/coldstart ratios with perturbations to A, $\delta = 0.01$

A perturbed	$\delta = 0.001$	
problem name	avg soln dist	avg ratio
adlittle	5.22e−01	0.72
afiro	5.78e−01	0.72
agg	5.81e−01	0.82
bandm	4.05e+00	0.68
beaconfd	2.89e+02	0.96
blend	5.81e−02	0.79
bore3d	1.15e+08	1.06
brandy	1.01e+06	1.58
capri	3.29e−01	0.73
degen2	4.80e−01	0.77
e226	9.30e−01	0.7
etamacro	1.05e+02	0.85
finnis	1.23e+01	0.7
gfrd_pnc	2.38e+03	0.94
grow7	3.08e−01	0.48
israel	1.36e−02	0.54
kb2	1.05e+00	1.02
lotfi	3.28e−01	0.69
qap8	4.29e+01	0.96
recipe	2.59e+03	1.18
sc105	4.12e−01	0.74
sc205	2.37e−01	0.73
sc50a	1.48e−01	0.81
sc50b	5.07e+02	0.85
scagr25	5.50e−02	0.67
scagr7	5.62e−01	0.76
scfxm1	6.02e+01	0.75
scorpion	7.09e+02	0.78
scrs8	1.49e+00	0.63
scsd1	1.83e−01	1.03
sctap1	1.40e−01	0.76
share1b	3.65e−01	0.72
share2b	3.28e−01	0.83
shell	2.67e−01	0.69
stair	3.59e−01	0.52
standata	1.17e+01	0.62
standgub	1.32e+00	1.13
standmps	1.83e+01	0.79
stocfor1	7.18e−01	1.03
vtp_base	1.31e+02	0.85

Table 6.4: Warmstart/coldstart ratios with perturbations to A, $\delta = 0.001$

b perturbed	$\delta = 0.1$	
problem name	avg soln dist	avg ratio
adlittle	5.31e−01	1.39
afiro	2.55e−01	0.6
agg	4.57e+00	0.7
bandm	2.39e+02	0.64
beaconfd	2.11e+03	0.61
blend	2.56e−02	0.56
bore3d	2.13e+04	1.14
brandy	6.77e+02	1.85
capri	2.19e−02	0.82
degen2	4.21e+02	0.84
e226	1.34e+04	2.12
etamacro	1.42e+03	0.66
finnis	7.28e+02	0.62
gfrd_pnc	1.64e−01	0.39
grow7	9.63e+01	0.3
israel	9.45e−02	0.57
kb2	4.67e−04	0.2
lotfi	2.39e−01	0.82
qap8	5.50e+04	1.57
recipe	1.07e+03	0.81
sc105	1.65e+01	0.81
sc205	3.30e−02	0.69
sc50a	1.55e+01	0.61
sc50b	3.61e+01	0.49
scagr25	1.94e−02	0.59
scagr7	4.23e−02	0.34
scfxm1	8.72e+00	0.86
scorpion	6.37e+04	2.09
scrs8	1.36e+03	1.07
scsd1	2.04e−01	1.75
sctap1	3.07e−01	0.86
share1b	1.34e−01	0.38
share2b	4.56e−02	0.52
shell	1.53e−02	0.31
stair	2.13e−01	0.64
standata	3.12e+02	0.46
standgub	7.34e+03	0.49
standmps	7.39e+02	0.45
stocfor1	3.10e−02	0.68
vtp_base	1.72e+01	1.21

Table 6.5: Warmstart/coldstart ratios with perturbations to b, $\delta = 0.1$

b perturbed	$\delta = 0.01$	
problem name	avg soln dist	avg ratio
adlittle	2.19e−02	0.33
afiro	2.30e−01	0.48
agg	4.02e−01	0.43
bandm	2.38e+01	0.36
beaconfd	2.09e+02	0.45
blend	3.02e−03	0.55
bore3d	8.00e+02	0.68
brandy	5.26e+00	0.51
capri	5.12e−03	0.43
degen2	4.20e+01	0.73
e226	6.99e+01	0.6
etamacro	1.42e+02	0.43
finnis	7.28e+01	0.47
gfrd_pnc	9.41e−03	0.36
grow7	5.80e+00	0.19
israel	1.77e−03	0.2
kb2	4.67e−05	0.2
lotfi	4.78e−02	0.4
qap8	5.50e+03	0.96
recipe	1.07e+02	0.71
sc105	1.70e+00	0.61
sc205	3.37e−02	0.6
sc50a	1.69e+00	0.61
sc50b	3.60e+00	0.49
scagr25	4.14e−03	0.44
scagr7	4.58e−03	0.3
scfxm1	8.60e−01	0.39
scorpion	6.30e+03	1.41
scrs8	1.36e+02	0.52
scsd1	1.52e−01	1.2
sctap1	2.60e−01	0.62
share1b	3.49e−04	0.12
share2b	7.71e−03	0.5
shell	1.54e−03	0.3
stair	6.11e−02	0.27
standata	3.12e+01	0.44
standgub	7.33e+02	0.44
standmps	7.39e+01	0.38
stocfor1	3.35e−02	0.6
vtp_base	1.56e+00	0.69

Table 6.6: Warmstart/coldstart ratios with perturbations to b, $\delta = 0.01$

b perturbed	$\delta = 0.001$	
problem name	avg soln dist	avg ratio
adlittle	3.21e−03	0.35
afiro	2.11e−01	0.46
agg	2.55e−02	0.37
bandm	2.28e+00	0.29
beaconfd	2.09e+01	0.35
blend	2.84e−03	0.5
bore3d	8.00e+01	0.6
brandy	5.18e−01	0.31
capri	1.62e−03	0.34
degen2	4.10e+00	0.56
e226	7.08e+00	0.41
etamacro	1.39e+01	0.33
finnis	7.20e+00	0.37
gfrd_pnc	2.03e−03	0.34
grow7	3.50e−06	0.26
israel	1.82e−04	0.2
kb2	4.67e−06	0.2
lotfi	3.14e−03	0.37
qap8	5.50e+02	0.82
recipe	1.07e+01	0.67
sc105	1.89e−01	0.53
sc205	2.96e−02	0.62
sc50a	1.74e−01	0.48
sc50b	3.48e−01	0.46
scagr25	6.26e−04	0.41
scagr7	5.37e−04	0.28
scfxm1	8.44e−02	0.34
scorpion	6.03e+02	1
scrs8	1.34e+01	0.4
scsd1	1.52e−01	0.97
sctap1	1.73e−01	0.47
share1b	3.49e−05	0.12
share2b	1.87e−03	0.5
shell	1.54e−04	0.3
stair	4.98e−04	0.17
standata	3.13e+00	0.45
standgub	7.31e+01	0.47
standmps	7.32e+00	0.37
stocfor1	2.76e−02	0.54
vtp_base	8.85e−02	0.42

Table 6.7: Warmstart/coldstart ratios with perturbations to b, $\delta = 0.001$

c perturbed	$\delta = 0.1$	
problem name	avg soln dist	avg ratio
adlittle	3.83e−01	0.66
afiro	3.89e−01	0.67
agg	1.71e−02	0.73
bandm	6.84e−02	0.61
beaconfd	1.28e+02	0.85
blend	1.30e−01	0.89
bore3d	6.18e−02	0.93
brandy	1.55e+02	0.68
capri	5.63e−01	0.75
degen2	1.05e−01	0.65
e226	4.90e+01	0.7
etamacro	2.49e−01	0.68
finnis	5.45e−02	0.6
gfrd_pnc	2.20e−02	0.64
grow7	9.58e−02	0.5
israel	5.41e−02	0.55
kb2	3.88e−01	0.75
lotfi	3.37e+02	3.11
qap8	1.32e−01	0.93
recipe	8.63e+05	1.51
sc105	1.71e+00	2.21
sc205	5.79e+00	2.42
sc50a	2.37e−01	1.24
sc50b	3.34e−01	1.16
scagr25	4.19e−02	0.64
scagr7	2.06e−01	0.52
scfxm1	3.85e−02	0.66
scorpion	9.55e−02	0.68
scrs8	2.71e−02	0.57
scsd1	1.72e−01	0.87
sctap1	1.92e−01	0.62
share1b	6.99e−01	0.63
share2b	1.94e+00	1.11
shell	6.58e−03	0.63
stair	4.95e−01	0.65
standata	1.26e−01	0.44
standgub	1.38e−01	0.47
standmps	1.89e−02	0.69
stocfor1	1.01e−02	0.79
vtp_base	1.58e−03	0.67

Table 6.8: Warmstart/coldstart ratios with perturbations to c, $\delta = 0.1$

c perturbed	$\delta = 0.01$	
problem name	avg soln dist	avg ratio
adlittle	3.55e−01	0.58
afiro	3.34e−01	0.66
agg	1.95e−03	0.72
bandm	6.02e−02	0.6
beaconfd	1.30e+01	0.86
blend	4.00e−03	0.83
bore3d	2.97e−02	0.93
brandy	1.55e+01	0.6
capri	1.19e−03	0.81
degen2	1.35e−02	0.66
e226	4.91e+00	0.61
etamacro	2.71e−01	0.66
finnis	5.72e−02	0.59
gfrd_pnc	7.60e−04	0.67
grow7	1.59e−02	0.5
israel	1.15e−02	0.56
kb2	5.10e−02	0.67
lotfi	7.60e+00	2.42
qap8	1.31e−01	0.88
recipe	1.10e+05	2.07
sc105	4.10e−02	0.75
sc205	4.51e+00	1.45
sc50a	1.97e−02	0.67
sc50b	1.37e−02	0.56
scagr25	1.06e−03	0.61
scagr7	1.27e−02	0.53
scfxm1	9.40e−03	0.63
scorpion	1.52e−03	0.57
scrs8	2.60e−03	0.59
scsd1	1.71e−01	0.86
sctap1	1.93e−01	0.59
share1b	1.70e−01	0.56
share2b	3.45e−01	0.72
shell	5.42e−04	0.64
stair	1.40e−01	0.46
standata	1.14e−01	0.44
standgub	1.33e−01	0.46
standmps	1.87e−03	0.69
stocfor1	8.50e−04	0.79
vtp_base	1.58e−04	0.67

Table 6.9: Warmstart/coldstart ratios with perturbations to c, $\delta = 0.01$

c perturbed	$\delta = 0.001$	
problem name	avg soln dist	avg ratio
adlittle	2.86e−01	0.59
afiro	2.61e−01	0.65
agg	1.95e−04	0.72
bandm	9.97e−03	0.6
beaconfd	1.28e+00	0.81
blend	8.60e−04	0.85
bore3d	7.87e−03	0.92
brandy	1.57e+00	0.58
capri	2.10e−04	0.8
degen2	7.11e−03	0.66
e226	5.02e−01	0.58
etamacro	1.41e−02	0.63
finnis	2.76e−03	0.58
gfrd_pnc	7.80e−05	0.67
grow7	1.59e−03	0.48
israel	2.73e−03	0.58
kb2	1.12e−02	0.62
lotfi	1.89e−01	0.65
qap8	1.24e−01	0.75
recipe	1.10e+04	1.71
sc105	3.89e−03	0.71
sc205	1.07e−02	0.84
sc50a	1.74e−03	0.61
sc50b	1.36e−03	0.56
scagr25	7.65e−05	0.59
scagr7	1.20e−02	0.54
scfxm1	8.37e−03	0.63
scorpion	2.13e−04	0.57
scrs8	1.92e−04	0.59
scsd1	1.71e−01	0.87
sctap1	1.93e−01	0.59
share1b	1.78e−02	0.54
share2b	2.38e−01	0.73
shell	5.55e−05	0.64
stair	1.19e−02	0.43
standata	2.68e−04	0.44
standgub	8.20e−04	0.46
standmps	1.63e−04	0.69
stocfor1	8.54e−05	0.79
vtp_base	1.58e−05	0.67

Table 6.10: Warmstart/coldstart ratios with perturbations to c, $\delta = 0.001$

Bibliography

[1] K. M. Anstreicher. Potential reduction algorithms. Technical report, Department of Management Sciences, University of Iowa, Iowa City, Iowa 52242, 1995. *Interior Point Methods in Mathematical Programming*, T. Terlaky, editor (Kluwer, 1996).

[2] H. Y. Benson and D. F. Shanno. An exact primal-dual penalty method approach to warmstarting interior-point methods for linear programming. Technical report, Drexel University, Philadelphia, PA, September 2005.

[3] D. Bertsekas and A. Ozdaglar. Pseudonormality and a Lagrange multiplier theory for constrained optimization. Technical report, 2000.

[4] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 2nd edition, 1999.

[5] S. P. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, Cambridge, UK, 2004.

[6] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998. Also appeared in SIAM Review 43(1), 129-159 (2001).

[7] CPLEX Optimization Inc. CPLEX Linear Optimizer and Mixed Integer Optimizer. Suite 279, 930 Tahoe Blvd. Bldg 802, Incline Village, NV 89541.

[8] J. Czyzyk, S. Mehrotra, M. Wagner, and S. J. Wright. PCx: An interior-point code for linear programming. Technical report, 1999.

[9] Dash Associates, Ltd. XPRESS-MP. Blisworth, Northants, England.

[10] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19(2):400–408, 1982.

[11] A. S. El-Bakry, R. A. Tapia, T. Tsuchiya, and Y. Zhang. On the formulation and theory of the primal-dual newton interior-point method for nonlinear programming. *Journal of Optimization Theory and Applications*, 89:507–541, 1996.

[12] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley and Sons, New York, 1968. Reprinted as Volume 4 of the SIAM Classics in Applied Mathematics Series, 1990.

[13] Anthony V. Fiacco and Garth P. McCormick. *Nonlinear Programming*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 1990. Reprint of the 1968 original.

[14] R. Fletcher. *Practical Methods of Optimization, 2nd Edition*. John Wiley, Chichester, 2000.

[15] Roger Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, Chichester and New York, second edition, 1987.

[16] A. Forsgren, P. E. Gill, and M. H. Wright. Interior methods for nonlinear optimization. *SIAM Review*, 44(4):525–597, 2002.

[17] Anders Forsgren, Philip E. Gill, and Margaret H. Wright. Interior methods for nonlinear optimization. *SIAM Rev.*, 44:525–597, 2002.

[18] R. Fourer and S. Mehrotra. Performance of an augmented system approach for solving least-squares problems in an interior-point method for linear programming. *Math. Program.*, 19:26–31, August 1991.

[19] R. M. Freund. A potential–function reduction algorithm for solving a linear program directly from an infeasible "warm start". *Mathematical Programming*, 52:441–466, 1991.

[20] R. M. Freund and S. Mizuno. Interior point methods: Current status and future directions. *Optima*, 51:1–9, 1996.

[21] K. R. Frisch. The logarithmic potential method of convex programming. Memorandum of May 13, University Institute of Economics, Oslo, Norway, 1955.

[22] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright. On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. *Mathematical Programming*, 36:183–209, 1986.

[23] F. Glineur. *Topics in Convex Optimization: Interior-Point Methods, Conic Duality and Approximations*. PhD thesis, Mons, Belgium, January 2001.

[24] J. Gondzio. HOPDM (ver 2.12) — A fast LP solver based on a primal-dual interior point method. *European Journal of Operational Research*, 85:221–225, 1995.

[25] J. Gondzio and T. Terlaky. A computational view of interior-point methods for linear programming. Delft University of Technology, Faculty of Technical Mathematics and Informatics, 1994.

[26] Jacek Gondzio and Andreas Grothey. A new unblocking technique to warmstart interior point methods based on sensitivity analysis. Technical Report MS-06-005, School of Mathematics, The University of Edinburgh, Mayfield Road, Edinburgh EH9 3JZ, UK, 2006.

[27] C. C. Gonzaga. Path following methods for linear programming. *SIAM Review*, 34(2):167–224, 1992.

[28] IBM. *IBM Optimization Subroutine Library Guide and Reference*, August 1990. Publication number SC23–0519–1.

[29] E. John and E. A. Yildirim. Implementation of warm-start strategies in interior-point methods for linear programming in fixed dimension. Technical report, Bilkent, Ankara, Turkey, 2006.

[30] N. K. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[31] B. Kim. *Numerical Optimization Methods for Image Restoration*. PhD thesis, December 2002.

[32] M. Kojima, S. Mizuno, and A. Yoshise. A primal-dual interior point algorithm for linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior Point and Related Methods*, pages 29–47. Springer Verlag, New York, NY, 1989.

[33] M. Kojima, S. Mizuno, and A. Yoshise. An iteration potential reduction algorithm for linear complementarity problems. *Mathematical Programming*, 50(1):331–342, 1991.

[34] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Computational experience with a primal-dual interior point method for linear programming. *Linear Algebra Appl.*, 152:191–222, 1991.

[35] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing*, 6(1):1–14, 1994. See also the following commentaries and rejoinder.

[36] K. A. McShane, C. L. Monma, and D. F. Shanno. An implementation of a primal-dual interior point method for linear programming. *ORSA Journal on Computing*, 1:70–83, 1989.

[37] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior Point and Related Methods*, pages 131–158. Springer Verlag, NY, 1989.

[38] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[39] Cs. Mészáros. The BPMPD interior solver for convex quadratic problems. *Optimization Methods and Software*, 11:431–449, 1999.

[40] J. G. Nagy and Z. Strakos. Enforcing nonnegativity in image reconstruction algorithms. *Mathematical Modeling, Estimation, and Imaging*, 4121:182–190, 2000.

[41] Y. E. Nesterov and A. S. Nemirovsky. *Interior Point Polynomial Methods in Convex Programming: Theory and Algorithms*. SIAM Publications. SIAM, Philadelphia, USA, 1993.

[42] J. Nocedal, A. Wachter, and R.Ã. Waltz. Adaptive barrier strategies for nonlinear interior methods. *Research Report RC*, 23563, 2005.

[43] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag, New York, second edition, 2006.

[44] C. C. Paige and Michael A. Saunders. Algorithm 583; LSQR: Sparse linear equations and least-squares problems. *ACM Trans. Math. Softw.*, 8(2):195–209, 1982.

[45] Christopher C. Paige and Michael A. Saunders. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8(1):43–71, 1982.

[46] R. Polyak. Modified barrier functions (theory and methods). *Mathematical Programming*, 54:177–222, 1992.

[47] M. G. C. Resende and P. M. Pardalos. Interior point methods for large-scale linear programming. In *Handbook of Optimization in Telecommunications*, pages 3–25. 2006.

[48] Stephen M. Robinson. Generalized equations and their solutions. II. Applications to nonlinear programming. *Mathematical Programming Study*, 19:200–221, 1982.

[49] C. Roos. A full-newton step $o(n)$ infeasible interior-point algorithm for linear optimization. *SIAM Journal on Optimization*, 16(4):1110–1136, 2006.

[50] C. Roos, T. Terlaky, and J.-Ph. Vial. *Theory and Algorithms for Linear Optimization: An Interior Point Approach*. John Wiley, Chichester, 1997.

[51] R. Saigal. *Linear Programming: A Modern Integrated Analysis*. Kluwer Academic Publishers, Boston, MA, 1995.

[52] M. A. Saunders. PDCO: MATLAB software for convex optimization. http://www.stanford.edu/group/SOL/software.html.

[53] M. J. Todd. Potential-reduction methods in mathematical programming. *Mathematical Programming*, 76:3–45, 1997.

[54] J. A. Tomlin. A note on comparing simplex and interior methods for linear programming. pages 91–104, 1989.

[55] R. J. Vanderbei. LOQO optimization software. http://orfe.princeton.edu/~loqo/.

[56] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, Boston, 1996. Second Edition: 2001.

[57] R. J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 11:451–484, 1999.

[58] S. J. Wright. Stability of augmented system factorizations in interior-point methods. *SIAM J. Matrix Anal. Appl.*, 18:191–222, 1997.

[59] E. A. Yildirim and S. J. Wright. Warm-start strategies in interior-point methods for linear programming. *SIAM Journal on Optimization*, 12(3):692–714, 2002.

[60] Y. Zhang. User's guide to LIPSOL: Linear programming interior point solvers v0.4. *Optimization Methods and Software*, 11:385–396, 1999.

Chapter 7

Appendix

problem name	# rows	# cols	# nonzeros
adlittle	57	97	465
afiro	28	32	88
agg	469	163	2541
bandm	306	472	2659
beaconfd	174	262	3476
blend	75	83	521
bore3d	234	315	1525
brandy	221	249	2150
capri	272	253	1786
degen2	445	534	4449
e226	224	282	2767
etamacro	401	668	2489
finnis	498	614	2714
gfrd_pnc	617	1092	3467
grow7	141	301	2633
israel	175	142	2358
kb2	44	41	291
lotfi	154	308	1086
qap8	913	1632	8304
recipe	92	180	752
sc105	106	103	281
sc205	206	203	552
sc50a	51	48	131
sc50b	51	48	119
scagr25	472	500	2029
scagr7	130	140	553
scfxm1	331	457	2612
scorpion	389	358	1708
scrs8	491	1169	4029
scsd1	78	760	3148
sctap1	301	480	2052
share1b	118	225	1182
share2b	97	79	730
shell	537	1775	4900
stair	357	467	3857
standata	360	1075	3038
standgub	362	1184	3147
standmps	468	1075	3686
stocfor1	118	111	474
vtp_base	199	203	914

Table 7.1: Netlib LP problems used and problem sizes