Chapter 1

Introduction

Linear programming is one of the great success stories of optimization. Since its formulation in the 1930s and 1940s and the development of the simplex algorithm by Dantzig in the mid 1940s, generations of workers in economics, finance, and engineering have been trained to formulate and solve linear programming problems. Even when the situations being modeled are actually nonlinear, linear formulations are favored because the software is highly sophisticated, because the algorithms guarantee convergence to a global minimum, and because uncertainties in the model and data often make it impractical to construct a more elaborate nonlinear model.

The publication in 1984 of Karmarkar's paper [57] was probably the most significant event in linear programming since the discovery of the simplex method. The excitement that surrounded this paper was due partly to a theoretical property of Karmarkar's algorithm—polynomial complexity—and partly to the author's claims of excellent practical performance on large linear programs. These claims were never fully borne out, but the paper sparked a revolution in linear programming research that led to theoretical and computational advances on many fronts. Karmarkar's paper, and earlier work whose importance was recognized belatedly, gave rise to the field of *interior-point methods*, and the years following 1984 saw rapid development and expansion of this new field that continue even today. Theoreticians focused much of their attention on *primal-dual methods*, the elegant and powerful class of interior-point methods that is the subject of this book. Computational experiments, which took place simultaneously with the theoretical development, showed that primal-dual algorithms also performed better than other interior-point methods on practical problems, outperforming even simplex codes on many large problems. A puzzling gap remained between theory and practice, however, since the best practical algorithms differed in a number of respects from the algorithms with nice theoretical properties. Research in the 1990s largely closed this gap, and by 1994, the field had matured.

In this book, we describe the state of the art in primal-dual interior-point methods in their application to linear programming. We start by presenting some background on the duality theory of linear programming and on interior-point methods in Chapter 2. Chapter 3 sketches the theory of algorithmic complexity. The theory for the most successful algorithms in the primal-dual class—potential-reduction, path-following, and infeasible-interior-point algorithms—is presented in Chapters 4, 5, and 6, respectively. In succeeding chapters, we discuss issues of rapid asymptotic convergence (Chapter 7); extensions to more general classes of problems, such as convex quadratic programming (Chapter 8); and detection of infeasible linear programs (Chapter 9). The last two chapters turn to more immediate practical concerns. Chapter 10 discusses Mehrotra's predictor-corrector approach, which, since 1990, has been the basis for most interior-point software. Finally, in Chapter 11, we discuss the issues that arise in implementing primal-dual algorithms. Appendix A contains background information on linear algebra, optimization and complementarity theory, and numerical analysis. Some current software is reviewed in Appendix B.

The remainder of this chapter is devoted to a thumbnail sketch of each of these topics.

Linear Programming

The fundamental properties of a linear programming problem are

a. a vector of real variables, whose optimal values are found by solving the problem;

b. a linear objective function;

c. linear constraints, both inequalities and equalities.

One particular formulation of the linear programming problem—the standard form—is frequently used to describe and analyze algorithms. This form is

$$\min c^T x \text{ subject to } Ax = b,\ x \geq 0, \tag{1.1}$$

where c and x are vectors in $\mathbb{R}^n$, b is a vector in $\mathbb{R}^m$, and A is an $m \times n$ matrix. If x satisfies the constraints $Ax = b$, $x \geq 0$, we call it a *feasible point*; the set of all feasible points is the *feasible set.*

We can convert any linear program into standard form by introducing additional variables—called slack variables and artificial variables—into its formulation.

Associated with any linear program is another linear program called the *dual*, which consists of the same data objects arranged in a different way. The dual for (1.1) is

$$\max\ b^T\lambda \ \text{ subject to } A^T\lambda + s = c,\ s \geq 0, \tag{1.2}$$

where λ is a vector in $\mathbb{R}^m$ and s is a vector in $\mathbb{R}^n$. We call components of λ the *dual variables*, while s is the vector of *dual slacks.* The dual problem could be stated more compactly by eliminating s from (1.2) and rewriting the constraints as $A^T\lambda \leq c$. However, it turns out to be expedient for the analysis and implementation of interior-point methods to include s explicitly.

The linear programming problem (1.1) often is called the *primal*, to distinguish it from (1.2), and the two problems together are referred to as the *primal-dual pair.*

A *duality theory* that explains the relationship between the two problems (1.1) and (1.2) has been developed. The feasible set and the solution set for the primal tell us a lot about the dual, and vice versa. For instance, given any feasible vectors x for (1.1) and (λ, s) for (1.2), we have that

$$b^T\lambda \leq c^Tx. \tag{1.3}$$

In other words, the dual objective gives a lower bound on the primal objective, and the primal gives an upper bound on the dual. The two objective functions coincide at solutions, so that $b^T\lambda^* = c^Tx^*$ whenever x^* solves (1.1) and (λ^*, s^*) solves (1.2).

In Chapter 2, we discuss those aspects of the duality theory that have a direct bearing on the design and analysis of interior-point methods. We do not attempt a complete treatment of this fascinating topic, referring the reader instead to the standard reference texts.

Optimality conditions—the algebraic conditions that must be satisfied by solutions of linear programming problems—are derived from first principles and the duality theory in many treatments of linear programming. They also can be stated as special cases of the optimality conditions for general constrained optimization, known as the Karush–Kuhn–Tucker (or KKT) conditions. We state the KKT conditions in Appendix A, in a form that is sufficiently general for the purposes of this book (see Theorem A.1). The optimality conditions for the primal problem (1.1) are obtained by specializing Theorems A.1 and A.2:

The vector $x^* \in \mathbb{R}^n$ *is a solution of* (1.1) *if and only if there exist vectors* $s^* \in \mathbb{R}^n$ *and* $\lambda^* \in \mathbb{R}^m$ *for which the following conditions hold for* $(x, \lambda, s) = (x^*, \lambda^*, s^*)$*:*

$$\begin{aligned} A^T\lambda + s &= c, && (1.4a) \\ Ax &= b, && (1.4b) \\ x_i s_i &= 0, \quad i = 1, 2, \ldots, n, && (1.4c) \\ (x, s) &\geq 0. && (1.4d) \end{aligned}$$

In the terminology of general constrained optimization, the vectors λ and s are *Lagrange multipliers* for the constraints $Ax = b$ and $x \geq 0$, respectively. Condition (1.4c) implies that for each index $i = 1, 2, \ldots, n$, one of the components x_i or s_i must be zero. This condition is known as the *complementarity condition*, since it implies that the nonzeros of x and s appear in complementary locations.

Note that in (1.4) the Lagrange multipliers are denoted by the same symbols—λ and s—that we use for the unknowns in the dual problem (1.2). This choice is not an accident, for if we apply Theorems A.1 and A.2 to the dual problem, we find that the optimality conditions make use of exactly the same conditions (1.4). This is the crucial observation that defines the relationship between the primal and dual problems. Formally, we state the dual optimality conditions as follows:

The vector $(\lambda^*, s^*) \in \mathbb{R}^m \times \mathbb{R}^n$ *is a solution of* (1.2) *if and only if there exists a vector* $x^* \in \mathbb{R}^n$ *such that the conditions* (1.4) *hold for* $(x, \lambda, s) = (x^*, \lambda^*, s^*)$.

By examining the conditions (1.4) from both the primal and the dual viewpoints, we conclude that a vector (x^*, λ^*, s^*) solves the system (1.4) if and only if x^* solves the primal problem (1.1) and (λ^*, s^*) solves the dual problem (1.2). The vector (x^*, λ^*, s^*) is called a *primal-dual solution.*

Primal-Dual Methods

This book is about *primal-dual interior-point methods.* These methods find primal-dual solutions (x^*, λ^*, s^*) by applying variants of Newton's method to the three equality conditions in (1.4) and modifying the search directions and step lengths so that the inequalities $(x, s) \geq 0$ are satisfied *strictly* at every iteration. It is precisely the innocuous-looking bounds on x and s that give rise to all the complications in the design, analysis, and implementation of the methods described in this book.

Let us restate the optimality conditions (1.4) in a slightly different form by means of a mapping F from $\mathbb{R}^{2n+m}$ to $\mathbb{R}^{2n+m}$:

$$F(x, \lambda, s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe \end{bmatrix} = 0, \tag{1.5a}$$

$$(x, s) \geq 0, \tag{1.5b}$$

where

$$X = \operatorname{diag}(x_1, x_2, \ldots, x_n), \qquad S = \operatorname{diag}(s_1, s_2, \ldots, s_n), \tag{1.6}$$

and $e = (1, 1, \ldots, 1)^T$. Note that F is actually linear in its first two terms $Ax - b$, $A^T\lambda + s - c$, and only mildly nonlinear in the remaining term XSe.

All primal-dual methods generate iterates (x^k, λ^k, s^k) that satisfy the bounds (1.5b) strictly, that is, $x^k > 0$ and $s^k > 0$. This property is the origin of the term *interior-point.* By respecting these bounds, the methods avoid spurious solutions, which are points that satisfy $F(x, \lambda, s) = 0$ but not $(x, s) \geq 0$. Spurious solutions abound, and none of them gives any useful information about solutions of (1.1) or (1.2), so it is best to exclude them altogether from the region of search. Most interior-point methods actually require the iterates to be *strictly feasible*; that is, each (x^k, λ^k, s^k) must satisfy the linear equality constraints for the primal and dual problems. If we define the primal-dual *feasible set* $\mathcal{F}$ and *strictly feasible set* $\mathcal{F}^o$ by

$$\mathcal{F} = \{(x, \lambda, s) \mid Ax = b, A^T\lambda + s = c, (x, s) \geq 0\}, \tag{1.7a}$$

$$\mathcal{F}^o = \{(x, \lambda, s) \mid Ax = b, A^T\lambda + s = c, (x, s) > 0\}, \tag{1.7b}$$

the strict feasibility condition can be written concisely as

$$(x^k, \lambda^k, s^k) \in \mathcal{F}^o.$$

Like most iterative algorithms in optimization, primal-dual interior-point methods have two basic ingredients: a procedure for determining the step and a measure of the desirability of each point in the search space. As mentioned earlier, the search direction procedure has its origins in Newton's method for the nonlinear equations (1.5a). Newton's method forms a linear model for F around the current point and obtains the search direction $(\Delta x, \Delta\lambda, \Delta s)$ by solving the following system of linear equations:

$$J(x,\lambda,s)\begin{bmatrix}\Delta x\\ \Delta\lambda\\ \Delta s\end{bmatrix} = -F(x,\lambda,s),$$

where J is the Jacobian of F. If the current point is strictly feasible (that is, $(x,\lambda,s)\in\mathcal{F}^o$), the Newton step equations become

$$\begin{bmatrix}0 & A^T & I\\ A & 0 & 0\\ S & 0 & X\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta\lambda\\ \Delta s\end{bmatrix} = \begin{bmatrix}0\\ 0\\ -XSe\end{bmatrix}. \tag{1.8}$$

A full step along this direction usually is not permissible, since it would violate the bound $(x,s)\geq 0$. To avoid this difficulty, we perform a line search along the Newton direction so that the new iterate is

$$(x,\lambda,s)+\alpha(\Delta x,\Delta\lambda,\Delta s)$$

for some line search parameter $\alpha\in(0,1]$. Unfortunately, we often can take only a small step along the direction ($\alpha \ll 1$) before violating the condition $(x,s)>0$; hence, the pure Newton direction (1.8) often does not allow us to make much progress toward a solution.

Primal-dual methods modify the basic Newton procedure in two important ways:

1. They bias the search direction toward the interior of the nonnegative orthant $(x,s)\geq 0$ so that we can move further along the direction before one of the components of (x,s) becomes negative.

2. They keep the components of (x,s) from moving "too close" to the boundary of the nonnegative orthant. Search directions computed from points that are close to the boundary tend to be distorted, and little progress can be made along them.

We consider these modifications in turn.

The Central Path

The central path $\mathcal{C}$ is an arc of strictly feasible points that plays a vital role in the theory of primal-dual algorithms. It is parametrized by a scalar $\tau > 0$, and each point $(x_\tau, \lambda_\tau, s_\tau) \in \mathcal{C}$ solves the following system:

$$\begin{aligned} A^T\lambda + s &= c, && && \text{(1.9a)} \\ Ax &= b, && && \text{(1.9b)} \\ x_i s_i &= \tau, && i = 1, 2, \ldots, n, && \text{(1.9c)} \\ (x, s) &> 0. && && \text{(1.9d)} \end{aligned}$$

These conditions differ from the KKT conditions only in the term τ on the right-hand side of (1.9c). Instead of the complementarity condition (1.4c), we require that the pairwise products $x_i s_i$ have the same value for all indices i. From (1.9), we can define the central path as

$$\mathcal{C} = \{(x_\tau, \lambda_\tau, s_\tau) \mid \tau > 0\}.$$

Another way of defining $\mathcal{C}$ is to use the notation introduced in (1.5) and write

$$F(x_\tau, \lambda_\tau, s_\tau) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix}, \qquad (x_\tau, s_\tau) > 0. \tag{1.10}$$

We show in Chapter 2 that $(x_\tau, \lambda_\tau, s_\tau)$ is defined uniquely for each $\tau > 0$ if and only if $\mathcal{F}^o$ is nonempty. Hence, the entire path $\mathcal{C}$ is well defined.

The equations (1.9) approximate (1.4) more and more closely as τ goes to zero. If $\mathcal{C}$ converges to anything as $\tau \downarrow 0$, it must converge to a primal-dual solution of the linear program. The central path thus guides us to a solution along a route that steers clear of spurious solutions by keeping all the pairwise products $x_i s_i$ strictly positive and decreasing them to zero at the same rate.

Most primal-dual algorithms take Newton steps toward points on $\mathcal{C}$ for which $\tau > 0$, rather than pure Newton steps for F. Since these steps are biased toward the interior of the nonnegative orthant defined by $(x, s) \geq 0$, it usually is possible to take longer steps along them than along the pure Newton steps for F before violating the positivity condition. To describe the biased search direction, we introduce a *centering parameter* $\sigma \in [0, 1]$ and a *duality measure* μ defined by

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i s_i = x^T s / n, \tag{1.11}$$

which measures the average value of the pairwise products x_is_i. The generic step equations are then

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XSe + \sigma\mu e \end{bmatrix}. \qquad (1.12)$$

The step $(\Delta x, \Delta\lambda, \Delta s)$ is a Newton step toward the point $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu}) \in \mathcal{C}$, at which the pairwise products x_is_i are all equal to $\sigma\mu$. In contrast, the step (1.8) aims directly for the point at which the KKT conditions (1.4) are satisfied.

If $\sigma = 1$, the equations (1.12) define a *centering direction*, a Newton step toward the point $(x_\mu, \lambda_\mu, s_\mu) \in \mathcal{C}$, at which all the pairwise products x_is_i are identical to μ. Centering directions are usually biased strongly toward the interior of the nonnegative orthant and make little, if any, progress in reducing μ. However, by moving closer to $\mathcal{C}$, they set the scene for substantial progress on the next iteration. (Since the next iteration starts near $\mathcal{C}$, it will be able to take a relatively long step without leaving the nonnegative orthant.) At the other extreme, the value $\sigma = 0$ gives the standard Newton step (1.8), sometimes known as the *affine-scaling direction* for reasons to be discussed later. Many algorithms use intermediate values of σ from the open interval $(0, 1)$ to trade off between the twin goals of reducing μ and improving centrality.

A Primal-Dual Framework

With these basic concepts in hand, we can define Framework PD, a general framework for primal-dual algorithms. Most methods in this book are special cases of Framework PD.

Framework PD
Given $(x^0, \lambda^0, s^0) \in \mathcal{F}^o$
for $k = 0, 1, 2, \ldots$
    solve

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta\lambda^k \\ \Delta s^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -X^kS^ke + \sigma_k\mu_k e \end{bmatrix}, \qquad (1.13)$$

where $\sigma_k \in [0, 1]$ and $\mu_k = (x^k)^Ts^k/n$;

    set

```
    $(x^{k+1}, \lambda^{k+1}, s^{k+1}) \leftarrow (x^k, \lambda^k, s^k) + \alpha_k(\Delta x^k, \Delta\lambda^k, \Delta s^k),$    (1.14)
        choosing $\alpha_k$ so that $(x^{k+1}, s^{k+1}) > 0$.
end (for).
```

Path-Following Methods

A path-following algorithm explicitly restricts the iterates to a neighborhood of the central path $\mathcal{C}$ and follows $\mathcal{C}$ to a solution of the linear program. The neighborhood excludes points (x, s) that are too close to the boundary of the nonnegative orthant. Therefore, search directions calculated from any point in the neighborhood make at least minimal progress toward the solution set.

Recall that a key ingredient of any optimization algorithm is a measure of the desirability of each point in the search space. In path-following algorithms, the duality measure μ defined by (1.11) fills this role. The duality measure μ_k is forced to zero as $k \to \infty$, so the iterates (x^k, λ^k, s^k) come closer and closer to satisfying the KKT conditions (1.4).

The two most interesting neighborhoods of $\mathcal{C}$ are the so-called 2-norm neighborhood $\mathcal{N}_2(\theta)$ defined by

$$\mathcal{N}_2(\theta) = \{(x, \lambda, s) \in \mathcal{F}^o \mid \|XSe - \mu e\|_2 \leq \theta\mu\} \tag{1.15}$$

for some $\theta \in (0, 1)$, and the one-sided ∞-norm neighborhood $\mathcal{N}_{-\infty}(\gamma)$ defined by

$$\mathcal{N}_{-\infty}(\gamma) = \{(x, \lambda, s) \in \mathcal{F}^o \mid x_i s_i \geq \gamma\mu \quad \text{all } i = 1, 2, \ldots, n\} \tag{1.16}$$

for some $\gamma \in (0, 1)$. (Typical values of the parameters are $\theta = 0.5$ and $\gamma = 10^{-3}$.) If a point lies in $\mathcal{N}_{-\infty}(\gamma)$, each pairwise product $x_i s_i$ must be at least some small multiple γ of their average value μ. This requirement is actually quite modest, and we can make $\mathcal{N}_{-\infty}(\gamma)$ encompass most of the feasible region $\mathcal{F}$ by choosing γ close to zero. The $\mathcal{N}_2(\theta)$ neighborhood is more restrictive, since certain points in $\mathcal{F}^o$ do not belong to $\mathcal{N}_2(\theta)$ no matter how close θ is chosen to its upper bound of 1.

By keeping all iterates inside one or another of these neighborhoods, path-following methods reduce all the pairwise products $x_i s_i$ to zero at more or less the same rate.

Path-following methods are akin to homotopy methods for general nonlinear equations, which also define a path to be followed to the solution.

Traditional homotopy methods stay in a tight tubular neighborhood of their path, making incremental changes to the parameter and chasing the homotopy path all the way to a solution. For primal-dual methods, this neighborhood is conical rather than tubular, and it tends to be broad and loose for larger values of the duality measure μ. It narrows as $\mu \to 0$, however, because of the positivity requirement $(x, s) > 0$.

Some path-following methods choose conservative values for the centering parameter σ (that is, σ only slightly less than 1) so that unit steps can be taken along the resulting direction from (1.12) without leaving the chosen neighborhood. These methods, which are known as *short-step* path-following methods, make only slow progress toward the solution because a restrictive $\mathcal{N}_2$ neighborhood is needed to make them work.

Better results are obtained with the *predictor-corrector* method, which uses two $\mathcal{N}_2$ neighborhoods, nested one inside the other. Every second step of this method is a "predictor" step, which starts in the inner neighborhood and moves along the affine-scaling direction (computed by setting $\sigma = 0$ in (1.12)) to the boundary of the outer neighborhood. The gap between neighborhood boundaries is wide enough to allow this step to make significant progress in reducing μ. Between these predictor steps, the algorithm takes "corrector" steps (computed with $\sigma = 1$ and $\alpha = 1$), which take it back inside the inner neighborhood to prepare for the next predictor step. The predictor-corrector algorithm converges superlinearly, as we show in Chapter 7.

Long-step path-following methods make less conservative choices of σ than their short-step cousins. As a consequence, they need to perform a line search along $(\Delta x, \Delta\lambda, \Delta s)$ to avoid leaving the chosen neighborhood. By making judicious choices of σ, however, long-step methods can make much more rapid progress than short-step methods, particularly when the $\mathcal{N}_{-\infty}$ neighborhood is used.

All three methods are discussed and analyzed in Chapter 5.

Potential-Reduction Methods

Potential-reduction methods take steps of the same form as do path-following methods, but they do not explicitly follow $\mathcal{C}$ and can be motivated independently of it. They use a logarithmic potential function to measure the worth of each point in $\mathcal{F}^o$ and aim to achieve a certain fixed reduction in this function at each iteration. The primal-dual potential function, which we denote generically by Φ, usually has two important properties:

$$\Phi \quad \to \quad \infty \text{ if } x_i s_i \to 0 \text{ for some } i, \text{ but } \mu = x^T s/n \not\to 0, \qquad (1.17a)$$

$$\Phi \;\to\; -\infty \text{ if and only if } (x, \lambda, s) \to \Omega, \tag{1.17b}$$

where Ω is the set of primal-dual solutions to (1.1). The first property (1.17a) stops any one of the pairwise products $x_i s_i$ from approaching zero independently of the others and therefore keeps the iterates away from the boundary of the nonnegative orthant. The second property (1.17b) relates Φ to the solution set Ω. If our algorithm forces Φ to $-\infty$, then (1.17b) ensures that the sequence approaches the solution set.

The most interesting primal-dual potential function is the Tanabe–Todd–Ye function Φ_ρ, defined by

$$\Phi_\rho(x, s) = \rho \log x^T s - \sum_{i=1}^{n} \log x_i s_i \tag{1.18}$$

for some parameter $\rho > n$. Like all algorithms based on Framework PD, potential-reduction algorithms obtain their search directions by solving (1.13) for some $\sigma_k \in (0, 1)$. The step length α_k is chosen to approximately minimize Φ_ρ along the computed direction. In Chapter 4, we see that cookbook choices of σ_k and α_k are sufficient to guarantee constant reduction in Φ_ρ at every iteration. Hence, Φ_ρ will approach $-\infty$, forcing convergence. Smarter choices of σ_k and α_k (adaptive and heuristic) are also covered by the theory, provided that they yield at least as much reduction in Φ_ρ as do the cookbook values.

Infeasible Starting Points

So far, we have assumed that the starting point (x^0, λ^0, s^0) is strictly feasible and, in particular, that it satisfies the linear equations $Ax^0 = b$, $A^T\lambda^0 + s^0 = c$. All subsequent iterates also respect these constraints, because of the zero right-hand side terms in (1.13).

For most problems, however, a strictly feasible starting point is difficult to find. This task can always be made trivial by reformulating the problem, but the reformulation sometimes introduces distortions that can make the problem harder to solve. An alternative approach is provided by *infeasible-interior-point* methods, which require nothing more of the starting point than positivity of x^0 and s^0. The search direction needs to be modified so that it moves closer to feasibility as well as to centrality, but only a slight change is needed to the step equation (1.12). If we define the residuals for the two linear equations as

$$r_b = Ax - b, \qquad r_c = A^T\lambda + s - c, \tag{1.19}$$

the step equation becomes

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe + \sigma\mu e \end{bmatrix}. \tag{1.20}$$

The search direction is still a Newton step toward the point $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu}) \in \mathcal{C}$. It tries to bite off all the infeasibility in the equality constraints in a single step. If a full step is ever taken (that is, $\alpha = 1$), the residuals r_b and r_c become zero, and all subsequent iterates remain strictly feasible.

In Chapter 6, we give a complete description and analysis of a path-following infeasible-interior-point method, which is a variant of the long-step feasible method from Chapter 5. It confines all iterates to a central path neighborhood like $\mathcal{N}_{-\infty}(\gamma)$, extended to allow violation of the linear equality conditions. In this extended neighborhood, the residual norms $\|r_b\|$ and $\|r_c\|$ are bounded by a constant multiple of the duality measure μ. By squeezing μ to zero, we also force r_b and r_c to zero so that we approach complementarity and feasibility simultaneously.

Superlinear Convergence

We now return to the place from which we started, where we motivated primal-dual algorithms as modifications of Newton's method applied to $F(x, \lambda, s) = 0$. Because of the modifications, the search direction often contains a centering term (when $\sigma > 0$), and the step length α is often smaller than its "natural" value of 1. The modifications lead to nice global convergence properties and good practical performance, but they interfere with the best-known characteristic of Newton's method: fast asymptotic convergence. Fortunately, it is possible to design algorithms that recover this important property without sacrificing the benefits of the modified algorithm. When the iterates get close enough to the solution set, there is less danger of the pure Newton step (1.8) being attracted to spurious solutions. By relaxing some of the restrictions and modifications that are applied on earlier iterations, we can allow the algorithm to take steps more like the pure Newton step and therefore achieve a fast asymptotic rate of convergence. We do not abandon the restrictions altogether, however, since we wish to retain the excellent global convergence properties.

The superlinear convergence theory for interior-point methods has a slightly different character from the theory for nonlinear equations. For a general nonlinear system $F(z) = 0$, where the mapping $F : \mathbb{R}^N \to \mathbb{R}^N$ is continuously differentiable, the sequence $\{z^k\}$ generated by Newton's method converges superlinearly to a solution z^* if the solution is *nondegenerate*; that is, the Jacobian $J(z^*)$ is nonsingular. In the interior-point setting, we can achieve superlinear convergence even when the Jacobian of F approaches a singular limit and even when the solution is not unique! It turns out, as we show in Chapter 7, that the constraints $(x, s) \geq 0$ and our insistence on staying away from the boundary of the nonnegative orthant provide the extra structure that we need for superlinearity.

Extensions

Primal-dual methods for linear programming can be extended to wider classes of problems. Extensions to the monotone linear complementarity problem (LCP) and convex quadratic programming (QP) are straightforward. The monotone LCP—the qualifier "monotone" is implicit throughout this book—is the problem of finding vectors x and s in $\mathbb{R}^n$ that satisfy the following conditions:

$$s = Mx + q, \qquad (x, s) \geq 0, \qquad x^T s = 0, \tag{1.21}$$

where M is a positive semidefinite $n \times n$ matrix and $q \in \mathbb{R}^n$. The similarity between (1.21) and the KKT conditions (1.4) is obvious: the last two conditions in (1.21) correspond to (1.4d) and (1.4c), respectively, while the condition $s = Mx + q$ is similar to the equations (1.4a) and (1.4b).

In many situations, it is more convenient to use a formulation of the LCP that differs slightly from (1.21). In Chapter 8, we discuss two formulations that are more flexible than (1.21) but are equivalent to this basic formulation in a strong sense. The LCP deserves the attention we pay to it in Chapter 8 because it is such a versatile tool for formulating and solving a wide range of problems via primal-dual methods.

In convex QP, we minimize a convex quadratic objective subject to linear constraints. A convex quadratic generalization of the standard form linear program (1.1) is

$$\min c^T x + \tfrac{1}{2} x^T Q x \ \text{ subject to } Ax = b, \, x \geq 0, \tag{1.22}$$

where Q is a symmetric $n \times n$ positive semidefinite matrix. The KKT conditions for this problem are similar to (1.4) and (1.21). In fact, we can show that any LCP can be formulated as a convex quadratic program, and vice versa.

We also discuss extensions to a new class known as *semidefinite programming*. The name comes from the fact that some of the variables in these problems are square matrices that are constrained to be positive semidefinite. This class, which has been the topic of concentrated research since 1993, has applications in many areas, including control theory and combinatorial optimization.

The monotone nonlinear complementarity problem is obtained by replacing the first (linear) condition in (1.21) by a nonlinear monotone function. That is, we look for vectors x and s in $\mathbb{R}^n$ such that

$$s = f(x), \qquad (x, s) \geq 0, \qquad x^T s = 0,$$

where f is a smooth function with the property that $(x_1 - x_0)^T(f(x_1) - f(x_0)) \geq 0$ for all x_0 and x_1 in some domain of interest. Conceptually, the approach is not much different from the one in Framework PD. The central path is defined analogously to (1.9), and the search directions are still Newton steps toward points on this path. Handling of infeasibility becomes a more prominent issue, however, since it is no longer obvious how we can retain feasibility of the iterates (x^k, s^k) with respect to the nonlinear constraint $s^k = f(x^k)$.

In Chapter 8, we show how primal-dual methods can be extended to these more general problems. In the case of convex QP and LCP, the algorithms and their analysis are fairly straightforward generalizations of the linear programming techniques of Chapters 4, 5, and 6. On the other hand, extensions to nonconvex problems (especially nonlinear programming problems) raise many new and challenging issues that are beyond the scope of this book.

Mehrotra's Predictor-Corrector Algorithm

Most existing interior-point codes for general-purpose linear programming problems are based on Mehrotra's predictor-corrector algorithm [88]. The two key features of this algorithm are

a. addition of a *corrector step* to the search direction of Framework PD so that the algorithm more closely follows a trajectory to the solution set Ω;

b. adaptive choice of the centering parameter σ.

Mehrotra's method can be motivated by considering path-following algorithms for trajectories in $\mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$. The central path $\mathcal{C}$ (1.9) is one such trajectory. As we show in Chapter 2, $\mathcal{C}$ is a well-defined curve that terminates at the solution set Ω. A modified trajectory that leads from the current

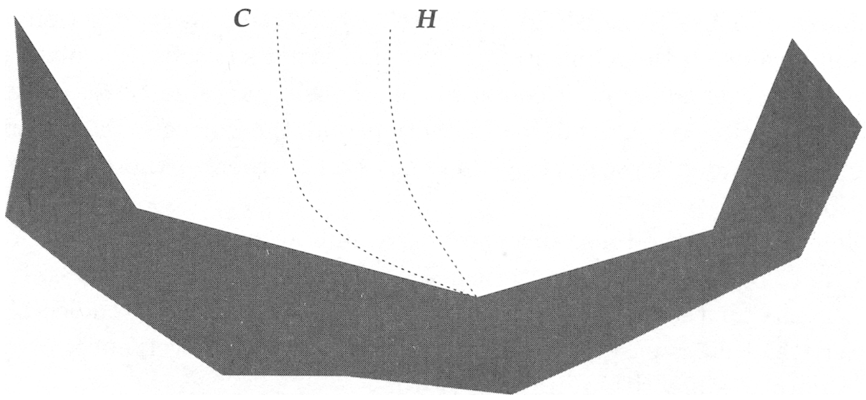

Figure 1.1. Central path $\mathcal{C}$ and a trajectory $\mathcal{H}$ from the current (noncentral) point (x, λ, s) to the solution set Ω.

point (x, λ, s) to Ω is of more immediate interest in designing algorithms, however, since the current iterate rarely lies on the central path and usually is not even feasible. These modified trajectories can be defined in a number of different ways, as we show in Chapter 10. Their common features are that they consist of points $(\hat{x}_\tau, \hat{\lambda}_\tau, \hat{s}_\tau)$ for $\tau \in [0, 1)$, with $(\hat{x}_0, \hat{\lambda}_0, \hat{s}_0) = (x, \lambda, s)$, and if the limit $\lim_{\tau \uparrow 1} (\hat{x}_\tau, \hat{\lambda}_\tau, \hat{s}_\tau)$ exists, it belongs to the solution set Ω. A trajectory $\mathcal{H}$ with these properties is depicted in Figure 1.1.

Algorithms from Framework PD can be thought of as first-order methods in that they find the tangent to a trajectory like $\mathcal{H}$ and perform a line search along it. Mehrotra's algorithm takes the next logical step of calculating the *curvature* of $\mathcal{H}$ at the current point as well as the tangent, thereby obtaining a second-order approximation to the trajectory. The curvature, which is equivalent to the corrector step mentioned above, can be obtained at relatively low cost. Computational experience shows that the extra cost is easily justified because it usually leads to a significant reduction in iteration count over methods based strictly on Framework PD.

The second important feature of Mehrotra's algorithm is that it chooses the centering parameter σ_k *adaptively*, in contrast to algorithms from Framework PD, which assign a value to σ_k prior to calculating the search direction. At each iteration, Mehrotra's algorithm first calculates the affine-scaling direction and assesses its usefulness as a search direction. If this direction yields a large reduction in μ without violating the positivity condition $(x, s) > 0$, the algorithm concludes that little centering is needed, so it chooses σ_k close to zero and calculates a centered search direction with this small value. If the affine-scaling direction is not so useful, the algorithm enforces a larger amount of centering by choosing a value of σ_k closer to 1. Computation of the centered direction and the corrector step can be combined, so adaptive centering does not add further to the cost of each iteration.

Mehrotra's method also incorporates a number of other algorithmic devices that contribute to its practical success, including the use of different step lengths for the primal and dual variables and a heuristic choice of the step lengths that is designed to speed the asymptotic convergence. We discuss Mehrotra's algorithm in Chapter 10.

Linear Algebra Issues

Most of the computational effort in implementations of primal-dual methods is taken up in solving linear systems of the forms (1.12) and (1.20). The coefficient matrix in these systems is usually large and sparse, since the constraint matrix A is itself large and sparse in most applications. The special structure in the step equations (1.12) and (1.20) allows us to reformulate them as systems with more compact symmetric coefficient matrices, which are easier and cheaper to factor than the original form.

The reformulation procedures are simple, as we show by applying them to the system (1.20). Since the current point (x, λ, s) has x and s strictly positive, the diagonal matrices X and S are nonsingular. Hence, by eliminating Δs from (1.20), we obtain the following equivalent system:

$$\begin{aligned} \begin{bmatrix} 0 & A \\ A^T & -D^{-2} \end{bmatrix} \begin{bmatrix} \Delta\lambda \\ \Delta x \end{bmatrix} &= \begin{bmatrix} -r_b \\ -r_c + s - \sigma\mu X^{-1}e \end{bmatrix}, && \text{(1.23a)} \\ \Delta s &= -s + \sigma\mu X^{-1}e - X^{-1}S\Delta x, && \text{(1.23b)} \end{aligned}$$

where we have introduced the notation

$$D = S^{-1/2}X^{1/2}. \qquad \text{(1.24)}$$

This form of the step equations usually is known as the *augmented system.* Since the matrix $X^{-1}S$ is also diagonal and nonsingular, we can go a step further, eliminating Δx from (1.23a) to obtain another equivalent form:

$$\begin{aligned} AD^2A^T\Delta\lambda &= -r_b + A(-S^{-1}Xr_c + x - \sigma\mu S^{-1}e), && \text{(1.25a)} \\ \Delta s &= -r_c - A^T\Delta\lambda, && \text{(1.25b)} \\ \Delta x &= -x + \sigma\mu S^{-1}e - S^{-1}X\Delta s. && \text{(1.25c)} \end{aligned}$$

This form often is called the *normal equations* form because, in the case of $r_b = 0$, the system (1.25a) reduces to the normal equations for a linear least squares problem with coefficient matrix DA^T.

Most implementations are based on the normal equations form; they apply a direct sparse Cholesky algorithm to factor the matrix AD^2A^T. General-purpose sparse Cholesky software can be used to perform this operation; for example, two of the codes that we discuss in Appendix B make use of the sparse Cholesky code of Ng and Peyton [103]. A few modifications are needed to standard Cholesky codes, however, because AD^2A^T may be ill conditioned or singular. Ill conditioning is often observed during the final stages of a primal-dual algorithm, when the elements of the diagonal weighting matrix D^2 take on both huge and tiny values.

It becomes inefficient to formulate and solve the system (1.25a) when $AS^{-1}XA^T$ is much denser than A. This situation arises when A has one or more dense columns, and it is not uncommon in practice. We can modify the normal equations strategy by excluding the dense columns from the matrix-matrix product $AS^{-1}XA^T$ and correcting for this omission in a subsequent calculation. Details are given in Chapter 11.

The formulation (1.23) has received less attention than (1.25), mainly because algorithms and software for factoring sparse symmetric indefinite matrices are more complicated, slower, and less readily available than sparse Cholesky algorithms. This situation is changing, however, as the result of an increasing recognition that symmetric indefinite matrices are important in many contexts besides (1.23a) (see, for instance, Vavasis [140]). The formulation (1.23) is cleaner and more flexible than (1.25) in a number of respects: The difficulty caused by dense columns in A does not arise, and free variables (that is, components of x with no explicit lower or upper bounds) can be handled without resorting to the various artificial devices needed in the normal equations form.

More details about solving both formulations (1.23) and (1.25) and about other issues associated with primal-dual implementations can be found in Chapter 11.

Karmarkar's Algorithm

For many nonspecialists, the name Karmarkar is more or less synonymous with the whole field of interior-point methods. Hence, a few words about Karmarkar's method [57] and its place in history are in order.

Although the practical efficiency of the simplex method was appreciated from the time it was discovered, theoreticians remained uneasy because the method is not guaranteed to perform well on every linear program. A famous example due to Klee and Minty [61] requires 2^n simplex iterations to solve a linear program in $\mathbb{R}^n$. By contrast, other computational problems such as sorting and searching were known to be solvable by polynomial algorithms—algorithms whose run time is, at worst, a polynomial function of the amount of storage needed for the problem data.

The first polynomial algorithm for linear programming, Khachiyan's ellipsoid algorithm [60], was a computational disappointment. Its convergence is much too slow, it is not robust in the presence of rounding errors, and it requires a large, dense $n \times n$ matrix to be stored and updated at every iteration. It is not competitive with the simplex method on any but the smallest problems.

Karmarkar's algorithm was announced in 1984, five years after the ellipsoid method. It too is polynomial, requiring $O(n \log 1/\epsilon)$ iterations to identify a primal feasible point x such that $c^T x$ is within ϵ of its optimal value for (1.1). Karmarkar's is a *primal* algorithm; that is, it is described, motivated, and implemented purely in terms of the primal problem (1.1) without reference to the dual. At each iteration, Karmarkar's algorithm performs a projective transformation on the primal feasible set that maps the current iterate x^k to the center of the set and takes a step in the feasible steepest descent direction for the transformed space. Progress toward optimality is measured by a logarithmic potential function, as discussed in Chapter 2. Nice descriptions of the algorithm can be found in Karmarkar's original paper [57] and in Fletcher [30].

Karmarkar's method falls outside the scope of this book, and, in any case, its practical performance does not appear to match the most efficient primal-dual methods. The algorithms we discuss in Chapters 4, 5, and 6 are all polynomial, like Karmarkar's method. Further historical information on the development of interior-point methods is given in Chapter 2.

Exercises

1. Explain why we cannot have $\mu_k = 0$ for an iterate $(x^k, \lambda^k, s^k) \in \mathcal{F}^o$ from Framework PD.

2. (i) Show that $\mathcal{N}_2(\theta_1) \subset \mathcal{N}_2(\theta_2)$ when $0 \le \theta_1 < \theta_2 < 1$ and that $\mathcal{N}_{-\infty}(\gamma_1) \subset \mathcal{N}_{-\infty}(\gamma_2)$ for $0 < \gamma_2 \le \gamma_1 \le 1$.

   (ii) Show that $\mathcal{N}_2(\theta) \subset \mathcal{N}_{-\infty}(\gamma)$ if $\gamma \le 1 - \theta$.

3. Why is the neighborhood $\mathcal{N}_{-\infty}$ called the "one-sided ∞-norm neighborhood"?

4. Given an arbitrary point $(x, \lambda, s) \in \mathcal{F}^o$, find the range of γ values for which $(x, \lambda, s) \in \mathcal{N}_{-\infty}(\gamma)$. (The range depends on x and s.)

5. For $n = 2$, find a point $(x, s) > 0$ for which the condition
$$\|XSe - \mu e\|_2 \leq \theta\mu$$
is *not* satisfied for any $\theta \in (0, 1)$.

6. Show that Φ_ρ defined by (1.18) has the property (1.17a).

7. Write down the KKT conditions for the convex quadratic program (1.22).

8. By eliminating the vector s from (1.21), express the LCP as a convex quadratic program.

9. Use the fact that (x, s) is strictly positive to show that the coefficient matrix in (1.25a) is symmetric positive semidefinite. Show that this matrix is positive definite if A has full row rank.