1

Models and Algorithms: 
An Overview

by

Dimitri P. Bertsekas

6.253 Notes, Spring 2014

MIT

Contents

1.1. Lagrange Duality . . . . . . . . . . . . . . . . . . . p. 3
  1.1.1. Separable Problems - Decomposition . . . . . . . p. 8
  1.1.2. Partitioning . . . . . . . . . . . . . . . . . . p. 9
1.2. Fenchel Duality and Conic Programming . . . . . . . . p. 10
  1.2.1. Linear Conic Problems . . . . . . . . . . . . . p. 14
  1.2.2. Second Order Cone Programming . . . . . . . . p. 17
  1.2.3. Semidefinite Programming . . . . . . . . . . . . p. 22
1.3. Additive Cost Problems . . . . . . . . . . . . . . p. 25
1.4. Large Number of Constraints . . . . . . . . . . . . p. 31
1.5. Exact Penalty Functions . . . . . . . . . . . . . . p. 35
1.6. Iterative Descent Algorithms . . . . . . . . . . . . p. 43
  1.6.1. Cost Function Descent - Gradient Projection . . p. 45
  1.6.2. Alternative Descent Methods . . . . . . . . . . p. 56
  1.6.3. Incremental Algorithms . . . . . . . . . . . . . p. 58
1.7. Approximation Methods . . . . . . . . . . . . . . p. 76
  1.7.1. Polyhedral Approximation . . . . . . . . . . . . p. 77
  1.7.2. Penalty, Proximal, and Interior Point Methods . . p. 78
  1.7.3. Proximal Algorithm and Tikhonov Regularization . p. 79
  1.7.4. Smoothing of Nondifferentiable Problems . . . . . p. 80
1.8. Notes and Sources . . . . . . . . . . . . . . . . . p. 85

DISCLAIMER: This chapter (1st chapter of the book “Convex Optimization Algorithms,” currently being written) is still in draft form and is provided for the needs of the MIT 6.253 class. It should not be quoted and/or reproduced beyond the needs of the class. The author appreciates comments, typo reports, and other suggestions, at dimitrib@mit.edu.
In this chapter we provide a broad overview of some broad classes of convex optimization problems, and associated algorithms. The choice of algorithm for a given problem often depends crucially on the structure of the problem. It is therefore important to delineate the major problem structures and understand the characteristics that make them suitable for various types of algorithms.

Our primary focus will be on large challenging problems, often connected in some way to duality. We will primarily consider two types of duality. The first is Lagrange duality for constrained optimization, which arises by assigning dual variables to inequality constraints. The second is Fenchel duality together with its special case, conic duality, which involves a cost function that is the sum of convex function components. Both of these duality structures arise often in applications, and in Sections 1.1 and 1.2 we provide an overview and discuss some examples.

In Sections 1.3 and 1.4, we discuss some additional structures involving a large number of additive terms in the cost, or a large number of constraints. These types of problems also arise often in the context of duality, as well in other contexts such as machine learning with large amounts of data. In Section 1.5, we discuss the exact penalty function technique, whereby we can transform convex constrained optimization problems to equivalent unconstrained (or less constrained) ones.

The main subject of this book is algorithms for convex optimization, their underlying ideas, and their performance characteristics. We are generally interested in conditions that guarantee convergence to desired solutions, and the associated rate of convergence and complexity analysis. We are also interested in special characteristics of algorithms that make them suitable for particular types of large scale problem structures, and distributed asynchronous computation.

Algorithms for minimizing a convex function $f : \mathbb{R}^n \mapsto \mathbb{R}$ over a convex set $X$ generate a sequence $\{x_k\} \subset X$ that aims to converge to an optimal solution. Most of these algorithms involve one or both of the following two ideas:

(a) **Iterative descent**, whereby the generated sequence $\{x_k\}$ satisfies

$$\phi(x_{k+1}) < \phi(x_k) \quad \text{if and only if} \quad x_k \text{ is not optimal},$$

where $\phi$ is a merit function, that measures the progress of the algorithm towards optimality, and is minimized only at optimal points, i.e.,

$$\arg\min_{x \in X} \phi(x) = \arg\min_{x \in X} f(x).$$

Examples are $\phi(x) = f(x)$ and $\phi(x) = \inf_{x^* \in X^*} \|x - x^*\|$, where $X^*$ is the set of optimal points, assumed nonempty. In some cases, iterative descent may be the primary idea, but modifications or approximations are introduced for a variety of possible reasons. For example one may
modify an iterative descent method to make it suitable for distributed asynchronous computation or computation in the presence of random or nonrandom errors, but in the process lose the iterative descent property. In this case, the analysis of the original descent algorithm should be appropriately modified. We will summarize the ideas of iterative algorithms under various computational conditions in Section 1.6.

(b) Approximation, whereby the generated sequence \( \{x_k\} \) is obtained by solving at each \( k \) an approximation to the original optimization problem, i.e.,

\[
x_{k+1} \in \arg\min_{x \in X_k} F_k(x),
\]

where \( F_k \) is a function that approximates \( f \) and \( X_k \) is a set that approximates \( X \). These may depend on the prior iterates \( x_0, \ldots, x_k \), as well as other parameters. Key ideas here are that minimization of \( F_k \) over \( X_k \) should be easier than minimization of \( f \) over \( X \), and that \( x_k \) should be a good starting point for obtaining \( x_{k+1} \) via some (possibly special purpose) method. Of course, the approximation of \( f \) by \( F_k \) and/or \( X \) by \( X_k \) should improve as \( k \) increases, and there should be some convergence guarantees as \( k \to \infty \). We will summarize the main approximation ideas of this book in Section 1.7.

A major class of problems that we aim to solve is dual problems, which by their nature involve convex nondifferentiable minimization. The fundamental reason is that the negative of a dual function is typically a conjugate function, which is generically closed and convex, but often nondifferentiable. Accordingly most of the algorithms that we discuss in this book do not require differentiability for their application. In Section 1.6, we summarize some of the main ideas of algorithms that rely on differentiability, such as gradient and Newton methods. In Chapter 2, we discuss descent-type iterative algorithms that use subgradients. In Chapters 3 and 4, we discuss primarily the approximation approach, focusing on two types of algorithms and their combinations: polyhedral approximation and proximal. In Chapter 5, we discuss a number of additional algorithms, which extend the ideas of the preceding chapters.

Since the present chapter has an overview character, our discussion will not be supplemented by complete proofs; in many cases we will provide just intuitive explanations and refer to the literature for a more detailed analysis. Our analysis will be far more detailed in subsequent chapters.

1.1 LAGRANGE DUALITY

We start our overview of Lagrange duality with the basic case of nonlinear inequality constraints, and then consider extensions involving linear
inequality and equality constraints. Consider the problem

$$\text{minimize} \quad f(x)$$

subject to $x \in X, \quad g(x) \leq 0, \quad (1.1)$

where $X$ is a nonempty set, $g(x) = (g_1(x), \ldots, g_r(x))^t$, and $f : X \rightarrow \mathbb{R}$ and $g_j : X \rightarrow \mathbb{R}, \ j = 1, \ldots, r$, are given functions. We refer to this as the primal problem, and we denote its optimal value by $f^*$. A vector $x$ satisfying the constraints of the problem is referred to as feasible.

The dual of problem (1.1) is given by

$$\text{maximize} \quad q(\mu)$$

subject to $\mu \in \mathbb{R}^r, \quad (1.2)$

where the dual function $q$ is

$$q(\mu) = \begin{cases} 
\inf_{x \in X} L(x, \mu) & \text{if } \mu \geq 0, \\
-\infty & \text{otherwise}, 
\end{cases} \quad (1.3)$$

and $L$ is the Lagrangian function defined by

$$L(x, \mu) = f(x) + \mu^t g(x), \quad x \in X, \ \mu \in \mathbb{R}^r;$$

(cf. Section 5.3 of Appendix A). The dual optimal value is

$$q^* = \sup_{\mu \in \mathbb{R}^r} q(\mu).$$

The weak duality relation $q^* \leq f^*$ is easily shown by writing for all $\mu \geq 0$, and $x \in X$ with $g(x) \leq 0$,

$$q(\mu) \leq \inf_{z \in X} L(z, \mu) \leq f(x) + \sum_{j=1}^r \mu_j g_j(x) \leq f(x),$$

so

$$q^* = \sup_{\mu \in \mathbb{R}^r} q(\mu) = \sup_{\mu \geq 0} q(\mu) \leq \inf_{x \in X, g(x) \leq 0} f(x) = f^*.$$

We state this formally as follows.

**Proposition 1.1.1: (Weak Duality Theorem)** For any feasible solution $x$ and any $\mu$, we have $q(\mu) \leq f(x)$. Moreover, $q^* \leq f^*$. 
Generally the analytical and algorithmic solution process is simplified when strong duality ($q^* = f^*$) holds. This typically requires convexity assumptions, as exemplified by the following result, given in Prop. 5.3.1/App. B.†

**Proposition 1.1.2:** (Strong Duality - Existence of Dual Optimal Solutions) Consider the problem (1.1) under the assumption that the set $X$ is convex, and the functions $f$, and $g_1, \ldots, g_r$ are convex. Assume further that $f^*$ is finite, and that one of the following two conditions holds:

1. There exists $\pi \in X$ such that $g_j(\pi) < 0$ for all $j = 1, \ldots, r$.
2. The functions $g_j$, $j = 1, \ldots, r$, are affine, and there exists $\pi \in \text{ri}(X)$ such that $g_j(\pi) \leq 0$.

Then $q^* = f^*$ and there exists at least one dual optimal solution. Under condition (1) the set of dual optimal solutions is also compact.

The following proposition gives necessary and sufficient conditions for strong duality, and primal and dual optimality (see Prop. 5.3.2/App. B).

**Proposition 1.1.3:** (Optimality Conditions) Consider the problem (1.1). There holds $q^* = f^*$, and $(x^*, \mu^*)$ are a primal and dual optimal solution pair if and only if $x^*$ is feasible, $\mu^* \geq 0$, and

$$x^* \in \arg\min_{x \in X} L(x, \mu^*), \quad \mu^* g_j(x^*) = 0, \quad j = 1, \ldots, r. \quad (1.4)$$

**Linear Equality Constraints**

Let us consider an extension of problem (1.1) where there are additional

† Appendix B contains a full list of definitions and propositions (without proofs) relating to nonalgorithmic aspects of convex optimization. This list reflects and summarizes the content of the author’s “Convex Optimization Theory” book [Ber09]. The proposition numbers have been preserved from [Ber09], so all propositions in Appendix B can be readily accessed with proofs from [Ber09]. We will generally refer to Appendix B for material that will be used in the present book without proof. On a few occasions we will also refer to other sources, such as the author’s nonlinear programming text [Ber99].
linear equality constraints:

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in X, \ g(x) \leq 0, \ Ax = b,
\end{align*}
\] (1.5)

where \( X \) is a set, \( g(x) = (g_1(x), \ldots, g_r(x))^\prime \), \( f : X \rightarrow \mathbb{R} \) and \( g_j : X \rightarrow \mathbb{R} \), \( j = 1, \ldots, r \), are given functions, \( A \) is an \( m \times n \) matrix, and \( b \in \mathbb{R}^m \). We can deal with this problem by converting the constraint \( Ax = b \) to the equivalent set of linear inequality constraints

\[
Ax \leq b, \quad -Ax \leq -b,
\] (1.6)

with corresponding dual variables \( \lambda^+ \geq 0 \) and \( \lambda^- \geq 0 \). The Lagrangian function is

\[
f(x) + \mu^\prime g(x) + (\lambda^+ - \lambda^-)'(Ax - b),
\]

and by introducing a dual variable

\[
\lambda = \lambda^+ - \lambda^-
\] (1.7)

with no sign restriction, it can be written as

\[
L(x, \mu, \lambda) = f(x) + \mu^\prime g(x) + \lambda'(Ax - b).
\]

The dual problem is

\[
\begin{align*}
\text{maximize} & \quad \inf_{x \in X} L(x, \mu, \lambda) \\
\text{subject to} & \quad \mu \geq 0, \ \lambda \in \mathbb{R}^m.
\end{align*}
\]

In this manner, Prop. 1.1.2 under condition (2), together with Prop. 1.1.3, yield the following.

**Proposition 1.1.4: (Convex Programming - Linear Equality and Inequality Constraints)** Consider problem (1.5).

(a) Assume that \( f^* \) is finite, that the functions \( g_j \) are affine, and that there exists \( \overline{x} \in \text{ri}(X) \) such that \( A\overline{x} = b \) and \( g(\overline{x}) \leq 0 \). Then \( q^* = f^* \) and there exists at least one dual optimal solution.

(b) There holds \( f^* = q^* \), and \( (x^*, \mu^*, \lambda^*) \) are a primal and dual optimal solution pair if and only if \( x^* \) is feasible, \( \mu^* \geq 0 \), and

\[
x^* \in \arg \min_{x \in X} L(x, \mu^*, \lambda^*), \quad \mu^*_j g_j(x^*) = 0, \quad j = 1, \ldots, r.
\]
In the special case of a problem with just linear equality constraints:

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in X, \quad Ax = b,
\end{align*}
\]  

(1.8)

the Lagrangian function is

\[
L(x, \lambda) = f(x) + \lambda' (Ax - b),
\]

and the dual problem is

\[
\begin{align*}
\text{maximize} & \quad \inf_{x \in X} L(x, \lambda) \\
\text{subject to} & \quad \lambda \in \mathbb{R}^m.
\end{align*}
\]

The corresponding result, a simpler special case of Prop. 1.1.4, is given in the following proposition.

**Proposition 1.1.5: (Convex Programming - Linear Equality Constraints)** Consider problem (1.8).

(a) Assume that \( f^* \) is finite and that there exists \( \bar{x} \in \text{ri}(X) \) such that \( A\bar{x} = b \). Then \( f^* = q^* \) and there exists at least one dual optimal solution.

(b) There holds \( f^* = q^* \), and \((x^*, \lambda^*)\) are a primal and dual optimal solution pair if and only if \( x^* \) is feasible and

\[
x^* \in \arg \min_{x \in X} L(x, \lambda^*).
\]  

(1.9)

**Discrete Optimization and Lower Bounds**

The preceding propositions deal with situations where strong duality holds \((q^* = f^*)\). However, duality can be useful even when there is duality gap, as often occurs in problems that have a finite constraint set \( X \). An example is integer programming, where the components of \( x \) must be integers from a bounded range (usually 0 or 1). An important special case is the linear 0-1 integer programming problem

\[
\begin{align*}
\text{minimize} & \quad c'x \\
\text{subject to} & \quad Ax \leq b, \quad x_i = 0 \text{ or } 1, \quad i = 1, \ldots, n,
\end{align*}
\]

where \( x = (x_1, \ldots, x_n) \).
A principal approach for solving such problems is the \textit{branch-and-bound method}, which is described in many sources. This method relies on obtaining lower bounds to the optimal cost of restricted problems of the form

\begin{equation}
\text{minimize } f(x) \\
\text{subject to } x \in \hat{X}, \quad g(x) \leq 0,
\end{equation}

where $\hat{X}$ is a subset of $X$; for example in the 0-1 integer case where $X$ specifies that all $x_i$ should be 0 or 1, $\hat{X}$ may be the set of all 0-1 vectors $x$ such that one or more components $x_i$ are fixed at either 0 or 1 (i.e., are restricted to satisfy $x_i = 0$ for all $x \in \hat{X}$ or $x_i = 1$ for all $x \in \hat{X}$). These lower bounds can often be obtained by finding a dual-feasible (possibly dual-optimal) solution $\mu \geq 0$ of this problem and the corresponding dual value

\begin{equation}
q(\mu) = \inf_{x \in \hat{X}} \left\{ f(x) + \mu'g(x) \right\},
\end{equation}

which by weak duality, is a lower bound to the optimal value of the restricted problem (1.10). In a strengthened version of this approach, the given inequality constraints $g(x) \leq 0$ may be augmented by additional inequalities that are known to be satisfied by optimal solutions of the original problem.

An important point here is that when $\hat{X}$ is finite, the dual function $q$ is concave and polyhedral. Thus solving the dual problem amounts to minimizing the polyhedral function $-q$ over the nonnegative orthant. This is a major context within which polyhedral functions arise in convex optimization.

\subsection*{1.1.1 Separable Problems - Decomposition}

Let us now discuss an important problem structure that involves Lagrange duality, and arises frequently in applications. Consider the problem

\begin{equation}
\text{minimize } \sum_{i=1}^{n} f_i(x_i) \\
\text{subject to } a_j'x \leq b_j, \quad j = 1, \ldots, r,
\end{equation}

Bounds on the cost function can be used to exclude from consideration portions of the feasible set. To illustrate, consider minimizing $F(x)$ over $x \in X$, and let $Y_1, Y_2$ be two subsets of $X$. Suppose that we have bounds

$E_1 \leq \min_{x \in Y_1} f(x), \quad T_2 \geq \min_{x \in Y_2} f(x).$

Then, if $T_2 \leq E_1$, the solutions in $Y_1$ may be disregarded since their cost cannot be smaller than the cost of the best solution in $Y_2$. The lower bound $E_1$ can be obtained by using weak duality, while for the upper bound $T_2$, a value $f(x)$, where $x \in Y_2$, may be used.
where \( x = (x_1, \ldots, x_n) \), each \( f_i : \mathbb{R} \rightarrow \mathbb{R} \) is a convex function of the single scalar component \( x_i \), and \( a_j \) and \( b_j \) are some vectors and scalars, respectively. By assigning a dual variable \( \mu_j \) to the constraint \( a_j^T x \leq b_j \), we obtain the dual problem [cf. Eq. (1.2)]

\[
\begin{align*}
\text{maximize} & \quad \sum_{i=1}^{n} q_i(\mu) - \sum_{j=1}^{r} \mu_j b_j \\
\text{subject to} & \quad \mu \geq 0,
\end{align*}
\]

where

\[
q_i(\mu) = \inf_{x_i \in \mathbb{R}} \left\{ f_i(x_i) + x_i \sum_{j=1}^{r} \mu_j a_{ji} \right\},
\]

and \( \mu = (\mu_1, \ldots, \mu_r) \).

Note that the minimization involved in the calculation of the dual function has been decomposed into \( n \) simpler minimizations. These minimizations are often conveniently done either analytically or computationally, in which case the dual function can be easily evaluated. This is the key advantageous structure of separable problems: it facilitates computation of dual function values (as well as subgradients as we will see in Section 2.1), and it is amenable to decomposition/distributed computation.

There are also other separable problems that are more general than the one of Eq. (1.12). An example is when \( x \) has \( m \) components \( x_1, \ldots, x_m \) of dimensions \( n_1, \ldots, n_m \), respectively, and the problem has the form

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m} f_i(x_i) \\
\text{subject to} & \quad \sum_{i=1}^{m} g_i(x_i) \leq 0, \quad x_i \in X_i, \ i = 1, \ldots, m,
\end{align*}
\]

where \( f_i : \mathbb{R}^{n_i} \rightarrow \mathbb{R} \) and \( g_i : \mathbb{R}^{n_i} \rightarrow \mathbb{R}^r \) are given functions, and \( X_i \) are given subsets of \( \mathbb{R}^{n_i} \). The advantage of convenient computation of the dual function value using decomposition extends to such problems as well. We may also note that when the components \( x_i \) are one-dimensional, and the functions \( f_i \) and sets \( X_i \) are convex, there is a particularly favorable strong duality result for the separable problem (1.14), even when the constraint functions \( g_i \) are nonlinear but consist of convex components \( g_{ij} : \mathbb{R} \rightarrow \mathbb{R}, \ j = 1, \ldots, r \); see [Tse09].

1.1.2 Partitioning

It is important to note that there are several different ways to introduce duality in the solution of large-scale optimization problems. For example a
strategy, often called partitioning, is to divide the variables in two subsets, and minimize first with respect to one subset while taking advantage of whatever simplification may arise by fixing the variables in the other subset.

In particular, the problem

\[
\text{minimize } F(x) + G(y) \\
\text{subject to } Ax + By = c, \quad x \in X, \quad y \in Y,
\]

can be written as

\[
\text{minimize } F(x) + \inf_{By=c-Ax, y \in Y} G(y) \\
\text{subject to } x \in X,
\]
or

\[
\text{minimize } F(x) + p(c - Ax) \\
\text{subject to } x \in X,
\]

where \( p \) is the primal function of the minimization problem involving \( y \) above:

\[
p(u) = \inf_{By=u, y \in Y} G(y);
\]

(cf. Section 4.2.3 of Appendix B).

### 1.2 Fenchel Duality and Conic Programming

Let us consider the Fenchel duality framework (see Section 5.3.5 of Appendix B). It involves the problem

\[
\text{minimize } f_1(x) + f_2(Ax) \\
\text{subject to } x \in \mathbb{R}^n,
\]

where \( A \) is an \( m \times n \) matrix, \( f_1 : \mathbb{R}^n \mapsto (-\infty, \infty] \) and \( f_2 : \mathbb{R}^m \mapsto (-\infty, \infty] \) are closed convex functions, and we assume that there exists a feasible solution, i.e., an \( x \in \mathbb{R}^n \) such that \( x \in \text{dom}(f_1) \) and \( Ax \in \text{dom}(f_2) \).

The problem is equivalent to the following constrained optimization problem in the variables \( x_1 \in \mathbb{R}^n \) and \( x_2 \in \mathbb{R}^m \):

\[
\text{minimize } f_1(x_1) + f_2(x_2) \\
\text{subject to } x_1 \in \text{dom}(f_1), \quad x_2 \in \text{dom}(f_2), \quad x_2 = Ax_1.
\]

Viewing this as a convex programming problem with the linear equality constraint \( x_2 = Ax_1 \), we obtain the dual function as

\[
q(\lambda) = \inf_{x_1 \in \text{dom}(f_1), \ x_2 \in \text{dom}(f_2)} \left\{ f_1(x_1) + f_2(x_2) + \lambda' (x_2 - Ax_1) \right\} \\
= \inf_{x_1 \in \mathbb{R}^n} \left\{ f_1(x_1) - \lambda' Ax_1 \right\} + \inf_{x_2 \in \mathbb{R}^m} \left\{ f_2(x_2) + \lambda' x_2 \right\}.
\]
The dual problem of maximizing $q$ over $\lambda \in \mathbb{R}^m$, after a sign change to convert it to a minimization problem, takes the form

$$
\begin{align*}
\text{minimize} & \quad f^*(A'\lambda) + f^*_2(-\lambda) \\
\text{subject to} & \quad \lambda \in \mathbb{R}^m,
\end{align*}
$$

where $f^*_1$ and $f^*_2$ are the conjugate functions of $f_1$ and $f_2$. We denote by $f^*$ and $q^*$ the corresponding optimal primal and dual values.

We have the following proposition, which is given as Prop. 5.3.8/App. B. Parts (a) and (b) are obtained by applying Prop. 1.1.5(a) to problem (1.16), viewed as a problem with $x_2 = Ax_1$ as the only linear equality constraint. The first equation of part (c) is a consequence of Prop. 1.1.5(b), and its equivalence with the last two equations is a consequence of the conjugate subgradient theorem (Prop. 5.4.3, App. B).

**Proposition 1.2.1: (Fenchel Duality)**

(a) If $f^*$ is finite and $(A \cdot \text{ri}(\text{dom}(f_1))) \cap \text{ri}(\text{dom}(f_2)) \neq \emptyset$, then $f^* = q^*$ and there exists at least one dual optimal solution.

(b) If $q^*$ is finite and $\text{ri}(\text{dom}(f^*_1)) \cap (A' \cdot \text{ri}(\text{dom}(f^*_2))) \neq \emptyset$, then $f^* = q^*$ and there exists at least one primal optimal solution.

(c) There holds $f^* = q^*$, and $(x^*, \lambda^*)$ is a primal and dual optimal solution pair if and only if any one of the following equivalent conditions hold

$$
\begin{align*}
x^* & \in \arg \min_{x \in \mathbb{R}^n} \{ f_1(x) - x' A' \lambda^* \} \quad \text{and} \quad Ax^* \in \arg \min_{z \in \mathbb{R}^m} \{ f_2(z) + z' \lambda^* \}, \\
A' \lambda^* & \in \partial f_1(x^*) \quad \text{and} \quad \lambda^* \in -\partial f_2(Ax^*), \\
x^* & \in \partial f^*_1(A' \lambda^*) \quad \text{and} \quad Ax^* \in \partial f^*_2(-\lambda^*).
\end{align*}
$$

**Minimax Problems**

Minimax problems involve minimization over a set $X$ of a function $\overline{F}$ of the form

$$
\overline{F}(x) = \sup_{z \in Z} \phi(x, z),
$$

where $X$ and $Z$ are subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively, and $\phi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is a given function. Some (but not all) problems of this type are connected with constrained optimization, and some are connected with Fenchel duality.
In particular, let $\phi$ and $Z$ have the form

$$
\phi(x,z) = f(x) + z'g(x), \quad Z = \{z \mid z \geq 0\},
$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ are given functions. Then it is seen that

$$
F(x) = \begin{cases} 
  f(x) & \text{if } g(x) \leq 0, \\
  \infty & \text{otherwise.}
\end{cases}
$$

Thus minimization of $F$ over $x \in X$ is equivalent to solving the constrained optimization problem (1.1):

$$
\begin{aligned}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in X, \quad g(x) \leq 0.
\end{aligned} \quad (1.21)
$$

The dual problem is to maximize over $z \geq 0$ the function

$$
F^*(z) = \inf_{x \in X} \phi(x,z),
$$

and the minimax equality

$$
\inf_{x \in X} \sup_{z \in Z} \phi(x,z) = \sup_{z \in Z} \inf_{x \in X} \phi(x,z) \quad (1.22)
$$
is equivalent to problem (1.21) having no duality gap.

Alternatively, let $\phi$ have the form

$$
\phi(x,z) = f(x) + z'Ax - g(z),
$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $g : \mathbb{R}^m \mapsto \mathbb{R}$ are given functions, and $A$ is a given $m \times n$ matrix. Then we have

$$
\begin{aligned}
F^*(x) = \sup_{z \in Z} \phi(x,z) &= f(x) + \sup_{z \in Z} \left\{ (Ax)'z - g(z) \right\} \\
&= f(x) + \hat{g}^*(Ax),
\end{aligned}
$$

where $\hat{g}^*$ is the conjugate of the function

$$
\hat{g}(z) = \begin{cases} 
  g(z) & \text{if } z \in Z, \\
  \infty & \text{otherwise.}
\end{cases}
$$

Thus the minimax problem of minimizing $F^*$ over $x \in X$ comes under the Fenchel framework (1.15) with $f_2 = \hat{g}^*$ and $f_1$ given by

$$
f_1(x) = \begin{cases} 
  f(x) & \text{if } x \in X, \\
  \infty & \text{if } x \notin X.
\end{cases}
$$

It can also be verified that the Fenchel dual problem (1.17) is equivalent to maximizing over $z \in Z$ the function $F^*(z) = \inf_{z \in X} \phi(x,z)$. Again having no duality gap is equivalent to the minimax equality (1.22) holding.
**Conic Programming**

An important problem structure, which can be analyzed as a special case of the Fenchel duality framework is *conic programming*. This is the problem

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in C,
\end{align*}
\]

where \( f : \mathbb{R}^n \rightarrow (-\infty, \infty] \) is a proper convex function and \( C \) is a convex cone in \( \mathbb{R}^n \).

Indeed, let us apply Fenchel duality with \( A \) equal to the identity and the definitions

\[
\begin{align*}
f_1(x) &= f(x), \\
\text{and} \quad f_2(x) &= \begin{cases} 
0 & \text{if } x \in C, \\
\infty & \text{if } x \notin C.
\end{cases}
\end{align*}
\]

The corresponding conjugates are

\[
\begin{align*}
f_1^\star(\lambda) &= \sup_{x \in \mathbb{R}^n} \{ \lambda' x - f(x) \}, \\
f_2^\star(\lambda) &= \sup_{x \in C} \lambda' x = \begin{cases} 
0 & \text{if } \lambda \in C^*, \\
\infty & \text{if } \lambda \notin C^*,
\end{cases}
\end{align*}
\]

where

\[
C^* = \{ \lambda \mid \lambda' x \leq 0, \forall x \in C \}
\]

is the polar cone of \( C \) (note that \( f_2^\star \) is the support function of \( C \); cf. Section 1.6 of Appendix B). The dual problem is

\[
\begin{align*}
\text{minimize} & \quad f^\star(\lambda) \\
\text{subject to} & \quad \lambda \in \hat{C},
\end{align*}
\]

where \( f^\star \) is the conjugate of \( f \) and \( \hat{C} \) is the negative polar cone (also called the *dual cone* of \( C \):

\[
\hat{C} = -C^* = \{ \lambda \mid \lambda' x \geq 0, \forall x \in C \}.
\]

Note the symmetry between primal and dual problems. The strong duality relation \( f^\star = q^\star \) can be written as

\[
\inf_{x \in C} f(x) = -\inf_{\lambda \in \hat{C}} f^\star(\lambda).
\]

The following proposition translates the conditions of Prop. 1.2.1(a) to guarantee that there is no duality gap and that the dual problem has an optimal solution.
Proposition 1.2.2: (Conic Duality Theorem) Assume that the optimal value of the primal conic problem (1.23) is finite, and that \( \text{ri}(\text{dom}(f)) \cap \text{ri}(C) \neq \emptyset \). Then, there is no duality gap and the dual problem (1.24) has an optimal solution.

Using the symmetry of the primal and dual problems, we also obtain that there is no duality gap and the primal problem (1.23) has an optimal solution if the optimal value of the dual conic problem (1.24) is finite and \( \text{ri}(\text{dom}(f^*)) \cap \text{ri}(\hat{C}) \neq \emptyset \). It is also possible to derive primal and dual optimality conditions using Prop. 1.2.1(c).

1.2.1 Linear-Conic Problems

An important special case of conic programming, called linear-conic problem, arises when \( \text{dom}(f) \) is an affine set and \( f \) is linear over \( \text{dom}(f) \), i.e.,

\[
f(x) = \begin{cases} c' x & \text{if } x \in b + S, \\ \infty & \text{if } x \notin b + S, \end{cases}
\]

where \( b \) and \( c \) are given vectors, and \( S \) is a subspace. Then the primal problem can be written as

\[
\begin{align*}
\text{minimize} & \quad c' x \\
\text{subject to} & \quad x - b \in S, \quad x \in C;
\end{align*}
\]

see Fig. 1.2.1.
To derive the dual problem, we note that

\[
f^*(\lambda) = \sup_{x - b \in S} (\lambda - c)'x
\]

\[
= \sup_{y \in S} (\lambda - c)'(y + b)
\]

\[
= \begin{cases} (\lambda - c)'b & \text{if } \lambda - c \in S^\perp, \\ \infty & \text{if } \lambda - c \notin S. \end{cases}
\]

It can be seen that the dual problem \( \min_{\lambda \in \hat{C}} f^*(\lambda) \) [cf. Eq. (1.24)], after discarding the superfluous term \( c'b \) from the cost, can be written as

\[
\begin{align*}
\text{minimize} & \quad b'\lambda \\
\text{subject to} & \quad \lambda - c \in S^\perp, \quad \lambda \in \hat{C},
\end{align*}
\]

(1.26)

where \( \hat{C} \) is the dual cone:

\[
\hat{C} = \{ \lambda \mid x'\lambda \geq 0, \forall x \in C \}.
\]

An important special case of conic programming, called linear-conic problem, arises when \( \text{dom}(f) \) is an affine set and \( f \) is linear over \( \text{dom}(f) \), i.e.,

\[
f(x) = \begin{cases} c'x & \text{if } x \in b + S, \\ \infty & \text{if } x \notin b + S, \end{cases}
\]

where \( b \) and \( c \) are given vectors, and \( S \) is a subspace. Then the primal problem can be written as

\[
\begin{align*}
\text{minimize} & \quad c'x \\
\text{subject to} & \quad x - b \in S, \quad x \in C.
\end{align*}
\]

(1.27)

To derive the dual problem, we note that

\[
f^*(\lambda) = \sup_{x - b \in S} (\lambda - c)'x
\]

\[
= \sup_{y \in S} (\lambda - c)'(y + b)
\]

\[
= \begin{cases} (\lambda - c)'b & \text{if } \lambda - c \in S^\perp, \\ \infty & \text{if } \lambda - c \notin S. \end{cases}
\]

It can be seen that the dual problem (1.24), after discarding the superfluous term \( c'b \) from the cost, can be written as

\[
\begin{align*}
\text{minimize} & \quad b'\lambda \\
\text{subject to} & \quad \lambda - c \in S^\perp, \quad \lambda \in \hat{C}.
\end{align*}
\]

(1.28)
The following proposition translates the conditions of Prop. 1.2.2 to the linear-conic duality context.

**Proposition 1.2.3: (Linear-Conic Duality Theorem)** Assume that the primal problem (1.27) has finite optimal value. Assume further that \((b + S) \cap \text{ri}(C) = \emptyset\). Then, there is no duality gap and the dual problem has an optimal solution.

**Special Forms of Linear-Conic Problems**

The primal and dual linear-conic problems (1.27) and (1.28) have been placed in an elegant symmetric form. There are also other useful formats that parallel and generalize similar formats in linear programming (cf. Section 5.2 of [Ber09]). For example, we have the following dual problem pairs:

\[
\begin{align*}
\min_{Ax=b, \ x \in C} c'x & \iff \max_{c-A'\lambda \in \hat{C}} b'\lambda, & \quad (1.29) \\
\min_{Ax=b \in C} c'x & \iff \max_{A'\lambda = c, \ \lambda \in \hat{C}} b'\lambda, & \quad (1.30)
\end{align*}
\]

where \(x \in \mathbb{R}^n, \lambda \in \mathbb{R}^m, c \in \mathbb{R}^n, b \in \mathbb{R}^m\), and \(A\) is an \(m \times n\) matrix.

To verify the duality relation (1.29), let \(\overline{x}\) be any vector such that \(A\overline{x} = b\), and let us write the primal problem on the left in the primal conic form (1.27) as

\[
\begin{align*}
\text{minimize} & \quad c'x \\
\text{subject to} & \quad x - \overline{x} \in N(A), \quad x \in C,
\end{align*}
\]

where \(N(A)\) is the nullspace of \(A\). The corresponding dual conic problem (1.28) is to solve for \(\mu\) the problem

\[
\begin{align*}
\text{minimize} & \quad \overline{x}'\mu \\
\text{subject to} & \quad \mu - c \in N(A)^\perp, \quad \mu \in \hat{C}.
\end{align*}
\]

Since \(N(A)^\perp = \text{Ra}(A')\), the range of \(A'\), the constraints of problem (1.32) can be equivalently written as \(c - \mu \in -\text{Ra}(A') = \text{Ra}(A'), \mu \in \hat{C}\), or

\[c - \mu = A'\lambda, \quad \mu \in \hat{C},\]

for some \(\lambda \in \mathbb{R}^m\). Making the change of variables \(\mu = c - A'\lambda\), the dual problem (1.32) can be written as

\[
\begin{align*}
\text{minimize} & \quad \overline{x}'(c - A'\lambda) \\
\text{subject to} & \quad c - A'\lambda \in \hat{C}.
\end{align*}
\]
By discarding the constant $\mathbf{c}'\mathbf{c}$ from the cost function, using the fact $A\mathbf{x} = \mathbf{b}$, and changing from minimization to maximization, we see that this dual problem is equivalent to the one in the right-hand side of the duality pair (1.29). The duality relation (1.30) is proved similarly.

We next discuss two important special cases of conic programming: second order cone programming and semidefinite programming. These problems involve two different special cones, and an explicit definition of the affine set constraint. They arise in a variety of practical settings, and their computational difficulty tends to lie between that of linear and quadratic programming on one hand, and general convex programming on the other hand.

1.2.2 Second Order Cone Programming

In this section we consider the linear-conic problem (1.30), with the cone

$$C = \left\{ (x_1, \ldots, x_n) \mid x_n \geq \sqrt{x_1^2 + \cdots + x_{n-1}^2} \right\},$$

which is known as the second order cone (see Fig. 1.2.2). The dual cone is

$$\hat{C} = \{ y \mid 0 \leq y'x, \forall x \in C \} = \left\{ y \mid 0 \leq \inf_{\|(x_1, \ldots, x_{n-1})\| \leq x_n} y'x \right\},$$

and it can be shown that $\hat{C} = C$. This property is referred to as self-duality of the second order cone, and is fairly evident from Fig. 1.2.2. For a proof, we write

$$\inf_{\|(x_1, \ldots, x_{n-1})\| \leq x_n} y'x = \inf_{x_n \geq 0} \left\{ y_n x_n + \inf_{\|(x_1, \ldots, x_{n-1})\| \leq x_n} \sum_{i=1}^{n-1} y_i x_i \right\}$$

$$= \inf_{x_n \geq 0} \left\{ y_n x_n - \|(y_1, \ldots, y_{n-1})\|_1 x_n \right\}$$

$$= \begin{cases} 0 & \text{if } \|(y_1, \ldots, y_{n-1})\|_1 \leq y_n, \\ -\infty & \text{otherwise}, \end{cases}$$

where the second equality follows because the minimum of the inner product of a vector $z \in \mathbb{R}^{n-1}$ with vectors in the unit ball of $\mathbb{R}^{n-1}$ is $-\|z\|$. Combining the preceding two relations, we have

$$y \in \hat{C} \text{ if and only if } 0 \leq y_n - \|(y_1, \ldots, y_{n-1})\|_1,$$

so $\hat{C} = C$.

The second order cone programming problem (SOCP for short) is

$$\begin{align*}
\text{minimize} & \quad \mathbf{c}'\mathbf{x} \\
\text{subject to} & \quad A_i \mathbf{x} - b_i \in C_i, \ i = 1, \ldots, m,
\end{align*}$$

(1.33)
Figure 1.2.2. The second order cone

\[ C = \left\{ (x_1, \ldots, x_n) \mid x_n \geq \sqrt{x_1^2 + \cdots + x_{n-1}^2} \right\}, \]

in \( \mathbb{R}^3 \).

where \( x \in \mathbb{R}^n \), \( c \) is a vector in \( \mathbb{R}^n \), and for \( i = 1, \ldots, m \), \( A_i \) is an \( n_i \times n \) matrix, \( b_i \) is a vector in \( \mathbb{R}^{n_i} \), and \( C_i \) is the second order cone of \( \mathbb{R}^{n_i} \). It is seen to be a special case of the primal problem in the left-hand side of the duality relation (1.30), where

\[
A = \begin{pmatrix}
A_1 \\
\vdots \\
A_m
\end{pmatrix}, \quad b = \begin{pmatrix}
b_1 \\
\vdots \\
b_m
\end{pmatrix}, \quad C = C_1 \times \cdots \times C_m.
\]

Note that linear inequality constraints of the form \( a'_i x - b_i \geq 0 \) can be written as

\[
\begin{pmatrix} 0 \\ a'_i \end{pmatrix} x - \begin{pmatrix} 0 \\ b_i \end{pmatrix} \in C_i,
\]

where \( C_i \) is the second order cone of \( \mathbb{R}^2 \). As a result, linear-conic problems involving second order cones contain as special cases linear programming problems.

We now observe that from the right-hand side of the duality relation (1.30), and the self-duality relation \( C = \hat{C} \), the corresponding dual linear-conic problem has the form

\[
\begin{aligned}
\text{maximize} & \quad \sum_{i=1}^{m} b'_i \lambda_i \\
\text{subject to} & \quad \sum_{i=1}^{m} A'_i \lambda_i = c, \quad \lambda_i \in C_i, \quad i = 1, \ldots, m,
\end{aligned}
\]

(1.34)
where $\lambda = (\lambda_1, \ldots, \lambda_m)$. By applying the duality result of Prop. 1.2.3, we have the following.

**Proposition 1.2.4: (Second Order Cone Duality Theorem)**

Consider the primal SOCP (1.33), and its dual problem (1.34).

(a) If the optimal value of the primal problem is finite and there exists a feasible solution $x$ such that

$$A_ix - b_i \in \text{int}(C_i), \quad i = 1, \ldots, m,$$

then there is no duality gap, and the dual problem has an optimal solution.

(b) If the optimal value of the dual problem is finite and there exists a feasible solution $\lambda = (\lambda_1, \ldots, \lambda_m)$ such that

$$\lambda_i \in \text{int}(C_i), \quad i = 1, \ldots, m,$$

then there is no duality gap, and the primal problem has an optimal solution.

Note that while Prop. 1.2.3 requires a relative interior point condition, the preceding proposition requires an interior point condition. The reason is that the second order cone has nonempty interior, so its relative interior coincides with its interior.

The SOCP arises in many application contexts, and significantly, it can be solved numerically with powerful specialized algorithms that belong to the class of interior point methods, to be discussed in Chapter 5. We refer to the literature for a more detailed description and analysis (see e.g., [BeT01], and [BoV04]).

Generally, SOCPs can be recognized from the presence of convex quadratic functions in the cost or the constraint functions. The following are illustrative examples.

**Example 1.2.1: (Robust Linear Programming)**

Frequently, there is uncertainty about the data of an optimization problem, so one would like to have a solution that is adequate for a whole range of the uncertainty. A popular formulation of this type, is to assume that the constraints contain parameters that take values in a given set, and require that the constraints are satisfied for all values in that set. This approach is also known as a set membership description of the uncertainty and has also been used in fields other than optimization, such as set membership estimation, and minimax control.
As an example, consider the problem

$$ \begin{align*}
\text{minimize} & \quad c'x \\
\text{subject to} & \quad a_j'x \leq b_j, \quad \forall (a_j, b_j) \in T_j, \quad j = 1, \ldots, r,
\end{align*} \tag{1.35} $$

where $c \in \mathbb{R}^n$ is a given vector, and $T_j$ is a given subset of $\mathbb{R}^{n+1}$ to which the constraint parameter vectors $(a_j, b_j)$ must belong. The vector $x$ must be chosen so that the constraint $a_j'x \leq b_j$ is satisfied for all $(a_j, b_j) \in T_j$, $j = 1, \ldots, r$.

Generally, when $T_j$ contains an infinite number of elements, this problem involves a correspondingly infinite number of constraints. To convert the problem to one involving a finite number of constraints, we note that

$$ a_j'x \leq b_j, \quad \forall (a_j, b_j) \in T_j \quad \text{if and only if} \quad g_j(x) \leq 0, $$

where $g_j(x) = \sup_{(a_j, b_j) \in T_j} \{a_j'x - b_j\}$. \tag{1.36}

Thus, the robust linear programming problem (1.35) is equivalent to

$$ \begin{align*}
\text{minimize} & \quad c'x \\
\text{subject to} & \quad g_j(x) \leq 0, \quad j = 1, \ldots, r.
\end{align*} $$

For special choices of the set $T_j$, the function $g_j$ can be expressed in closed form, and in the case where $T_j$ is an ellipsoid, it turns out that the constraint $g_j(x) \leq 0$ can be expressed in terms of a second order cone. To see this, let

$$ T_j = \{ (\pi_j + P_j u_j, \bar{b}_j + q_j' u_j) \mid \|u_j\| \leq 1, \ u_j \in \mathbb{R}^{n_j} \}, \tag{1.37} $$

where $P_j$ is a given $n \times n_j$ matrix, $\pi_j \in \mathbb{R}^n$ and $q_j \in \mathbb{R}^{n_j}$ are given vectors, and $\bar{b}_j$ is a given scalar. Then, from Eqs. (1.36) and (1.37),

$$ g_j(x) = \sup_{\|u_j\| \leq 1} \{ (\pi_j + P_j u_j)'x - (\bar{b}_j + q_j' u_j) \} $$

$$ = \sup_{\|u_j\| \leq 1} (P_j' x - q_j)' u_j + \pi_j x - \bar{b}_j, $$

and finally

$$ g_j(x) = \|P_j' x - q_j\| + \pi_j x - \bar{b}_j. $$

Thus,

$$ g_j(x) \leq 0 \quad \text{if and only if} \quad (P_j' x - q_j, \bar{b}_j - \pi_j') \in C_j, $$

where $C_j$ is the second order cone of $\mathbb{R}^{n_j+1}$; i.e., the “robust” constraint $g_j(x) \leq 0$ is equivalent to a second order cone constraint. It follows that in the case of ellipsoidal uncertainty, the robust linear programming problem (1.35) is a SOCP of the form (1.33).
Example 1.2.2: (Quadratically Constrained Quadratic Problems)

Consider the quadratically constrained quadratic problem

$$\begin{align*}
\text{minimize} & \quad x'Q_0x + 2q_0'x + p_0 \\
\text{subject to} & \quad x'Q_jx + 2q_j'x + p_j \leq 0, \quad j = 1, \ldots, r,
\end{align*}$$

where $Q_0, \ldots, Q_r$ are symmetric $n \times n$ positive definite matrices, $q_0, \ldots, q_r$ are vectors in $\mathbb{R}^n$, and $p_0, \ldots, p_r$ are scalars. We show that the problem can be converted to the second order cone format. A similar conversion is also possible for the quadratic programming problem where $Q_0$ is positive definite and $Q_j = 0$, $j = 1, \ldots, r$.

Indeed, since each $Q_j$ is symmetric and positive definite, we have

$$x'Q_jx + 2q_j'x + p_j = (Q_j^{1/2}x)^' Q_j^{1/2}x + 2 (Q_j^{-1/2}q_j)^' Q_j^{1/2}x + p_j$$

for $j = 0, 1, \ldots, r$. Thus, the problem can be written as

$$\begin{align*}
\text{minimize} & \quad \|Q_j^{1/2}x + Q_j^{-1/2}q_j\|^2 + p_j - q_j'Q_j^{-1}q_j \\
\text{subject to} & \quad \|Q_j^{1/2}x + Q_j^{-1/2}q_j\| \leq (q_j'Q_j^{-1}q_j - p_j)^{1/2}, \quad j = 1, \ldots, r.
\end{align*}$$

or, by neglecting the constant $p_0 - q_0'Q_0^{-1}q_0$,

$$\begin{align*}
\text{minimize} & \quad \|Q_0^{1/2}x + Q_0^{-1/2}q_0\| \\
\text{subject to} & \quad \|Q_j^{1/2}x + Q_j^{-1/2}q_j\| \leq (q_j'Q_j^{-1}q_j - p_j)^{1/2}, \quad j = 1, \ldots, r.
\end{align*}$$

By introducing an auxiliary variable $x_{n+1}$, the problem can be written as

$$\begin{align*}
\text{minimize} & \quad x_{n+1} \\
\text{subject to} & \quad \|Q_0^{1/2}x + Q_0^{-1/2}q_0\| \leq x_{n+1} \\
& \quad \|Q_j^{1/2}x + Q_j^{-1/2}q_j\| \leq (q_j'Q_j^{-1}q_j - p_j)^{1/2}, \quad j = 1, \ldots, r.
\end{align*}$$

It can be seen that this problem has the second order cone form (1.33).

We finally note that the problem of this example is special in that it has no duality gap, assuming its optimal value is finite, i.e., there is no need for the interior point conditions of Prop. 1.2.4. This can be traced to the fact that linear transformations preserve the closure of sets defined by quadratic constraints (see e.g., BNO03], Section 1.5.2).
1.2.3 Semidefinite Programming

In this section we consider the linear-conic problem (1.29) with $C$ being the cone of matrices that are positive semidefinite.† This is called the positive semidefinite cone. To define the problem, we view the space of symmetric $n \times n$ matrices as the space $\mathbb{R}^{n^2}$ with the inner product

$$<X,Y> = \text{trace}(XY) = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} y_{ij}.$$ 

The interior of $C$ is the set of positive definite matrices.

The dual cone is

$$\hat{C} = \{ Y \mid \text{trace}(XY) \geq 0, \ \forall \ X \in C \},$$

and it can be shown that $\hat{C} = C$, i.e., $C$ is self-dual. Indeed, if $Y \notin C$, there exists a vector $v \in \mathbb{R}^n$ such that

$$0 > v'Yv = \text{trace}(vv'Y).$$

Hence the positive semidefinite matrix $X = vv'$ satisfies $0 > \text{trace}(XY)$, so $Y \notin \hat{C}$ and it follows that $C \supset \hat{C}$. Conversely, let $Y \in C$, and let $X$ be any positive semidefinite matrix. We can express $X$ as

$$X = \sum_{i=1}^{n} \lambda_i e_i e_i',$n

where $\lambda_i$ are the nonnegative eigenvalues of $X$, and $e_i$ are corresponding orthonormal eigenvectors. Then,

$$\text{trace}(XY) = \text{trace} \left( Y \sum_{i=1}^{n} \lambda_i e_i e_i' \right) = \sum_{i=1}^{n} \lambda_i e_i'Ye_i \geq 0.$$ 

It follows that $Y \in \hat{C}$, and $C \subset \hat{C}$.

The semidefinite programming problem (SDP for short) is to minimize a linear function of a symmetric matrix over the intersection of an affine set with the positive semidefinite cone. It has the form

$$\begin{align*}
\text{minimize} & \quad <D,X > \\
\text{subject to} & \quad <A_i,X > = b_i, \quad i = 1, \ldots, m, \quad X \in C,
\end{align*}$$

† As noted in Appendix A, throughout this book a positive semidefinite matrix is implicitly assumed to be symmetric.
Sec. 1.2  Fenchel Duality and Conic Programming

where \( D, A_1, \ldots, A_m, \) are given \( n \times n \) symmetric matrices, and \( b_1, \ldots, b_m, \) are given scalars. It is seen to be a special case of the primal problem in the left-hand side of the duality relation (1.29).

We can view the SDP as a problem with linear cost, linear constraints, and a convex set constraint. Then, similar to the case of SOCP, it can be verified that the dual problem (1.28), as given by the right-hand side of the duality relation (1.29), takes the form

\[
\begin{align*}
\text{maximize} & \quad b^\prime \lambda \\
\text{subject to} & \quad D - (\lambda_1 A_1 + \cdots + \lambda_m A_m) \in C,
\end{align*}
\]

where \( b = (b_1, \ldots, b_m) \) and the maximization is over the vector \( \lambda = (\lambda_1, \ldots, \lambda_m) \). By applying the duality result of Prop. 1.2.3, we have the following proposition.

**Proposition 1.2.5: (Semidefinite Duality Theorem)** Consider the primal SDP (1.38), and its dual problem (1.39).

(a) If the optimal value of the primal problem is finite and there exists a primal-feasible solution, which is positive definite, then there is no duality gap, and the dual problem has an optimal solution.

(b) If the optimal value of the dual problem is finite and there exist scalars \( \lambda_1, \ldots, \lambda_m \) such that \( D - (\lambda_1 A_1 + \cdots + \lambda_m A_m) \) is positive definite, then there is no duality gap, and the primal problem has an optimal solution.

The SDP is a fairly general problem. In particular, it can be shown that a SOCP can be cast as a SDP. Thus SDP involves a more general structure than SOCP. This is consistent with the practical observation that the latter problem is generally more amenable to computational solution. We provide some examples of problem formulation as an SDP.

**Example 1.2.3: (Minimizing the Maximum Eigenvalue)**

Given a symmetric \( n \times n \) matrix \( M(\lambda) \), which depends on a parameter vector \( \lambda = (\lambda_1, \ldots, \lambda_m) \), we want to choose \( \lambda \) so as to minimize the maximum eigenvalue of \( M(\lambda) \). We pose this problem as

\[
\begin{align*}
\text{minimize} & \quad z \\
\text{subject to} & \quad \text{maximum eigenvalue of } M(\lambda) \leq z,
\end{align*}
\]

or equivalently

\[
\begin{align*}
\text{minimize} & \quad z \\
\text{subject to} & \quad zI - M(\lambda) \in D,
\end{align*}
\]
where $I$ is the $n \times n$ identity matrix, and $D$ is the semidefinite cone. If $M(\lambda)$ is an affine function of $\lambda$,

$$M(\lambda) = C + \lambda_1 M_1 + \cdots + \lambda_m M_m,$$

the problem has the form of the dual problem (1.39), with the optimization variables being $(z, \lambda_1, \ldots, \lambda_m)$.

**Example 1.2.4: (Semidefinite Relaxation - Lower Bounds for Discrete Optimization Problems)**

Semidefinite programming provides an effective means for deriving lower bounds to the optimal value of several types of discrete optimization problems. As an example, consider the following quadratic problem with quadratic equality constraints

$$\begin{align*}
\text{minimize} & \quad x' Q_0 x + a'_0 x + b_0 \\
\text{subject to} & \quad x' Q_i x + a'_i x + b_i = 0, \quad i = 1, \ldots, m, \quad (1.40)
\end{align*}$$

where $Q_0, \ldots, Q_m$ are symmetric $n \times n$ matrices, $a_0, \ldots, a_m$ are vectors in $\mathbb{R}^n$, and $b_0, \ldots, b_m$ are scalars.

This problem can be used to model broad classes of discrete optimization problems. To see this, consider an integer constraint that a variable $x_i$ must be either 0 or 1. Such a constraint can be expressed by the quadratic equality $x_i^2 - x_i = 0$. Furthermore, a linear inequality constraint $a'_j x \leq b_j$ can be expressed as the quadratic equality constraint $y_j^2 + a'_j x - b_j = 0$, where $y_j$ is an additional variable.

Introducing a multiplier vector $\lambda = (\lambda_1, \ldots, \lambda_m)$, the dual function is given by

$$q(\lambda) = \inf_{x \in \mathbb{R}^n} \{x' Q(\lambda) x + a(\lambda)' x + b(\lambda)\},$$

where

$$Q(\lambda) = Q_0 + \sum_{i=1}^m \lambda_i Q_i, \quad a(\lambda) = a_0 + \sum_{i=1}^m \lambda_i a_i, \quad b(\lambda) = b_0 + \sum_{i=1}^m \lambda_i b_i.$$

Let $f^*$ and $q^*$ be the optimal values of problem (1.40) and its dual, and note that by weak duality, we have $f^* \geq q^*$. By introducing an auxiliary scalar variable $\xi$, we see that the dual problem is to find a pair $(\xi, \lambda)$ that solves the problem

$$\begin{align*}
\text{maximize} & \quad \xi \\
\text{subject to} & \quad q(\lambda) \geq \xi.
\end{align*}$$

The constraint $q(\lambda) \geq \xi$ of this problem can be written as

$$\inf_{x \in \mathbb{R}^n} \{x' Q(\lambda) x + a(\lambda)' x + b(\lambda) - \xi\} \geq 0,$$
or equivalently, introducing a scalar variable $t$,
\[
\inf_{x \in \mathbb{R}^n, t \in \mathbb{R}} \left\{ (tx)'Q(\lambda)(tx) + a(\lambda)'tx + (b(\lambda) - \xi)t^2 \right\} \geq 0.
\]
This relation can be equivalently written as
\[
\inf_{x \in \mathbb{R}^n, t \in \mathbb{R}} \left\{ x'Q(\lambda)x + a(\lambda)'xt + (b(\lambda) - \xi)t^2 \right\} \geq 0,
\]
or
\[
\left(\begin{array}{cc}
Q(\lambda) & \frac{1}{2}a(\lambda) \\
\frac{1}{2}a(\lambda)' & b(\lambda) - \xi
\end{array}\right) \in C,
\]
where $C$ is the positive semidefinite cone. Thus the dual problem is equivalent to the SDP of maximizing $\xi$ over all $(\xi, \lambda)$ satisfying the constraint (1.41), and its optimal value $q^*$ is a lower bound to $f^*$.

1.3 ADDITIVE COST PROBLEMS

In this section we focus on a structural characteristic that arises in several important contexts: a cost function that is the sum of a large number of components,
\[
f(x) = \sum_{i=1}^{m} f_i(x),
\]
where the functions $f_i$ are real-valued and convex. Such functions can be minimized with specialized methods, called incremental, which exploit their additive structure, by updating $x$ using one component function $f_i$ at a time (see Section 1.6.3). Problems with additive cost functions can also be treated with specialized outer and inner linearization methods that approximate the component functions $f_i$ individually (rather than approximating $f$); see Section 3.4.

An important special case is the cost function of the dual of a separable problem
\[
\text{maximize } \sum_{i=1}^{m} q_i(\mu) - \mu'b
\]
subject to $\mu \geq 0$,
where
\[
q_i(\mu) = \inf_{x_i \in \mathbb{R}} \left\{ f_i(x_i) + x_i \sum_{j=1}^{r} \mu_j a_{ji} \right\},
\]
and $\mu = (\mu_1, \ldots, \mu_r)$ [cf. Eq. (1.13)]. After a sign change to convert to minimization it takes the form (1.42) with
\[
f_i(\mu) = -q_i(\mu) + p_i(\mu),
\]
where $p_i$ are any functions such that $p_1(\mu) + \cdots + p_m(\mu) = \mu'b$. This is a major class of additive cost problems. The next three examples are also very important and arise in many machine learning applications.
Example 1.3.1: (Regularized Regression)

This is a broad class of applications that relate to parameter estimation. The cost function involves a sum of terms \( f_i(x) \), each corresponding to the error between some data and the output of a parametric model, with \( x \) being the vector of parameters. An example is linear least squares problems, also referred to as linear regression problems, where \( f_i \) has quadratic structure. Often a convex regularization function \( R(x) \) is added to the least squares objective, to induce desirable properties of the solution. This gives rise to problems of the form

\[
\min_x R(x) + \frac{1}{2} \sum_{i=1}^{m} (c_i'x - b_i)^2
\]

subject to \( x \in \mathbb{R}^n \),

where \( c_i \) and \( b_i \) are given vectors and scalars, respectively.

In statistical applications, such a problem arises when constructing a linear model for an unknown input-output relation. The model involves a vector of parameters \( x \), to be determined, which weigh input data (the components of the vectors \( c_i \)). The inner products \( c_i'x \) produced by the model are matched against the scalars \( b_i \), which are observed output data, corresponding to inputs \( c_i \) from the true input-output relation that we try to represent. The optimal vector of parameters \( x^* \) provides the model that minimizes the sum of the squared errors \( c_i'x^* - b_i \).

In a more general version of the problem, a nonlinear parametric model is constructed, giving rise to a nonlinear least squares problem of the form

\[
\min_x R(x) + \sum_{i=1}^{m} \| g_i(x) \|^2
\]

subject to \( x \in \mathbb{R}^n \),

where \( g_i : \mathbb{R}^n \rightarrow \mathbb{R}^n \) are given nonlinear functions that depend on the data. This is also a common problem, referred to as nonlinear regression, which, however, is often nonconvex.

It is also possible to use a nonquadratic function of the error between some data and the output of a linear parametric model. Thus in place of the squared error \((1/2)(c_i'x - b_i)^2\), we may use

\[ h_i(c_i'x - b_i), \]

where \( h_i : \mathbb{R} \rightarrow \mathbb{R} \) is a convex function, leading to the problem

\[
\min_x R(x) + \sum_{i=1}^{m} h_i(c_i'x - b_i)
\]

subject to \( x \in \mathbb{R}^n \).
Generally the choice of the function \( h_i \) is dictated by statistical modeling considerations, for which the reader may consult the relevant literature. An example is
\[
h_i(c_i'x - b_i) = |c_i'x - b_i|,
\]
which tends to result in a more robust estimate than least squares in the presence of large outliers in the data. This is known as the least absolute deviations method.

The regularization function \( R \) is often taken to be differentiable, and particularly quadratic. Also popular are nondifferentiable regularization functions, such as for example \( \ell_1 \)-regularization, where
\[
R(x) = \gamma \|x\|_1 = \gamma \sum_{j=1}^{n} |x^j|, \tag{1.45}
\]
\( \gamma \) is a positive scalar and \( x^j \) is the \( j \)th coordinate of \( x \). The reason for the popularity of the \( \ell_1 \) norm \( \|x\|_1 \) is that it tends to produce optimal solutions where a greater number of components \( x^j \) are zero, relative to the case of quadratic regularization. This is considered desirable in many statistical applications, where the number of parameters to include in a model may not be known a priori. The special case where a linear least squares model is used,
\[
\begin{align*}
\text{minimize} \quad & \gamma \|x\|_1 + \frac{1}{2} \sum_{i=1}^{m} (c_i'x - b_i)^2 \\
\text{subject to} \quad & x \in \mathbb{R}^n,
\end{align*}
\]
is known as the lasso problem.

In a generalization of the lasso problem, the \( \ell_1 \) regularization function \( \|x\|_1 \) is replaced by a scaled version \( \|Sx\|_1 \), where \( S \) is some scaling matrix. The term \( \|Sx\|_1 \) then induces a penalty on some undesirable characteristic of the solution. For example the problem
\[
\begin{align*}
\text{minimize} \quad & \gamma \sum_{i=1}^{n-1} |x_{i+1} - x_i| + \frac{1}{2} \sum_{i=1}^{m} (c_i'x - b_i)^2 \\
\text{subject to} \quad & x \in \mathbb{R}^n,
\end{align*}
\]
is known as the total variation denoising problem.

Sometimes combinations of quadratic and \( \ell_1 \) regularization functions are used. There are also constrained variants of the problems just discussed, where the parameter vector \( x \) is required to belong to some subset of \( \mathbb{R}^n \), such as the nonnegative orthant or a “box” formed by given upper and lower bounds on the components of \( x \). Such constraints may be used to encode into the model some prior knowledge about the nature of the solution.
Example 1.3.2: (Classification)

In the regression problems of the preceding example we aim to construct a parametric model that matches well an input-output relationship based on given data. Similar problems arise in classification problems where we try to construct a parametric model for predicting whether an object with certain characteristics (also called features) belongs to a given category or not.

We assume that each object is characterized by a feature vector $c$ that belongs to $\mathbb{R}^n$ and a label $b$ that takes the values $+1$ or $-1$, if the object belongs to the category or not, respectively. As illustration consider a bank that wishes to classify loan applicants as low risk (+1) or high risk (-1), with each customer characterized by $n$ scalar features of financial and personal type.

We are given data, which is a set of feature-label pairs $(c_i, b_i), i = 1, \ldots, m$. Based on this data, we want to find a parameter vector $x \in \mathbb{R}^n$ and a scalar $y \in \mathbb{R}$ such that the sign of $c_i'x + y$ is a good predictor of the label of an object with feature vector $c$. Thus, loosely speaking, $x$ and $y$ should be such that for “most” of the given feature-label data $(c_i, b_i)$ we have

$$c_i'x + y > 0, \quad \text{if } b_i = +1,$$
$$c_i'x + y < 0, \quad \text{if } b_i = -1.$$ 

In the statistical literature, $c_i'x + y$ is often called the discriminant function, and the value of $b_i(c_i'x + y)$ for a given object $i$ is called the margin of the object.

Note that a classification error is made when the margin of an object is negative, so it makes sense to formulate this as an optimization problem where negative margins are penalized. This leads to the problem

$$\minimize \ R(x) + \sum_{i=1}^{m} h(b_i(c_i'x + y))$$

subject to $x \in \mathbb{R}^n$, $y \in \mathbb{R}$,

where $R$ is a suitable regularization function, and $h : \mathbb{R} \mapsto \mathbb{R}$ is a convex function that penalizes negative values of its argument. It would make some sense to use a penalty of one unit for misclassification, i.e.,

$$h(z) = \begin{cases} 
0 & \text{if } z \geq 0, \\
1 & \text{if } z < 0,
\end{cases}$$

but such a penalty function is discontinuous. To obtain a continuous cost function, we allow a continuous transition of $h$ from negative to positive values, leading to a variety of nonincreasing functions $h$. The specific choice of $h$ depends on the given application and other theoretical considerations for which we refer to the machine learning literature. Some common examples are

$$h(z) = e^{-z}, \quad \text{(exponential loss)},$$
Sec. 1.3  Additive Cost Problems

\[ h(z) = \log \left( 1 + e^{-z} \right), \quad \text{(logistic loss)}, \]
\[ h(z) = \max \{ 0, 1 - z \}, \quad \text{(hinge loss)}. \]

For the case of exponential loss the method comes under the methodology of boosting, for the case of logistic loss the method comes under the methodology of logistic regression, and for the case of hinge loss the method comes under the methodology of support vector machines. As in the case of regression, the regularization function \( R \) could be quadratic, the \( \ell_1 \) norm, or some scaled version or combination thereof. We refer to the machine learning literature for detailed discussions.

Example 1.3.3: (Maximum Likelihood Estimation)

The maximum likelihood approach is a major statistical inference approach for parameter estimation, which is described in many sources (see e.g., the textbooks [Was04], [HTF09]). In fact in many cases, a maximum likelihood formulation is used to provide a probabilistic justification of the regression and classification models of the preceding two examples.

Here we observe a sample of a random vector \( Z \) whose distribution \( P_Z(\cdot; x) \) depends on an unknown parameter vector \( x \in \mathbb{R}^n \). For simplicity we assume that \( Z \) can take only a finite set of values, so that \( P_Z(z; x) \) is the probability that \( Z \) takes the value \( z \) when the parameter vector has the value \( x \). We estimate \( x \) based on the given sample value \( z \), by solving the problem

\[
\begin{align*}
\text{maximize} & \quad P_Z(z; x) \\
\text{subject to} & \quad x \in \mathbb{R}^n.
\end{align*}
\]

(1.46)

The cost function \( P_Z(z; \cdot) \) of this problem may either have an additive structure or may be equivalent to a problem that has an additive structure. For example the event that \( Z = z \) may be the union of a large number of disjoint events, so \( P_Z(z; x) \) is the probability that \( Z \) takes the value \( z \) when the parameter vector has the value \( x \).

Then the maximization (1.46) is equivalent to the additive cost minimization

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m} f_i(x) \\
\text{subject to} & \quad x \in \mathbb{R}^n,
\end{align*}
\]

where
\[
f_i(x) = - \log P(z_i; x).
\]

In many applications the number of samples \( m \) is very large, in which case special methods that exploit the additive structure of the cost are recommended.
Example 1.3.4: (Minimization of an Expected Value - Stochastic Programming)

An important context where additive cost functions arise is the minimization of an expected value

\[
\begin{align*}
\text{minimize} & \quad E\{F(x,w)\} \\
\text{subject to} & \quad x \in X,
\end{align*}
\]

where \(w\) is a random variable taking a finite but very large number of values \(w_i, i = 1, \ldots, m\), with corresponding probabilities \(\pi_i\). Then the cost function consists of the sum of the \(m\) functions \(\pi_i F(x, w_i)\).

For example, in stochastic programming, a classical model of two-stage optimization under uncertainty, a vector \(x \in X\) is selected, a random event occurs that has \(m\) possible outcomes \(w_1, \ldots, w_m\), and another vector \(y \in Y\) is selected with knowledge of the outcome that occurred (see e.g., the books [BiL97], [KaW94], [Pre95], [SDR09]). Then for optimization purposes, we need to specify a different vector \(y_i \in Y\) for each outcome \(w_i\). The problem is to minimize the expected cost

\[
F(x) + \sum_{i=1}^{m} \pi_i G_i(y_i),
\]

where \(G_i(y_i)\) is the cost associated with the choice \(y_i\) and the occurrence of \(w_i\), and \(\pi_i\) is the corresponding probability. This is a problem with an additive cost function.

Additive cost functions also arise when the expected value cost function \(E\{F(x, w)\}\) is approximated by an \(m\)-sample average

\[
f(x) = \frac{1}{m} \sum_{i=1}^{m} F(x, w_i),
\]

where \(w_i\) are independent samples of the random variable \(w\). The minimum of the sample average \(f(x)\) is then taken as an approximation of the minimum of \(E\{F(x, w)\}\).

Generally additive cost problems arise when we want to strike a balance between several different types of costs by lumping them into a single cost function. The following is an example of a different character than the preceding ones.

Example 1.3.5: (Weber Problem in Location Theory)

A basic problem in location theory is to find a point \(x\) in the plane whose sum of weighted distances from a given set of points \(y_1, \ldots, y_m\) is minimized. Mathematically, the problem is

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m} w_i \|x - y_i\| \\
\text{subject to} & \quad x \in \mathbb{R}^n,
\end{align*}
\]
where \( w_1, \ldots, w_m \) are given positive scalars. This problem descends from the famous Fermat-Torricelli-Viviani problem (see [BMS99] for an account of the history of this problem).

Let us finally note a constrained version of additive cost problems where the functions \( f_i \) are convex and extended real-valued. This is essentially equivalent to constraining \( x \) to lie in the intersection of the domains of \( f_i \), resulting in a problem of the form

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m} f_i(x) \\
\text{subject to} & \quad x \in \cap_{i=1}^{m} X_i,
\end{align*}
\]

where \( f_i \) are convex and real-valued over \( X_i = \text{dom}(f_i) \). Methods that are well-suited for the unconstrained version of the problem where \( X_i \equiv \mathbb{R}^n \) can often be modified to apply to the constrained version, as we will see in Chapter 5. However, the case of constraint sets with many components arises independently of whether the cost function is additive or not, and has its own character, as we discuss in the next section.

### 1.4 LARGE NUMBER OF CONSTRAINTS

In this section we consider problems of the form

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in X, \quad g_j(x) \leq 0, \quad j = 1, \ldots, r,
\end{align*}
\]

where the number \( r \) of constraints is very large. Problems of this type occur often in practice, either directly or via reformulation from other problems. A similar type of problem arises when the abstract constraint set \( X \) consists of the intersection of many simpler sets:

\[
X = \cap_{\ell \in L} X_\ell,
\]

where \( L \) is a finite or infinite index set. There may or may not be additional inequality constraints \( g_j(x) \leq 0 \) like the ones in problem (1.48). We provide a few examples.

**Example 1.4.1: (Feasibility Problems)**

A simple but important problem, which arises in many contexts and embodies important algorithmic ideas, is a classical feasibility problem, where the objective is to find a common point within a collection of sets \( X_\ell, \ell \in L \), where each \( X_\ell \) is a closed convex set. In the feasibility problem the cost function is
zero. A somewhat more complicated problem with a similar structure arises when there is a convex cost function, i.e., a problem of the form

$$\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in \cap_{\ell \in L} X_{\ell},
\end{align*}$$

(1.49)

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is convex.

**Example 1.4.2: (Minimax Problems)**

In a minimax problem the cost function has the form

$$f(x) = \sup_{z \in Z} \phi(x, z),$$

where $Z$ is a subset of some space and $\phi(\cdot, z)$ is a real-valued function for each $z \in Z$. By introducing an artificial scalar variable $y$, we may transform such problems to the general form

$$\begin{align*}
\text{minimize} & \quad y \\
\text{subject to} & \quad x \in X, \quad \phi(x, z) \leq y, \quad \forall \ z \in Z,
\end{align*}$$

(1.50)

which involves a large number of constraints (one constraint for each $z$ in the set $Z$, which could be infinite). Of course in such problems the set $X$ may also be of the form $X = \cap_{\ell \in L} X_{\ell}$ as in the preceding example.

**Example 1.4.3: (Approximate Large-Scale Separable Problems - Dynamic Programming)**

Let us consider a large-scale separable problem of the form

$$\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m} f_i(y_i) \\
\text{subject to} & \quad \sum_{i=1}^{m} a_{ij}y_i \leq b_j, \quad \forall \ j = 1, \ldots, r, \quad y \geq 0.
\end{align*}$$

(1.51)

where the dimension $m$ of the vector $y = (y_1, \ldots, y_m)$ is very large. One possible way to address this problem is to approximate $y$ with a vector of the form $\Phi x$, where $\Phi$ is an $m \times n$ matrix. The columns of $\Phi$ may be relatively few, and may be viewed as basis functions for a low-dimensional approximation subspace

$$\{\Phi x \mid x \in \mathbb{R}^n\}.$$
We thus replace problem (1.51) with the approximate version

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m} f_i(\phi_i'x) \\
\text{subject to} & \quad \sum_{i=1}^{m} a_{ji}(\phi_i'x) \leq b_j, \quad \forall \ j = 1, \ldots, r, \\
& \quad \phi_i'x \geq 0, \quad i = 1, \ldots, m, 
\end{align*}
\]

(1.52)

where \(\phi_i'\) denotes the \(i\)th row of \(\Phi\), and \(\phi_i'x\) is viewed as an approximation of \(y_i\). Thus the dimension of the problem is reduced from \(m\) to \(n\). However, the constraint set of the problem became more complicated, because the simple constraints \(y_i \geq 0\) take the more complex form \(\phi_i'x \geq 0\). Moreover the number \(m\) of components in the cost function, as well as the number of its constraints is large. Thus the problem has the additive cost structure of the preceding section, as well as a large number of constraints.

A prominent application of this approach is in approximate dynamic programming (see e.g., [BeT96], [SuB98], [Pow07], [Ber12]), where \(y_i\) represents the optimal cost value starting from a state \(i\), and the function \(f_i\) is linear. Here the number of states \(m\) is often astronomical, and the exact solution of the corresponding problem (1.51) is impossible. For such problems, approximation based on problem (1.52) has been one of the major algorithmic approaches (see [Ber12] for a textbook presentation and references). Within this context, it is virtually impossible to calculate the cost function value \(\sum_{i=1}^{m} f_i(\phi_i'x)\) for a given \(x\), and one may at most be able to sample individual cost components \(f_i\). For this reason optimization by stochastic simulation is one of the most prominent approaches in large scale dynamic programming.

Problems with a large number of constraints also arise in problems involving a graph, and can often be handled with algorithms that take into account the graph structure. The following example is typical.

**Example 1.4.4: (Optimal Routing in a Data Network - Multicommodity Flows)**

We are given a directed graph, which is viewed as a model of a data communication network. We are also given a set \(W\) of ordered node pairs \(w = (i, j)\). The nodes \(i\) and \(j\) are referred to as the origin and the destination of \(w\), respectively, and \(w\) is referred to as an OD pair. For each \(w\), we are given a scalar \(r_w\) referred to as the input traffic of \(w\). In the context of routing of data in a communication network, \(r_w\) (measured in data units/second) is the arrival rate of traffic entering and exiting the network at the origin and the destination of \(w\), respectively. The routing objective is to divide each \(r_w\) among the many paths from origin to destination in a way that the resulting total arc flow pattern minimizes a suitable cost function. We denote:

\(P_w\): A given set of paths that start at the origin and end at the destination of \(w\). All arcs on each of these paths are oriented in the direction from the origin to the destination.
$x_p$: The portion of $r_w$ assigned to path $p$, also called the flow of path $p$.

The collection of all path flows $\{x_p \mid p \in P_w, w \in W\}$ must satisfy the constraints

$$\sum_{p \in P_w} x_p = r_w, \quad \forall w \in W, \quad (1.53)$$

$$x_p \geq 0, \quad \forall p \in P_w, w \in W. \quad (1.54)$$

The total flow $F_{ij}$ of arc $(i,j)$ is the sum of all path flows traversing the arc:

$$F_{ij} = \sum_{\text{all paths } p \text{ containing } (i,j)} x_p. \quad (1.55)$$

Consider a cost function of the form

$$\sum_{(i,j)} D_{ij}(F_{ij}). \quad (1.56)$$

The problem is to find a set of path flows $\{x_p\}$ that minimize this cost function subject to the constraints of Eqs. (1.53)-(1.55). We assume that $D_{ij}$ is a convex and continuously differentiable function of $F_{ij}$ with first derivative denoted by $D'_{ij}$. In data routing applications, the form of $D_{ij}$ is often based on a queueing model of average delay (see [BeG92]).

The preceding problem is known as a multicommodity network flow problem. The terminology reflects the fact that the arc flows consist of several different commodities; in the present example, the different commodities are the data of the distinct OD pairs. This problem also arises in essentially identical form in traffic network equilibrium problems (see e.g., [Ber99], Section 2.1).

By expressing the total flows $F_{ij}$ in terms of the path flows in the cost function (1.56) [using Eq. (1.55)], the problem can be formulated in terms of the path flow variables $\{x_p \mid p \in P_w, w \in W\}$ as

minimize $D(x)$

subject to $\sum_{p \in P_w} x_p = r_w, \quad \forall w \in W,$

$$x_p \geq 0, \quad \forall p \in P_w, w \in W,$$

where

$$D(x) = \sum_{(i,j)} D_{ij} \left( \sum_{\text{all paths } p \text{ containing } (i,j)} x_p \right)$$

and $x$ is the vector of path flows $x_p$. There is a potentially huge number of variables as well as constraints in this problem. However, by judiciously taking into account the special structure of the problem, the constraint set can be simplified and the number of variables can be reduced to a manageable size, using algorithms that will be discussed in Chapter 3.
There are several approaches to handle a large number of constraints. One possibility, which points the way to some major classes of algorithms, is to initially discard some of the constraints, solve the corresponding less constrained problem, and later selectively reintroduce constraints that seem to be violated at the optimum. In Chapters 3-5, we will discuss methods of this type in detail.

Another possibility is to replace constraints with penalties that assign high cost for their violation. In particular, we may replace problem (1.48) with

\[
\minimize f(x) + c \sum_{j=1}^{r} P(g_j(x))
\]

subject to \( x \in X \),

where \( P(\cdot) \) is a scalar penalty function satisfying \( P(u) = 0 \) if \( u \leq 0 \), and \( P(u) > 0 \) if \( u > 0 \), and \( c \) is a positive penalty parameter. For example, we may use the quadratic penalty

\[
P(u) = (\max\{0, u\})^2.
\]

An interesting alternative is to use

\[
P(u) = \max\{0, u\},
\]

in which case it can be shown that the optimal solutions of the penalized problem (1.48) and the original problem (1.57) coincide when \( c \) is sufficiently large. We discuss this possibility in the next section.

1.5 EXACT PENALTY FUNCTIONS

In this section, we discuss a transformation that is often useful in the context of constrained optimization algorithms. We will derive a form of equivalence between a constrained convex optimization problem, and a penalized problem that is less constrained or is entirely unconstrained. The motivation is that some convex optimization algorithms do not have constrained counterparts, but can be applied to a penalized unconstrained problem. Furthermore, in some analytical contexts, it is useful to be able to work with an equivalent problem that is less constrained.

We consider the problem

\[
\minimize f(x)
\]

subject to \( x \in X, \quad g(x) \leq 0, \quad i = 1, \ldots, r \),

where \( g(x) = \{g_1(x), \ldots, g_r(x)\} \), \( X \) is a convex subset of \( \mathbb{R}^n \), and \( f : \mathbb{R}^n \to \mathbb{R} \) and \( g_j : \mathbb{R}^n \to \mathbb{R} \) are real-valued convex functions. We denote
by $f^*$ the primal optimal value, and by $q^*$ the dual optimal value, i.e.,

$$q^* = \sup_{\mu \geq 0} q(\mu),$$

where

$$q(\mu) = \inf_{x \in X} \{ f(x) + \mu^t g(x) \}, \quad \forall \mu \geq 0.$$ 

We assume that $-\infty < q^*$ and $f^* < \infty$.

We introduce a convex penalty function $P : \mathbb{R}^r \mapsto \mathbb{R}$, which satisfies

$$P(u) = 0, \quad \forall \ u \leq 0, \quad (1.59)$$

$$P(u) > 0, \quad \text{if } u_i > 0 \text{ for some } i = 1, \ldots, r. \quad (1.60)$$

We consider solving in place of the original problem (1.58), the “penalized” problem

$$\text{minimize } f(x) + P(g(x))$$

subject to \( x \in X, \quad (1.61) \)

where the inequality contraints have been replaced by the extra cost $P(g(x))$ for their violation. Some interesting examples of penalty functions are

$$P(u) = \frac{c}{2} \sum_{i=1}^{r} (\max\{0, u_i\})^2,$$

and

$$P(u) = c \sum_{i=1}^{r} \max\{0, u_i\},$$

where $c$ is a positive penalty parameter.

The convex conjugate function of $P$ is given by

$$Q(\mu) = \sup_{u \in \mathbb{R}^r} \{ u^t \mu - P(u) \},$$

and it can be seen that

$$Q(\mu) \geq 0, \quad \forall \ \mu \in \mathbb{R}^r,$$

$$Q(\mu) = \infty, \quad \text{if } \mu_i < 0 \text{ for some } i = 1, \ldots, r.$$

Some interesting penalty functions $P$ are shown in Fig. 1.5.1, together with their conjugates.

Consider the primal function of the original constrained problem,

$$p(u) = \inf_{x \in X, g(x) \leq u} f(x), \quad u \in \mathbb{R}^r.$$
We have,

\[
\inf_{x \in X} \{ f(x) + P(g(x)) \} = \inf_{x \in X} \inf_{u \in \mathbb{R}} \{ f(x) + P(u) \}
\]

\[
= \inf_{x \in X} \inf_{u \in \mathbb{R}, g(x) \leq u} \{ f(x) + P(u) \}
\]

\[
= \inf_{x \in X, u \in \mathbb{R}, g(x) \leq u} \{ f(x) + P(u) \}
\]

\[
= \inf_{u \in \mathbb{R}, x \in X, g(x) \leq u} \{ f(x) + P(u) \}
\]

\[
= \inf_{u \in \mathbb{R}} \{ p(u) + P(u) \},
\]

where for the second equality, we use the monotonicity relation†

\[
u \leq v \quad \Rightarrow \quad P(u) \leq P(v).
\]

† To show this relation, we argue by contradiction: if there exist \( u \) and \( v \)
Moreover, \(-\infty < q^*\) and \(f^* < \infty\) by assumption, and since for any \(\mu\) with 
\[ q(\mu) > -\infty, \]
we have
\[ p(u) \geq q(\mu) - \mu' u > -\infty, \quad \forall u \in \mathbb{R}^r , \]
it follows that \(p(0) < \infty\) and \(p(u) > -\infty\) for all \(u \in \mathbb{R}^r\), so \(p\) is proper.

We can now apply the Fenchel Duality Theorem (Prop. 1.2.1) with the identifications
\[ f_1 = p, \quad f_2 = P, \quad A = I . \]
We use the conjugacy relation between the primal function \(p\) and the dual function \(q\) to write
\[
\inf_{u \in \mathbb{R}^r} \{ p(u) + P(u) \} = \sup_{\mu \geq 0} \{ q(\mu) - Q(\mu) \}, \tag{1.62}
\]
so that
\[
\inf_{x \in X} \{ f(x) + P(g(x)) \} = \sup_{\mu \geq 0} \{ q(\mu) - Q(\mu) \}; \tag{1.63}
\]
see Fig. 1.5.2. Note that the conditions for application of the theorem are satisfied since the penalty function \(P\) is real-valued, so that the relative interiors of \(\text{dom}(p)\) and \(\text{dom}(P)\) have nonempty intersection. Furthermore, as part of the conclusions of part (a) of the Fenchel Duality Theorem, it follows that the supremum over \(\mu \geq 0\) in Eq. (1.63) is attained.

Figure 1.5.2 suggests that in order for the penalized problem (1.61) to have the same optimal value as the original constrained problem (1.58), the conjugate \(Q\) must be “sufficiently flat” so that it is minimized by some dual optimal solution \(\mu^*\). This can be interpreted in terms of properties of subgradients, which are stated in Appendix B, and are developed in [Ber09]: we must have \(0 \in \partial Q(\mu^*)\) for some dual optimal solution \(\mu^*\), which by Prop. 5.4.3/App. B, is equivalent to \(\mu^* \in \partial P(0)\). This is part (a) of the following proposition from [Ber75a] (see also [BNO03]). Parts (b) and (c) of the proposition deal with issues of equality of corresponding optimal solutions. The proposition assumes the convexity and other assumptions made in the early part in this section regarding problem (1.58) and the penalty function \(P_\alpha\).

**Proposition 1.5.1:**

(a) The penalized problem (1.61) and the original constrained problem (1.58) have equal optimal values if and only if there exists a dual optimal solution \(\mu^*\) such that \(\mu^* \in \partial P(0)\).

with \(u \leq v\) and \(P(u) > P(v)\), there must exist \(\overline{u}\) close enough to \(u\) such that 
\[ \overline{u} < v \text{ and } P(\overline{u}) > P(v), \]
so that \(\lim_{\alpha \to \infty} P(v + \alpha(\overline{u} - v)) = \infty\), which contradicts 
Eq. (1.59), since \(v + \alpha(\overline{u} - v) < 0\) for sufficiently large \(\alpha\).
(b) In order for some optimal solution of the penalized problem (1.61) to be an optimal solution of the constrained problem (1.58), it is necessary that there exists a dual optimal solution \( \mu^* \) such that

\[
 u^T \mu^* \leq P(u), \quad \forall \ u \in \mathbb{R}^r. \tag{1.64}
\]

(c) In order for the penalized problem (1.61) and the constrained problem (1.58) to have the same set of optimal solutions, it is sufficient that there exists a dual optimal solution \( \mu^* \) such that

\[
 u^T \mu^* < P(u), \quad \forall \ u \in \mathbb{R}^r \text{ with } u_i > 0 \text{ for some } i. \tag{1.65}
\]
Proof: (a) We have using Eqs. (1.62) and (1.63),

\[ p(0) \geq \inf_{u \in \mathbb{R}^r} \{ p(u) + P(u) \} = \sup_{\mu \geq 0} \{ g(\mu) - Q(\mu) \} = \inf_{x \in X} \{ f(x) + P(g(x)) \}. \]

Since \( f^* = p(0) \), we have \( f^* = \inf_{x \in X} \{ f(x) + P(g(x)) \} \) if and only if equality holds in the above relation. This is true if and only if

\[ 0 \in \arg \min_{u \in \mathbb{R}^r} \{ p(u) + P(u) \}, \]

which by Prop. 5.4.7/App. B, is true if and only if there exists some \( \mu^* \in -\partial p(0) \) with \( \mu^* \in \partial P(0) \). Since the set of dual optimal solutions is \( -\partial p(0) \) (see Example 4.5.2, [Ber09]), the result follows.

(b) If \( x^* \) is an optimal solution of both problems (1.58) and (1.61), then by feasibility of \( x^* \), we have \( P(g(x^*)) = 0 \), so these two problems have equal optimal values. From part (a), there must exist a dual optimal solution \( \mu^* \in \partial P(0) \), which is equivalent to Eq. (1.64), by the subgradient inequality.

(c) If \( x^* \) is an optimal solution of the constrained problem (1.58), then \( P(g(x^*)) = 0 \), so we have

\[ f^* = f(x^*) = f(x^*) + P(g(x^*)) \geq \inf_{x \in X} \{ f(x) + P(g(x)) \}. \]

The condition (1.65) implies the condition (1.64), so that by part (a), equality holds throughout in the above relation, showing that \( x^* \) is also an optimal solution of the penalized problem (1.61).

Conversely, if \( x^* \) is an optimal solution of the penalized problem (1.61), then \( x^* \) is either feasible [satisfies \( g(x^*) \leq 0 \)], in which case it is an optimal solution of the constrained problem (1.58) [in view of \( P(g(x)) = 0 \) for all feasible vectors \( x \)], or it is infeasible in which case \( g_j(x^*) > 0 \) for some \( j \). In the latter case, by using the given condition (1.65), it follows that there exists an \( \epsilon > 0 \) such that

\[ \mu^* g(x^*) + \epsilon < P(g(x^*)). \]

Let \( \tilde{x} \) be a feasible vector such that \( f(\tilde{x}) \leq f^* + \epsilon \). Since \( P(g(\tilde{x})) = 0 \) and \( f^* = \min_{x \in X} \{ f(x) + \mu^* g(x) \} \), we obtain

\[ f(\tilde{x}) + P(g(\tilde{x})) = f(\tilde{x}) \leq f^* + \epsilon \leq f(x^*) + \mu^* g(x^*) + \epsilon. \]

By combining the last two equations, we obtain

\[ f(\tilde{x}) + P(g(\tilde{x})) < f(x^*) + P(g(x^*)), \]

which contradicts the hypothesis that \( x^* \) is an optimal solution of the penalized problem (1.61). This completes the proof. Q.E.D.
Note that in the case where the necessary condition (1.64) holds but the sufficient condition (1.65) does not, it is possible that the constrained problem (1.58) has optimal solutions that are not optimal solutions of the penalized problem (1.61), even though the two problems have the same optimal value.

To elaborate on Prop. 1.5.1, consider the penalty function

\[ P(u) = c \sum_{i=1}^{r} \max\{0, u_i\}, \]

where \( c > 0 \). The condition \( \mu^* \in \partial P(0) \), or equivalently, \( u' \mu^* \leq P(u) \) for all \( u \in \mathbb{R}^r \) [cf. Eq. (1.64)], is equivalent to

\[ \mu^*_i \leq c, \quad \forall \ i = 1, \ldots, r. \]

Similarly, the condition \( u' \mu^* < P(u) \) for all \( u \in \mathbb{R}^r \) with \( u_i > 0 \) for some \( j \) [cf. Eq. (1.65)], is equivalent to

\[ \mu^*_i < c, \quad \forall \ i = 1, \ldots, r. \]

A General Exact Penalty Function

Let us finally discuss the case of a general Lipschitz continuous (not necessarily convex) cost function and an abstract constraint set \( X \subset \mathbb{R}^n \). The idea is to use a penalty that is proportional to the distance from \( X \):

\[ \text{dist}(x; X) = \inf_{y \in X} \|x - y\|. \]

We have the following proposition.

\textbf{Proposition 1.5.2:} Let \( f : Y \mapsto \mathbb{R} \) be a function defined on a subset \( Y \) of \( \mathbb{R}^n \). Assume that \( f \) is Lipschitz continuous with constant \( L \), i.e.,

\[ |f(x) - f(y)| \leq L \|x - y\|, \quad \forall \ x, y \in Y. \quad (1.66) \]

Let also \( X \) be a nonempty closed subset of \( Y \), and \( c \) be a scalar with \( c > L \). Then \( x^* \) minimizes \( f \) over \( X \) if and only if \( x^* \) minimizes

\[ F_c(x) = f(x) + c \text{dist}(x; X) \]

over \( Y \).
Proof: For a vector \( x \in Y \), let \( \hat{x} \) denote a vector of \( X \) that is at minimum distance from \( x \). We have for all \( x \in Y \),

\[
F_c(x) = f(x) + c\|x - \hat{x}\| = f(\hat{x}) + (f(x) - f(\hat{x})) + c\|x - \hat{x}\| \geq f(\hat{x}) = F_c(\hat{x}),
\]

with strict inequality if \( x \neq \hat{x} \) [the inequality follows using the assumption (1.66)]. Hence minima of \( F_c \) over \( Y \) can only lie within \( X \), while \( F_c = f \) within \( X \). This shows that \( x^* \) minimizes \( f \) over \( X \) if and only if \( x^* \) minimizes \( F_c \) over \( Y \). Q.E.D.

The following proposition from [Ber11] provides a generalization.

**Proposition 1.5.3:** Let \( f : Y \mapsto \mathbb{R} \) be a function defined on a subset \( Y \) of \( \mathbb{R}^n \), and let \( X_i, i = 1, \ldots, m \), be closed subsets of \( Y \) with nonempty intersection. Assume that \( f \) is Lipschitz continuous over \( Y \). Then there is a scalar \( \bar{c} > 0 \) such that for all \( c \geq \bar{c} \), the set of minima of \( f \) over \( \cap_{i=1}^m X_i \) coincides with the set of minima of

\[
f(x) + c \sum_{i=1}^m \text{dist}(x; X_i)
\]

over \( Y \).

Proof: Let \( L \) be the Lipschitz constant for \( f \), and let \( c_1, \ldots, c_m \) be scalars satisfying

\[
c_k > L + c_1 + \cdots + c_{k-1}, \quad \forall \ k = 1, \ldots, m,
\]

where \( c_0 = 0 \). Define

\[H_k(x) = f(x) + c_1 \text{dist}(x; X_1) + \cdots + c_k \text{dist}(x; X_k), \quad k = 1, \ldots, m,
\]

and for \( k = 0 \), denote \( H_0(x) = f(x), c_0 = 0 \). By applying Prop. 1.5.2, the set of minima of \( H_m \) over \( Y \) coincides with the set of minima of \( H_{m-1} \) over \( X_m \), since \( c_m \) is greater than \( L + c_1 + \cdots + c_{m-1} \), the Lipschitz constant for \( H_{m-1} \). Similarly, for all \( k = 1, \ldots, m \), the set of minima of \( H_k \) over \( \cap_{i=k+1}^m X_i \) coincides with the set of minima of \( H_{k-1} \) over \( \cap_{i=k}^m X_i \). Thus, for \( k = 1 \), we obtain that the set of minima of

\[
f + c \sum_{i=1}^m \text{dist}(\cdot; X_i) = H_m
\]

over \( Y \) coincides with the set of minima of \( f = H_0 \) over \( \cap_{i=1}^m X_i \). Q.E.D.
Sec. 1.6 Iterative Descent Algorithms

1.6 ITERATIVE DESCENT ALGORITHMS

In this section and the next one we provide an overview of the ideas underlying iterative algorithms for convex optimization. These are algorithms that produce sequences \( \{x_k\} \) according to

\[
x_{k+1} = G_k(x_k),
\]

where \( G_k : \mathbb{R}^n \rightarrow \mathbb{R}^n \) is some function that may depend on \( k \), and \( x_0 \) is some starting point. We are typically interested in the convergence of the generated sequence \( \{x_k\} \) to some desirable point \( x^* \). We are also interested in questions of rate of convergence, such as for example the number of iterations needed to bring a measure of error to within a given tolerance. We will treat these issues somewhat informally in this section, and in the subsequent chapters, we will introduce more formal notions of convergence and rate of convergence analysis.

A common type of iterative algorithm is when \( G_k \) does not depend on \( k \). In this case we obtain the stationary iteration

\[
x_{k+1} = G(x_k),
\]

which aims to solve a fixed point problem: finding a solution of the equation \( x = G(x) \). A simple example is the iteration

\[
x_{k+1} = x_k - \alpha(Ax_k - b),
\]

that aims at solution of the linear system \( Ax = b \), or a gradient iteration

\[
x_{k+1} = x_k - \alpha \nabla f(x_k),
\]

that aims at satisfying the optimality condition \( \nabla f(x) = 0 \) for minimization of a differentiable function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \). In these equations, \( \alpha \) is a stepsize parameter that is used to ensure that the iterations make progress towards the solution set of the corresponding problem.

A common criterion for convergence of the stationary iteration \( x_{k+1} = G(x_k) \) is that \( G \) is a contraction mapping with respect to some norm, i.e., for some \( \rho < 1 \), and some norm \( \| \cdot \| \) (not necessarily the Euclidean norm), we have

\[
\| G(x) - G(y) \| \leq \rho \| x - y \|, \quad \forall x, y \in \mathbb{R}^n.
\]

Then it can be shown that \( G \) has a unique fixed point \( x^* \), and \( x_k \rightarrow x^* \), starting from any \( x_0 \in \mathbb{R}^n \); this is the well-known Banach fixed point theorem (see Section A.4 of Appendix A, where the contraction and other approaches for convergence analysis are discussed).

The case where \( G \) is a contraction mapping provides an example of convergence analysis based on a descent approach: at each iteration we have

\[
\| x_{k+1} - x^* \| \leq \rho \| x_k - x^* \|,
\]

(1.67)
so the function \( \phi(x) = \|x - x^*\| \) is decreased with each iteration at a nonsolution point \( x \). Moreover, in this case we obtain a rate of convergence estimate: \( \phi(x_k) \) is decreased at least as fast as the geometric progression \( \{ \rho^k \|x_0 - x^*\| \} \) (this is called \textit{linear} or \textit{geometric} convergence). There are also other types of fixed point iterations, which do not require that \( G \) is a contraction mapping, and where the convergence analysis is based on improvement of an appropriate figure of merit. For example there are cases where \( G \) is a nonexpansive mapping \([\rho = 1 \text{ in Eq. (1.67)}]\), and there is sufficient structure in \( G \) to ensure a suitable form of improvement at each iteration (see Chapter 4).

Many algorithms for optimization of a convex function \( f \) involve a contraction mapping as described above. However, there are also many cases of nonstationary iterations of the form

\[ x_{k+1} = G_k(x_k), \]

whose convergence analysis is difficult or impossible with a contraction mapping approach. An example is unconstrained minimization of a differentiable function \( f \) with a gradient method of the form

\[ x_{k+1} = x_k - \alpha_k \nabla f(x_k), \tag{1.68} \]

where the stepsize \( \alpha_k \) is not constant. Still many of these algorithms admit a convergence analysis based on a descent approach, whereby we introduce a function \( \phi \) that measures the progress of the algorithm towards optimality, and show that

\[ \phi(x_{k+1}) < \phi(x_k) \quad \text{if and only if} \quad x_k \text{ is not optimal}. \]

Two common cases are when \( \phi(x) = f(x) \) or \( \phi(x) = \text{dist}(x, X^*) \), the Euclidean minimum distance of \( x \) from the set \( X^* \) of minima of \( f \). For example convergence of the gradient algorithm (1.68) is often proved by showing that for all \( k \),

\[ f(x_{k+1}) \leq f(x_k) - \gamma_k \|\nabla f(x_k)\|^2, \]

where \( \gamma_k \) is a positive scalar that depends on \( \alpha_k \) and some characteristics of \( f \), and is such that \( \sum_{k=0}^{\infty} \gamma_k = \infty \); this guarantees that either \( \nabla f(x_k) \to 0 \) or \( f(x_k) \to -\infty \) (see e.g., [Ber99], Section 1.2).

In what follows in this section we will provide an overview of convex optimization algorithms that rely on some form of descent for their validity, we discuss some of their underlying motivation, and we raise various issues that will be discussed later. Our focus in subsequent chapters will be on the case of a nondifferentiable cost function, while in the overview of the present chapter we discuss in greater detail the differentiable cost function case and the potential benefits of differentiability. An issue that we will not discuss is the distributed asynchronous implementation of such algorithms, which is particularly relevant in large-scale contexts. We refer to [BeT89] for a detailed textbook development, including the case of distributed gradient methods (Chs. 6 and 7), and to the paper [NBB01] for the case of asynchronous incremental subgradient methods.
1.6.1 Cost Function Descent - Gradient Projection

A natural iterative descent approach to minimizing a real-valued convex function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) over a set \( X \) is based on cost improvement: starting with a point \( x_0 \in X \), construct a sequence \( \{x_k\} \subset X \) such that

\[
f(x_{k+1}) < f(x_k), \quad k = 0, 1, \ldots,
\]

unless \( x_k \) is optimal for some \( k \), at which time the method stops.

In the unconstrained case where \( X = \mathbb{R}^n \), it is natural to consider the directional derivative of \( f \) at a point \( x \) in a direction \( d \)

\[
f'(x; d) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha d) - f(x)}{\alpha}
\]

(cf. Section 5.4.4 of Appendix B). From this formula it follows that if \( f'(x; d) < 0 \), we may reduce the cost by moving from \( x \) along \( d \) with a small enough positive stepsize \( \alpha \). This leads to an algorithm of the form

\[
x_{k+1} = x_k + \alpha_k d_k,
\]

where \( \alpha_k \) is a positive scalar stepsize, and \( d_k \) is a descent direction at \( x_k \), in the sense that

\[
f'(x_k; d_k) < 0.
\]

If no descent direction can be found at \( x_k \), then \( x_k \) is optimal [this is because convexity implies that the ratio \( (f(x + \alpha d) - f(x))/\alpha \) is monotonically nondecreasing with \( \alpha \)].

Gradient Methods for Differentiable Unconstrained Minimization

For the case where \( f \) is differentiable and \( X = \mathbb{R}^n \), there are many popular descent algorithms of the form (1.69). An important example is the classical gradient method, where we use \( d_k = -\nabla f(x_k) \) in Eq. (1.69):

\[
x_{k+1} = x_k - \alpha_k \nabla f(x_k).
\]

Since for differentiable \( f \) we have

\[
f'(x_k; d) = \nabla f(x_k)'d,
\]

it follows that

\[
- \frac{\nabla f(x_k)}{\|\nabla f(x_k)\|} = \arg \min_{\|d\| \leq 1} f'(x_k; d)
\]

[assuming \( \nabla f(x_k) \neq 0 \)]. Thus the gradient method is the descent algorithm of the form (1.69) that uses the direction with greatest rate of cost improvement. For this reason it is also called the method of steepest descent.
With proper stepsize choice, the convergence rate of the steepest descent method is geometric, assuming that it generates a sequence \( \{x_k\} \) that converges to a vector \( x^* \) such that \( \nabla f(x^*) = 0 \) and \( \nabla^2 f(x^*) \) is positive definite. For example, if \( \alpha_k \) is a sufficiently small constant \( \alpha > 0 \), the corresponding iteration

\[
x_{k+1} = x_k - \alpha \nabla f(x_k), \tag{1.71}
\]

is contractive,† so it converges at a geometric rate. Still, even in this case the convergence rate can be very slow, because the corresponding contraction parameter can be very close to 1.

To improve the convergence rate of the steepest descent method one may “scale” the gradient \( \nabla f(x_k) \) by multiplication with a positive definite symmetric matrix \( S_k \), i.e., \( d_k = -S_k \nabla f(x_k) \), leading to the algorithm

\[
x_{k+1} = x_k - \alpha_k S_k \nabla f(x_k); \tag{1.72}
\]

cf. Fig. 1.6.1. Since for \( \nabla f(x_k) \neq 0 \) we have

\[
f'(x_k; d_k) = -\nabla f(x_k)' S_k \nabla f(x_k) < 0,
\]

it follows that we still have a cost descent method, as long as the positive stepsize \( \alpha_k \) is sufficiently small so that \( f(x_{k+1}) < f(x_k) \).

Much of the theory of unconstrained nonlinear programming algorithms deals with ways to compute “good” scaling matrices \( S_k \), i.e., matrices that result in fast convergence rate. The “best” scaling in this sense is attained with

\[
S_k = (\nabla^2 f(x_k))^{-1},
\]

† To get a sense of this, assume for convenience that \( f \) is quadratic, so by adding a suitable constant to \( f \), we have

\[
f(x) = \frac{1}{2} (x - x^*)' Q (x - x^*), \quad \nabla f(x) = Q (x - x^*),
\]

where \( Q \) is the positive definite symmetric Hessian of \( f \). Then for a constant stepsize \( \alpha \), the steepest descent iteration (1.71) can be written as

\[
x_{k+1} - x^* = (I - \alpha Q)(x_k - x^*).
\]

For \( \alpha \) less than twice the smallest eigenvalue of \( Q \), the matrix \( I - \alpha Q \) has eigenvalues strictly within the unit circle, and is a contraction with respect to the Euclidean norm. The modulus of contraction may be estimated in terms of the condition number of \( Q \), the ratio of largest to smallest eigenvalue. As the condition number increases to \( \infty \) the modulus of contraction approaches 1, for all values of \( \alpha \) that lead to convergence (see e.g., [Ber99], Section 1.3).
Sec. 1.6 Iterative Descent Algorithms

\[ x_k - \alpha \nabla f(x_k) \]

\[ x_k - \alpha S_k \nabla f(x_k) \]

Figure 1.6.1. Illustration of descent directions. Any direction of the form

\[ d_k = -S_k \nabla f(x_k), \]

where \( S_k \) is positive definite, makes an angle less than \( \pi/2 \) with the steepest descent direction \(-\nabla f(x_k)\), and is a descent direction.

assuming that the inverse above exists and is positive definite. This is Newton’s method, which will be discussed shortly. A simpler alternative is to use a diagonal approximation to the Hessian matrix \( \nabla^2 f(x_k) \), i.e., the diagonal matrix \( S_k \) that has the inverse second partial derivatives

\[ \left( \frac{\partial^2 f(x_k)}{\partial x^i \partial x^j} \right)^{-1}, \quad i = 1, \ldots, n, \]

along the diagonal. This often improves the performance of the classical gradient method dramatically, by providing automatic scaling of the units in which the components \( x^i \) of \( x \) are measured, and also facilitates the choice of stepsize - good values of \( \alpha_k \) are typically chose to 1 (see the subsequent discussion of Newton’s method and sources such as [Ber99], Section 2.3).

The nonlinear programming methodology also prominently includes quasi-Newton methods, which construct scaling matrices iteratively, using gradient information collected during the algorithmic process. Some of these methods approximate the full inverse Hessian of \( f \), and eventually attain the fast convergence rate of Newton’s method, while others use a limited amount of gradient information (have “limited memory”) but still attain a convergence rate that is considerably faster than the one of the unscaled gradient method; see nonlinear programming texts such as [Ber99], [Lue84], [NoW99].

There are also many methods to choose \( \alpha_k \) in iteration (1.72), for which we refer to nonlinear programming textbooks such as [Ber99], [Lue84],
and [NoW06]. For example, $\alpha_k$ may be chosen by line minimization:

$$\alpha_k \in \arg\min_{\alpha \geq 0} f(x_k + \alpha d_k).$$

In this book, we will focus primarily in two cases: when $\alpha_k$ is chosen to be constant,

$$\alpha_k = \alpha, \quad k = 0, 1, \ldots,$$

and when $\alpha_k$ is chosen to be diminishing to 0, while satisfying the conditions:

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty. \quad (1.73)$$

A convergence analysis for these two stepsize rules will be given in Chapter 2, in the context of subgradient methods, and also in Section 5.1. Another simple stepsize rule, which uses a test for sequential reduction of $\alpha_k$, is given in the exercises.

**Newton’s Method**

In Newton’s method the descent direction is

$$d_k = - (\nabla^2 f(x_k))^{-1} \nabla f(x_k),$$

provided $\nabla^2 f(x_k)$ exists and is positive definite, so the iteration takes the form

$$x_{k+1} = x_k - \alpha_k (\nabla^2 f(x_k))^{-1} \nabla f(x_k). \quad (1.74)$$

If $\nabla^2 f(x_k)$ is not positive definite, some modification is necessary. There are several possible modifications of this type, for which the reader may consult nonlinear programming textbooks. The simplest one is to add to $\nabla^2 f(x_k)$ a small positive multiple of the identity. Generally, when $f$ is convex, $\nabla^2 f(x_k)$ is positive semidefinite (Prop. 1.1.10/App. B), and this facilitates the implementation of reliable Newton-type algorithms.

The idea in Newton’s method is to minimize at each iteration the quadratic approximation of $f$ around the current point $x_k$ given by

$$\tilde{f}_k(x) = f(x_k) + \nabla f(x_k)'(x - x_k) + \frac{1}{2}(x - x_k)' \nabla^2 f(x_k)(x - x_k).$$

† The condition $\sum_{k=0}^{\infty} \alpha_k = \infty$ is needed so that the method can approach the minimum from arbitrarily far, and the condition $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$ is needed so that $\alpha_k \to 0$ and also for technical reasons relating to the convergence analysis (see Chapter 2). If $f$ is positive definite quadratic, the steepest descent method with a diminishing stepsize $\alpha_k$ satisfying $\sum_{k=0}^{\infty} \alpha_k = \infty$ can be shown to converge to the optimal solution, but at a rate that is slower than geometric (see Prop. A.4.2 in Appendix A).
By setting the gradient of \( \tilde{f}_k(x) \) to zero,
\[
\nabla f(x_k) + \nabla^2 f(x_k)(x - x_k) = 0,
\]
and solving for \( x \), we obtain as next iterate the minimizing point
\[
x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1}\nabla f(x_k).
\] (1.75)
This is the Newton iteration corresponding to a stepsize \( \alpha_k = 1 \). It follows that Newton’s method finds the global minimum of a positive definite quadratic function in a single iteration (assuming \( \alpha_k = 1 \)).

Newton’s method typically converges very fast asymptotically, assuming that it converges to a vector \( x^* \) such that \( \nabla f(x^*) = 0 \) and \( \nabla^2 f(x^*) \) is positive definite, and that a stepsize \( \alpha_k = 1 \) is used, at least after some iteration. For a simple argument, we may use Taylor’s theorem to write
\[
0 = \nabla f(x^*) = \nabla f(x_k) + \nabla^2 f(x_k)(x^* - x_k) + o(\|x_k - x^*\|).
\]
By multiplying this relation with \( (\nabla^2 f(x_k))^{-1} \) we have
\[
x_k - x^* - (\nabla^2 f(x_k))^{-1}\nabla f(x_k) = o(\|x_k - x^*\|),
\]
so for the Newton iteration with stepsize \( \alpha_k = 1 \) we obtain
\[
x_{k+1} - x^* = o(\|x_k - x^*\|),
\]
or, for \( x_k \neq x^* \),
\[
\lim_{k \to \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = \lim_{k \to \infty} \frac{o(\|x_k - x^*\|)}{\|x_k - x^*\|} = 0,
\]
implying convergence that is faster than geometric (also called superlinear). This argument can also be used to show local convergence to \( x^* \) with \( \alpha_k \equiv 1 \), that is, convergence assuming that \( x_0 \) is sufficiently close to \( x^* \).

In implementations of Newton’s method, some stepsize rule is often used to ensure cost reduction, but the rule is typically designed so that near convergence we have \( \alpha_k = 1 \), to ensure that a superlinear convergence rate is attained [assuming \( \nabla^2 f(x^*) \) is positive definite at the limit \( x^* \)]. Methods that approximate Newton’s method also use a stepsize close to 1, and modify the stepsize based on the results of the computation (see sources on nonlinear programming, such as [Ber99], Section 1.4).

The price for the fast speed of convergence of Newton’s method is the overhead required to calculate the Hessian matrix, and to solve the linear system of equations
\[
\nabla^2 f(x_k)d_k = -\nabla f(x_k)
\]
in order to find the Newton direction. There are many iterative algorithms that are patterned after Newton’s method, and aim to strike a balance between fast convergence and high overhead (e.g., quasi-Newton, conjugate direction, and others, extensive discussions of which may be found in nonlinear programming textbooks such as [Ber99], [Lue84], and [NoW06]).

We finally note that for some problems the special structure of the Hessian matrix can be exploited to facilitate the implementation of Newton’s method. For example the Hessian matrix of the dual function of the separable convex programming problem of Section 1.1, when it exists, has particularly favorable structure; see [Ber99], Section 6.1.

**Feasible Direction Methods - Gradient Projection**

In the constrained case where $X$ is a closed convex subset of $\mathbb{R}^n$, we may consider generating a feasible sequence $\{x_k\} \subset X$ with an iteration of the form

$$x_{k+1} = x_k + \alpha_k d_k,$$

while enforcing cost function descent. However, this becomes more complicated because it is not enough for $d_k$ to be a descent direction at $x_k$. It must also be a feasible direction in the sense that $x_k + \alpha d_k$ must belong to $X$ for small enough $\alpha > 0$, in order for the new iterate $x_{k+1}$ to belong to $X$ with suitably small choice of $\alpha_k$. This essentially restricts $d_k$ to be of the form $\bar{x}_k - x_k$ for some $\bar{x}_k \in X$ with $\bar{x}_k \neq x_k$. Thus, if $f$ is differentiable, for a feasible descent direction, we must have

$$d_k = \bar{x}_k - x_k, \quad \text{with } \bar{x}_k \in X, \quad \nabla f(x_k)'(\bar{x}_k - x_k) < 0.$$

One the simplest possibilities is to find

$$\bar{x}_k \in \arg\min_{x \in X} \nabla f(x_k)'(x - x_k),$$

and set

$$d_k = \bar{x}_k - x_k.$$

Clearly $\nabla f(x_k)'(\bar{x}_k - x_k) \leq 0$, with equality only if $\nabla f(x_k)'(x - x_k) \geq 0$ for all $x \in X$, in which case $x_k$ must be optimal (cf. Prop. 1.1.8/App. B). This is the conditional gradient method, which has been used widely in many contexts, as it is theoretically sound, quite simple, and often convenient. In particular, when $X$ is a polyhedral set, computation of $\bar{x}_k$ requires the solution of a linear program. In some important cases, this linear program has special structure, which results in great simplifications, e.g., in the multicommodity flow problem of Example 1.4.4 (see [BeG92]).

However, the conditional gradient method often tends to converge very slowly; see the discussion and analysis of [Ber99], Section 2.2.2. For this reason, other methods with better convergence rate properties are often
Sec. 1.6  Iterative Descent Algorithms  51

Figure 1.6.2. Illustration of the gradient projection iteration at $x_k$. We move from $x_k$ along the direction $-\nabla f(x_k)$ and project $x_k - \alpha \nabla f(x_k)$ onto $X$ to obtain $x_{k+1}$. We have

$$
\nabla f(x_k)'(x_{k+1} - x_k) \leq 0,
$$

and unless $x_{k+1} = x_k$, in which case $x_k$ minimizes $f$ over $X$, the angle between $\nabla f(x_k)$ and $(x_{k+1} - x_k)$ is strictly greater than 90 degrees, in which case

$$
\nabla f(x_k)'(x_{k+1} - x_k) < 0.
$$

preferred. One of the most popular ones is the gradient projection method, which has the form

$$
x_{k+1} = P_X(x_k - \alpha_k \nabla f(x_k)), \tag{1.76}
$$

where $\alpha_k > 0$ is a stepsize and $P_X(\cdot)$ denotes projection on $X$ (the projection is well defined since $X$ is closed and convex; see Fig. 1.6.2).

Indeed, from the projection theorem (Prop. 1.1.9/App. B), we have

$$
\nabla f(x_k)'(x_{k+1} - x_k) \leq 0,
$$

and by the optimality condition for convex functions (cf. Prop. 1.1.8/App. B), the inequality is strict unless $x_k$ is optimal. Thus $x_{k+1} - x_k$ defines a feasible descent direction at $x_k$, and based on this fact, we can show the descent property $f(x_{k+1}) < f(x_k)$ when $\alpha_k$ is sufficiently small. The stepsize $\alpha_k$ is chosen similar to the unconstrained gradient method, i.e., constant, diminishing, or through some kind of reduction rule to ensure cost function descent and guarantee convergence to the optimum; see Chapter 2, Section 5.1, and also [Ber99], Section 2.3, for a detailed discussion.

Two-Metric Projection Methods

Despite its simplicity, the gradient projection method has some significant drawbacks:

(a) Its rate of convergence is similar to the one of steepest descent, and is often slow. It is possible to overcome this potential drawback by a form of scaling. This can be accomplished with an iteration of the form

$$
x_{k+1} = \arg \min_{x \in X} \left\{ \nabla f(x_k)'(x - x_k) + \frac{1}{2\alpha_k}(x - x_k)'H_k(x - x_k) \right\}, \tag{1.77}
$$

where $H_k$ is a positive definite matrix.
where $H_k$ is a positive definite symmetric matrix and $\alpha_k$ is a positive stepsize. When $H_k$ is the identity, it can be seen that this iteration gives the same iterate $x_{k+1}$ as the unscaled gradient projection iteration (1.76). When $H_k = \nabla^2 f(x_k)$ and $\alpha_k = 1$, we obtain a constrained form of Newton’s method (see nonlinear programming textbooks such as [Ber99] for analysis).

(b) Depending on the nature of $X$, the projection operation may involve substantial overhead. The projection is simple when $H_k$ is the identity (or more generally, is diagonal), and $X$ consists of simple lower and/or upper bounds on the components of $x$:

$$X = \{(x^1, \ldots, x^n) \mid \underline{b}^i \leq x^i \leq \bar{b}^i, \ i = 1, \ldots, n\}. \quad (1.78)$$

This is an important special case where the use of gradient projection is convenient. Then the projection decomposes to $n$ scalar projections, one for each $i = 1, \ldots, n$: the $i$th component of $x_{k+1}$ is obtained by projection of the $i$th component of $x_k - \alpha_k \nabla f(x_k)$, 

$$\left(x_k - \alpha_k \nabla f(x_k)\right)^i,$$

onto the interval of corresponding bounds $[\underline{b}^i, \bar{b}^i]$, and is very simple. However, for general nondiagonal scaling the overhead for solving the quadratic programming problem (1.77) is substantial even if $X$ has a simple bound structure of Eq. (1.78).

To overcome the difficulty with the projection overhead, a scaled projection method known as two-metric projection method has been proposed for the case of the bound constraints (1.78) in [Ber82a], [Ber82b]. It has a similar form to the scaled gradient method (1.72), and it is given by

$$x_{k+1} = P_X(x_k - \alpha_k S_k \nabla f(x_k)). \quad (1.79)$$

It is thus a natural and simple adaptation of unconstrained Newton-like methods to bound-constrained optimization. The main difficulty here is that an arbitrary positive definite matrix $S_k$ will not necessarily yield a descent direction. However, it turns out that if some of the off-diagonal terms of $S_k$ that correspond to components of $x_k$ that are at their boundary are set to zero, one can obtain descent (see the exercises). Furthermore, one can select $S_k$ as the inverse of a partially diagonalized version of the Hessian matrix $\nabla^2 f(x_k)$ and attain the fast convergence rate of Newton’s method (see [Ber82a], [Ber82b], [GaB84]).

The idea of simple two-metric projection with partial diagonalization may be generalized to more complex constraint sets, and it has been adapted in [Ber82a] to problems of the form

$$\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad \underline{b} \leq x \leq \bar{b}, \ Ax = c,
\end{align*}$$
where $A$ is an $m \times n$ matrix, and $\tilde{b}, \tilde{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}^m$ are given vectors. For example the algorithm (1.79) can be easily modified when the constraint set involves bounds on the components of $x$ together with a few linear constraints, e.g., problems involving a simplex constraint such as

$$\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad \underline{b} \leq x \leq \bar{b}, \quad a'x = c,
\end{align*}$$

where $a \in \mathbb{R}^n$ and $c \in \mathbb{R}$, or a Cartesian product of simplexes. For an example of such an algorithm applied to the multicommodity flow problem of Example 1.4.4, see [BeG83]. Applications in large-scale applications have also been considered in [Dun91], [Pty98].

The advantage that the two-metric projection method can offer is to identify quickly the constraints that are active at an optimal solution. After this happens, the method reduces essentially to an unconstrained scaled gradient method (possibly Newton method, if $S_k$ is a partially diagonalized Hessian matrix), and attains a fast convergence rate. This property has also motivated variants of the two-metric projection method for problems involving $\ell_1$-regularization, such as the ones of Example 1.3.1 (see [SKS12]).

**Coordinate Descent**

The preceding methods require the computation of the gradient and possibly the Hessian of the cost function at each iterate. An alternative descent approach that does not require derivatives is the classical coordinate descent method, which we will briefly describe here and consider further in Chapter 5. The method applies to the problem

$$\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in X,
\end{align*}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a differentiable convex function, and $X$ is a Cartesian product of closed convex sets $X_1, \ldots, X_m$:

$$X = X_1 \times X_2 \times \cdots \times X_m. \quad (1.80)$$

The vector $x$ is partitioned as

$$x = (x^1, x^2, \ldots, x^m),$$

where each $x^i$ belongs to $\mathbb{R}^{n_i}$, so the constraint $x \in X$ is equivalent to

$$x^i \in X_i, \quad i = 1, \ldots, m.$$  

The most common case is when $n_i = 1$ for all $i$, so the components $x^i$ are scalars. The method involves minimization with respect to a single component $x^i$ at each iteration, with all other components kept fixed.
In an example of such a method, given the current iterate \( x_k = (x^1_k, \ldots, x^m_k) \), the block coordinate descent method, generates the next iterate \( x_{k+1} = (x^1_{k+1}, \ldots, x^m_{k+1}) \), according to the “cyclic” iteration

\[
x^i_{k+1} \in \arg \min_{\xi \in X_i} f(x^1_{k+1}, \ldots, x^{i-1}_{k+1}, \xi, x^{i+1}_{k}, \ldots, x^m_k), \quad i = 1, \ldots, m. \tag{1.81}
\]

Thus, at each iteration, the cost is minimized with respect to each of the “block coordinate” vectors \( x^i_k \), taken one-at-a-time in cyclic order.

Naturally, the method makes practical sense only if it is possible to perform this minimization fairly easily. This is frequently so when each \( x^i \) is a scalar, but there are also other cases of interest, where \( x^i \) is a multidimensional vector. Moreover, the method can take advantage of special structure of \( f \); an example of such structure is a form of “sparsity,” where \( f \) is the sum of component functions, and for each \( i \), only a relatively small number of the component functions depend on \( x^i \), thereby simplifying the minimization (1.81).

In the absence of special structure of \( f \), differentiability is essential for the validity of the method; this can be verified with simple examples. However, there are some interesting cases where nondifferentials with special structure can be dealt with; see Chapter 5. In our convergence analysis of Chapter 5, we will also require a form of strict convexity of \( f \) along each block component (subtle examples of nonconvergence have been constructed in the absence of a property of this kind [Pow73]).

An interesting variant of the method is one where the block components are iterated in an irregular order instead of a fixed cyclic order. In fact there is a substantial theory of asynchronous distributed versions of coordinate descent, for which we refer to the parallel and distributed algorithms book [BeT89], and the sources quoted there; see also the discussion in Chapter 5.

Nondifferentiable Cost Functions - Subgradient Methods

It is possible to generalize the steepest descent approach so that when \( f \) is nondifferentiable at \( x_k \), we use a direction \( d_k \) that minimizes \( f'(x_k; d) \) subject to \( \|d\| \leq 1 \); the implementation of this method is discussed in the exercises. Unfortunately, however, there is a major theoretical difficulty: the method may get stuck far from the optimum, depending on the stepsize rule. An example is given in Fig. 1.6.3. In this example, the algorithm fails even though it never encounters a point where \( f \) is nondifferentiable! The problem here is that the steepest descent direction may undergo a large/discontinuous change close to the convergence limit. By contrast, this would not happen if \( f \) were differentiable at the limit, and in fact the steepest descent method has sound convergence properties when used for minimization of differentiable functions, as we will see in Section 5.1.
Sec. 1.6 Iterative Descent Algorithms

Figure 1.6.3. An example of failure of the steepest descent method [Wol75]. Here we have the two-dimensional cost function
\[
f(x_1, x_2) = \begin{cases} 
5(9x_1^2 + 16x_2^2)^{1/2} & \text{if } x_1 > |x_2|, \\
9x_1 + 16|x_2| & \text{if } x_1 \leq |x_2|,
\end{cases}
\]
shown in the figure. Consider the method that moves in the direction of steepest descent from the current point, with the stepsize determined by cost minimization along that direction (this can be done analytically). Suppose that the algorithm starts anywhere within the set
\[
\{(x_1, x_2) \mid x_1 > |x_2| > (9/16)^2|x_1|\}.
\]
The generated iterates are shown in the figure, and it can be verified that they converge to the nonoptimal point (0, 0).

Because the cost descent approach has the theoretical limitations outlined above and also because minimization of \(f'(x_k; d)\) subject to \(\|d\| \leq 1\) (or more generally finding a descent direction) may involve a nontrivial computation, a different kind of descent approach is often used when \(f\) is nondifferentiable. This approach is based on the notion of subgradient of an extended real-valued convex function. Section 5.4 of Appendix B outlines the theory of subgradients, as developed in [Ber09]. The properties of subgradients of real-valued convex functions will also be discussed in detail in Section 2.1.

In the subgradient method an arbitrary subgradient \(g_k\) of \(f\) at \(x_k\) is used in an iteration of the form
\[
x_{k+1} = x_k - \alpha_k g_k,
\]
where \(\alpha_k\) is a positive stepsize. The method, together with its many variations, will be discussed extensively in this book, starting with Chapter 2. We will see that while it may not yield a cost reduction for any value of \(\alpha_k\)
it has another descent property, which enhances the convergence process: at any nonoptimal point $x_k$, it satisfies
\[
\text{dist}(x_{k+1}, X^*) < \text{dist}(x_k, X^*)
\]
for a sufficiently small stepsize $\alpha_k$, where dist($x, X^*$) denotes the Euclidean minimum distance of $x$ from the optimal solution set $X^*$.

### 1.6.2 Alternative Descent Methods

Another approach to extend the descent idea to nondifferentiable cost functions is the proximal algorithm, to be discussed in detail in Chapter 4. This algorithm embodies both the cost improvement and the approximation ideas. In its basic form, it approximates the minimization of a closed proper convex function $f : \mathbb{R}^n \mapsto (-\infty, \infty]$ with another minimization that involves a quadratic term. It is given by

\[
x_{k+1} \in \arg \min_{x \in \mathbb{R}^n} \left\{ f(x) + \frac{1}{2c_k} \|x - x_k\|^2 \right\}, \tag{1.82}
\]

where $x_0$ is an arbitrary starting point and $c_k$ is a positive scalar parameter (see Fig. 1.6.4). One of the motivations for the algorithm is that it “regularizes” the minimization of $f$ over $X$: the quadratic term in Eq. (1.82) when added to $f$ makes it strictly convex and coercive, so it has a unique minimum.

The algorithm has some inherent cost improvement properties for any convex cost function, differentiable or not, which facilitate its combination with other algorithmic schemes. To see the descent character of the algorithm, note that since $x_{k+1}$ minimizes $f(x) + \frac{1}{2c_k} \|x - x_k\|^2$, we have by setting $x = x_k$,

\[
f(x_{k+1}) + \frac{1}{2c_k} \|x_{k+1} - x_k\|^2 \leq f(x_k).
\]

It follows that \( \{f(x_k)\} \) is monotonically nondecreasing.

There are several variations of the proximal algorithm ideas, as we will see in Chapters 4 and 5. Some of these variations involve modification of the proximal minimization problem Eq. (1.82), since it is essential that this problem is conveniently solvable. Some examples are:

(a) Approximation of $f$ in Eq. (1.82) by a polyhedral or other function. One possibility is bundle methods, which involve a combination of the proximal and the polyhedral approximation ideas. The motivation here is to simplify the proximal minimization subproblem (1.82), replacing it for example with the quadratic programming problem, which results when $f$ is replaced by a polyhedral approximation.
Sec. 1.6 Iterative Descent Algorithms

Figure 1.6.4. Illustration of the proximal algorithm (1.82) and its descent property. The minimum of \( f(x) + \frac{1}{2c_k} \|x - x_k\|^2 \) is attained at the unique point \( x_{k+1} \) at which the graph of the quadratic function \( -\frac{1}{2c_k} \|x - x_k\|^2 \), raised by the amount

\[ \gamma_k = f(x_{k+1}) + \frac{1}{2c_k} \|x_{k+1} - x_k\|^2, \]

just touches the graph of \( f \). Since \( \gamma_k < f(x_k) \), it follows that \( f(x_{k+1}) < f(x_k) \), unless \( x_k \) minimizes \( f \), which happens if and only if \( x_{k+1} = x_k \).

(b) The use of a nonquadratic proximal term \( D_k(x; x_k) \) in Eq. (1.82), in place of \( (1/2c_k)\|x - x_k\|^2 \), i.e., the iteration

\[ x_{k+1} \in \arg \min_{x \in \mathbb{R}^n} \{ f(x) + D_k(x; x_k) \}. \]  

This approach may be useful when \( D_k \) has a special form that matches the structure of \( f \).

(c) Linear approximation of \( f \) using its gradient at \( x_k \)

\[ f(x) \approx f(x_k) + \nabla f(x_k)'(x - x_k), \]

assuming that \( f \) is differentiable. Then, in place of Eq. (1.83), we obtain the iteration

\[ x_{k+1} \in \arg \min_{x \in \mathbb{R}^n} \{ f(x_k) + \nabla f(x_k)'(x - x_k) + D_k(x; x_k) \}. \]

When the proximal term \( D_k(x; x_k) \) is the quadratic \( (1/2c_k)\|x - x_k\|^2 \), this can be seen to be equivalent to the gradient projection iteration (1.76): 

\[ x_{k+1} = P_X(x_k - c_k \nabla f(x_k)), \]

but there are other choices of \( D_k \) that lead to interesting methods, known as mirror descent algorithms.
(d) The proximal gradient algorithm, which applies to the problem
\[
\begin{align*}
\text{minimize} & \quad f(x) + h(x) \\
\text{subject to} & \quad x \in X
\end{align*}
\]
where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is a differentiable convex function, \( h : \mathbb{R}^n \rightarrow (-\infty, \infty] \) is a closed proper convex function, and \( X \) is a closed convex set. This algorithm combines ideas from the gradient projection method and the proximal method. It is given by
\[
x_{k+1} \in \arg \min_{x \in X} \left\{ \nabla f(x_k)'(x - x_k) + h(x) + \frac{1}{2c_k} \| x - x_k \|^2 \right\}, \quad (1.84)
\]
Thus if \( f \) is a linear function, we obtain the proximal algorithm for minimizing \( f + h \) over \( X \). If \( h(x) \equiv 0 \), then we obtain the gradient projection method as in (c) above. Note that there is an alternative/equivalent way to write the algorithm (1.84):
\[
z_k = x_k - \alpha \nabla f(x_k), \quad x_{k+1} \in \arg \min_{x \in X} \left\{ h(x) + \frac{1}{2\alpha} \| x - z_k \|^2 \right\} \quad \text{as can be verified by expanding the quadratic}
\]
\[
\| x - z_k \|^2 = \| x - x_k + \alpha \nabla f(x_k) \|^2.
\]
Thus the method alternates gradient steps on \( f \) with proximal steps on \( h \). The advantage that this method may have over the proximal algorithm is that the proximal step in Eq. (1.85) is executed with \( h \) rather than with \( f + h \), and this may be significant if \( h \) has simple/favorable structure (e.g., \( h \) is the \( \ell_1 \) norm), while \( f \) has unfavorable structure.

In Chapter 5, we will also discuss another descent approach, called \( \epsilon \)-descent, which aims to avoid the difficulties due to the discontinuous change of the steepest descent direction (cf. Fig. 1.6.3) by projecting on an appropriately enlarged version of the subdifferential. This method is theoretically interesting and it will be used in Section 5.8 for the duality analysis of extended monotropic programming, an important class of problems with partially separable structure.

1.6.3 Incremental Algorithms

An interesting form of approximate gradient, or more generally, subgradient method is an incremental variant, which applies to minimization over a closed convex set \( X \) of an additive cost function of the form
\[
f(x) = \sum_{i=1}^m f_i(x), \quad (1.86)
\]
where the functions \( f_i : \mathbb{R}^n \to \mathbb{R} \) are convex. We mentioned several contexts where cost functions of this type arise in Section 1.3. The idea of the incremental approach is to sequentially take steps along the subgradients of the component functions \( f_i \), with intermediate adjustment of \( x \) after processing each \( f_i \).

Incremental methods are particularly interesting when the number of cost terms \( m \) is very large. Then a full subgradient step is very costly, and one hopes to make progress with approximate but much cheaper incremental steps.

In a common type of incremental subgradient method, an iteration is viewed as a cycle of \( m \) subiterations. If \( x_k \) is the vector obtained after \( k \) cycles, the vector \( x_{k+1} \) obtained after one more cycle is

\[
x_{k+1} = \psi_{m,k},
\]

where starting with

\[
\psi_{0,k} = x_k,
\]

we obtain \( \psi_{m,k} \) after the \( m \) steps

\[
\psi_{i,k} = P_X(\psi_{i-1,k} - \alpha_k g_{i,k}), \quad i = 1, \ldots, m,
\]

with \( g_{i,k} \) being a subgradient of \( f_i \) at \( \psi_{i-1,k} \).

In a randomized version of the method, given \( x_k \) at iteration \( k \), an index \( i_k \) is chosen from the set \( \{1, \ldots, m\} \) randomly, and the next iterate \( x_{k+1} \) is generated by

\[
x_{k+1} = P_X(x_k - \alpha_k g_{i_k}), \quad i = 1, \ldots, m,
\]

where \( g_{i_k} \) is a subgradient of \( f_{i_k} \) at \( x_k \). Here it is important that all indexes are chosen with equal probability. It turns out that there is a rate of convergence advantage for this and other types of randomization, as we will discuss in Chapter 5. We will ignore for the moment the possibility of randomizing the component selection, and assume cyclic selection as in Eqs. (1.87)-(1.88).

In the present section we will explain the ideas underlying incremental methods by focusing primarily on the case where the component functions \( f_i \) are differentiable. We will thus consider incremental methods that make use of the gradients \( \nabla f_i(x_k) \) and possibly the Hessian matrices \( \nabla^2 f_i(x_k) \). We will discuss the case where \( f_i \) may be nondifferentiable in Chapters 2 and 5, after the analysis of nonincremental subgradient methods to be given in Chapter 2 (for some representative references on incremental subgradient other related methods for nondifferentiable \( f_i \), see [NeB00], [NeB01], [BNO03], [Ber11], [Ned11], [WaB13], and the references quoted there).
Incremental Gradient Method

Assume that the component functions $f_i$ are differentiable. We refer to the method
\[ x_{k+1} = \psi_{m,k}, \]  
where starting with $\psi_{0,k} = x_k$, we generate $\psi_{m,k}$ after the $m$ steps
\[ \psi_{i,k} = P_X \left( \psi_{i-1,k} - \alpha_k \nabla f_i(\psi_{i-1,k}) \right), \quad i = 1, \ldots, m, \]  
[cf. (1.87)-(1.88)], as the incremental gradient method. A well known and important example of such a method is the following. Together with its many variations, it is widely used in computerized imaging; see e.g., [Her09].

**Example 1.6.1: (Kaczmarz Method)**

Consider the least squares problem where
\[ f_i(x) = \frac{1}{2\|c_i\|^2}(c'_i x - b_i)^2, \quad i = 1, \ldots, m, \]
where $c_i$ are given vectors in $\mathbb{R}^n$ and $b_i$ are given scalars. The constant term $1/(2\|c_i\|^2)$ multiplying each of the squared functions $(c'_i x - b_i)^2$ serves a scaling purpose: with its inclusion, the components $f_i$ have a common Hessian matrix, $\nabla^2 f_i(x) = c e'$, where $e$ is the unit vector. This type of scaling is often used in least squares optimization problems (see [Ber99] for explanations). The incremental gradient method (1.90)-(1.91) is $x_{k+1} = \psi_{m,k}$, where $\psi_{m,k}$ is obtained after the $m$ steps
\[ \psi_{i,k} = \psi_{i-1,k} - \frac{\alpha_k}{\|c_i\|^2} (c'_i \psi_{i-1,k} - b_i) c_i, \quad i = 1, \ldots, m, \]  
starting with $\psi_{0,k} = x_k$.

The stepsize $\alpha_k$ may be chosen in a number of different ways, but if $\alpha_k$ is chosen identically equal to 1,
\[ \alpha_k \equiv 1, \]
we obtain the Kaczmaz method, which dates to 1937. The interpretation of the iteration (1.92) in this case is very simple: $\psi_{i,k}$ is obtained by projecting $\psi_{i,k-1}$ onto the hyperplane defined by the single equation $c'_i x = b_i$. Indeed from Eq. (1.92) with $\alpha_k = 1$, it is easily verified that $c'_i \psi_{i,k} = b_i$ and that $\psi_{i,k} - \psi_{i,k-1}$ is orthogonal to the hyperplane, since it is proportional to its normal $c_i$; see Fig. 1.6.5(a).

If the system of equations $c'_i x = b_i$, $i = 1, \ldots, m$, is consistent, i.e., has a unique solution $x^*$, then the unique minimum of $\sum_{i=1}^m f_i(x)$ is $x^*$. In this case it turns out that with $\alpha_k \equiv 1$, the method converges to $x^*$. The convergence process is illustrated in Fig. 1.6.5(b): the distance $\|\psi_{i,k} - x^*\|$ is
Sec. 1.6  Iterative Descent Algorithms

\[ \psi_{i,k} = \psi_{i-1,k} - \frac{1}{\|c_i\|^2} (c_i' \psi_{i-1,k} - b_i)c_i, \quad i = 1, \ldots, m. \]

(a) \( \psi_{i,k} \) is obtained by projecting \( \psi_{i-1,k} \) onto the hyperplane defined by the single equation \( c_i' x = b_i \). (b) The convergence process for the case where the system of equations \( c_i' x = b_i, i = 1, \ldots, m \), is consistent and has a unique solution \( x^* \). Here \( m = 3 \), and \( x_k \) is the vector obtained after \( k \) cycles through the equations. Each incremental iteration decreases the distance to \( x^* \), unless the current iterate lies on the hyperplane defined by the corresponding equation.

These convergence properties will be discussed further later in this section, and in Chapters 2 and 5.

The motivation for the incremental method is faster convergence: we hope that far from the solution, a single cycle of the incremental gradient method will be as effective as several (as many as \( m \)) iterations of the ordinary gradient method (think of the case where the components \( f_i \) are similar in structure). Near a solution, however, the incremental method may not be as effective. Still, the frequent superiority of the incremental method when far from convergence can be a decisive advantage for problems where solution accuracy is not of paramount importance.
To be more specific, we note that there are two complementary performance issues to consider in comparing incremental and nonincremental methods:

(a) **Progress when far from convergence.** Here the incremental method can be much faster. For an extreme case let $X = \mathbb{R}^n$ (no constraints), and take $m$ very large and all components $f_i$ identical to each other. Then an incremental iteration requires $m$ times less computation than a classical gradient iteration, but gives exactly the same result, when the stepsize is appropriately scaled to be $m$ times larger. While this is an extreme example, it reflects the essential mechanism by which incremental methods can be much superior: far from the minimum a single component gradient will point to “more or less” the right direction, at least most of the time.

(b) **Progress when close to convergence.** Here the incremental method can be inferior. As a case in point, assume that all components $f_i$ are differentiable functions. Then the nonincremental gradient projection method can be shown to converge with a constant stepsize under reasonable assumptions, as we will see in Section 5.1. However, the incremental method requires a diminishing stepsize, and its ultimate rate of convergence can be much slower. When the component functions $f_i$ are nondifferentiable, both the nonincremental and the incremental subgradient methods require a diminishing stepsize. The nonincremental method tends to require a smaller number of iterations, but each of the iterations involves all the components $f_i$ and
thus larger computation overhead, so that on balance, in terms of computation time, the incremental method tends to perform better.

As an illustration consider the following example.

Example 1.6.2:

Consider a scalar linear least squares problem where the components \( f_i \) have the form

\[
f_i(x) = \frac{1}{2} (c_i x - b_i)^2, \quad x \in \mathbb{R},
\]

where \( c_i \) and \( b_i \) are given scalars with \( c_i \neq 0 \) for all \( i \). The minimum of each of the components \( f_i \) is

\[
x_i^* = \frac{b_i}{c_i},
\]

while the minimum of the least squares cost function \( f = \sum_{i=1}^{m} f_i \) is

\[
x^* = \frac{\sum_{i=1}^{m} c_i b_i}{\sum_{i=1}^{m} c_i^2}.
\]

It can be seen that \( x^* \) lies within the range of the component minima

\[
R = \left[ \min_i x_i^*, \max_i x_i^* \right],
\]

and that for all \( x \) outside the range \( R \), the gradient

\[
\nabla f_i(x) = (c_i x - b_i)c_i
\]

has the same sign as \( \nabla f(x) \) (see Fig. 1.6.7). As a result, when outside the region \( R \), the incremental gradient method

\[
\psi_i = \psi_{i-1} - \alpha_k (c_i \psi_{i-1} - b_i)c_i
\]

approaches \( x^* \) at each step, provided the stepsize \( \alpha_k \) is small enough. In fact it is sufficient that

\[
\alpha_k \leq \min_i \frac{1}{c_i^2}.
\]

However, for \( x \) inside the region \( R \), the \( i \)th step of a cycle of the incremental gradient method need not make progress. It will approach \( x^* \) (for small enough stepsize \( \alpha_k \)) only if the current point \( \psi_{i-1} \) does not lie in the interval connecting \( x_i^* \) and \( x^* \). This induces an oscillatory behavior within the region \( R \), and as a result, the incremental gradient method will typically not converge to \( x^* \) unless \( \alpha_k \to 0 \).

Let us now compare the incremental gradient method with the nonincremental version, which takes the form

\[
x_{k+1} = x_k - \alpha_k \sum_{i=1}^{m} (c_i x_k - b_i)c_i.
\]
Figure 1.6.7. Illustrating the advantage of incrementalism when far from the optimal solution. The region of component minima

\[ R = \left[ \min_i x_i^*, \max_i x_i^* \right] \]

is labeled as the “region of confusion.” It is the region where the method does not have a clear direction towards the optimum. The ith step in an incremental gradient cycle is a gradient step for minimizing \((c_i x - b_i)^2\), so if \(x\) lies outside the region of component minima \(R = \left[ \min_i x_i^*, \max_i x_i^* \right] \)
(labeled as the “farout region”) and the stepsize is small enough, progress towards the solution \(x^*\) is made.

It can be shown that this method converges to \(x^*\) for any constant stepsize \(\alpha_k \equiv \alpha\) satisfying

\[ 0 < \alpha \leq \frac{1}{\sum_{i=1}^{n} c_i^2} \]

On the other hand, for \(x\) outside the region \(R\), an iteration of the nonincremental method need not make more progress towards the solution than a single step of the incremental method. In other words, with comparably intelligent stepsize choices, far from the solution (outside \(R\), a single cycle through the entire set of component functions by the incremental method is roughly as effective as \(m\) iterations by the nonincremental method, which require \(m\) times as many component gradient calculations.

Example 1.6.3:

The preceding example assumes that each component function \(f_i\) has a minimum, so that the range of component minima is defined. In cases where the components \(f_i\) have no minima, a similar phenomenon may occur. As an
example consider the case where $f$ is the sum of increasing and decreasing convex exponentials, i.e.,

$$f_i(x) = a_i e^{b_i x}, \quad x \in \mathbb{R},$$

where $a_i$ and $b_i$ are scalars with $a_i > 0$ and $b_i \neq 0$. Let

$$I^+ = \{i \mid b_i > 0\}, \quad I^- = \{i \mid b_i < 0\},$$

and assume that $I^+$ and $I^-$ have roughly equal numbers of components. Let also $x^\star$ be the minimum of $\sum_{i=1}^m f_i$.

Then it can be seen that if $x_k >> x^\star$, $x_{k+1}$ will be substantially closer to $x^\star$ if $i \in I^+$, and negligibly further away than $x^\star$ if $i \in I^-$. The net effect, averaged over many incremental iterations, is that if $x_k >> x^\star$, an incremental gradient iteration makes roughly one half the progress of a full gradient iteration, with $m$ times less overhead for calculating gradients. The same is true if $x_k << x^\star$. On the other hand as $x_k$ gets closer to $x^\star$ the advantage of incrementalism diminishes, similar to the preceding example. In fact in order for the incremental method to converge, a diminishing stepsize is necessary, which will ultimately make the convergence slower than the one of the nonincremental gradient method with a constant stepsize.

The preceding examples rely on $x$ being one-dimensional, but in many multidimensional problems the same qualitative behavior can be observed. In particular, the incremental gradient method, by processing the $i$th component $f_i$, can make progress towards the solution in the region where the component function gradient $\nabla f_i(\psi_{i-1})$ makes an angle less than 90 degrees with the full cost function gradient $\nabla f(\psi_{i-1})$. If the components $f_i$ are not “too dissimilar,” this is likely to happen in a region of points that are not too close to the optimal solution set.

The choice of the stepsize $\alpha_k$ plays an important role in the performance of incremental gradient methods. On close examination, it turns out that the iterate differential $x_k - x_{k+1}$ corresponding to a full cycle of the incremental gradient method and the corresponding vector $\alpha_k \nabla f(x_k)$ of its nonincremental counterpart differ by an error that is proportional to the stepsize. For this reason a diminishing stepsize is essential for convergence to a stationary point of $f$. However, it turns out that a peculiar form of convergence also typically occurs for the incremental gradient method if the stepsize is constant but sufficiently small. In this case, the iterates converge to a “limit cycle”, whereby the $i$th iterates $\psi_i$ within the cycles converge to a different limit than the $j$th iterates $\psi_j$ for $i \neq j$. The sequence $\{x_k\}$ of the iterates obtained at the end of cycles converges, except
that the limit obtained need not be a stationary point of $f$. The limit tends to be close to a stationary point when the constant stepsize is small (see [BeT96], Section 3.2, or [Ber99], Section 1.5 for an analysis of the case where the components $f_i$ are quadratic).

In practice, it is common to use a constant stepsize for a (possibly prespecified) number of iterations, then decrease the stepsize by a certain factor, and repeat, up to the point where the stepsize reaches a prespecified minimum. An alternative possibility is to use a stepsize $\alpha_k$ that diminishes to 0 at an appropriate rate [cf. Eq. (1.73)].

Incremental gradient and subgradient methods have a rich theory, which includes convergence and rate of convergence analysis, optimization and randomization issues of the component order selection, and distributed computation aspects. Moreover they admit interesting combinations with other methods, such as the proximal algorithm. We will fully discuss their properties and extensions in Chapter 5.

### Aggregated Gradient Methods

Another variant of incremental gradient is the *incremental aggregated gradient method*, which has the form

$$x_{k+1} = P_X \left( x_k - \alpha_k \sum_{\ell=0}^{m-1} \nabla f_{i_{k-\ell}}(x_{k-\ell}) \right), \quad (1.93)$$

where $f_{i_k}$ is the new component function selected for iteration $k$. Here, the component indexes $i_k$ may either be selected in a cyclic order [$i_k = (k \bmod m) + 1$], or according to some randomization scheme, consistently with Eq. (1.89). Also for $k < m$, the summation should go up to $\ell = k$, and $\alpha$ should be replaced by a corresponding larger value, such as $\alpha_k = m\alpha/(k+1)$. This method computes the gradient incrementally, one component per iteration, but in place of the single component gradient, it uses an approximation to the total cost gradient $\nabla f(x_k)$, which is the sum of the component gradients computed in the past $m$ iterations.

A disadvantage of this method is that in order for the effect of aggregating the component gradients to fully manifest itself, at least one pass through the components must be made, which may be too long if $m$ is very large. However, there is analytical and experimental evidence that by aggregating the component gradients one may be able to attain a faster asymptotic convergence rate; see [BFG07], [SLB13].

Another disadvantage is that the method requires that the most recent component gradients be kept in memory, so that when a component gradient is reevaluated at a new point, the preceding gradient of the same component is discarded from the sum of gradients of Eq. (1.93). There have been alternative implementations of the incremental aggregated gradient method idea that ameliorate this memory issue, by recalculating the full
Sec. 1.6 Iterative Descent Algorithms

Gradient periodically and replacing an old component gradient by a new one; see [JoZ13], [ZMJ13], [XiZ14]. In particular, instead of the gradient sum

\[ d_k = \sum_{\ell=0}^{m-1} \nabla f_{i_k-\ell}(x_{k-\ell}), \]

in Eq. (1.93), these methods use

\[ \tilde{d}_k = \nabla f_{i_k}(x_k) + \sum_{\ell=1}^{m-1} \nabla f_{i_k-\ell}(\tilde{x}_k), \]

where \( \tilde{x}_k \) is the most recent point where the full gradient has been calculated. To calculate \( \tilde{d}_k \) one only needs to compute the difference of the two gradients

\[ \nabla f_{i_k}(x_k) - \nabla f_{i_k}(\tilde{x}_k) \]

and add it to the full gradient \( \sum_{\ell=0}^{m-1} \nabla f_{i_k-\ell}(\tilde{x}_k) \). This bypasses the need for extensive memory storage, and with proper implementation, typically leads to small degradation in performance.

**Incremental Gradient Method with Momentum**

The incremental aggregated gradient method (1.93), for the unconstrained case where \( X = \mathbb{R}^n \), bears a relation with an important variant of the incremental gradient method that involves extrapolation along the direction of the difference of the preceding two iterates:

\[ x_{k+1} = x_k - \alpha_k \nabla f_{i_k}(x_k) + \beta_k (x_k - x_{k-1}), \quad (1.94) \]

where \( f_{i_k} \) is the component function selected for iteration \( k \), \( \beta_k \) is a scalar in \([0, 1)\), and we define \( x_{-1} = x_0 \). This is sometimes called incremental gradient method with momentum; see e.g., [MaS94], [Tse98]. The nonincremental version of this method is the so called heavy ball method [Pol64], which for constant \( \alpha \) and \( \beta \) can be shown to have faster convergence rate than the corresponding gradient method (see [Pol87], Section 3.2.1). A nonincremental method of this type, but with value of \( \beta_k \) that converges to 1 at a special rate, proposed in [Nes83], has received a lot of attention because it has optimal iteration complexity properties under certain conditions (see Section 5.2).

The heavy ball method (1.94) is related with the aggregated gradient method (1.93) when \( \beta_k \approx 1 \). In particular, when \( \alpha_k \equiv \alpha \) and \( \beta_k \equiv \beta \), the sequence generated by Eq. (1.94) satisfies

\[ x_{k+1} = x_k - \alpha \sum_{\ell=0}^{k} \beta^\ell \nabla f_{i_{k-\ell}}(x_{k-\ell}) \quad (1.95) \]
[both iterations (1.93) and (1.95) involve different types of diminishing dependence on past gradient components]. Thus, the iteration (1.94) provides an approximate implementation of the incremental aggregated gradient method (1.93), which does not have the memory storage issue of the latter.

A further way to intertwine the ideas of the aggregated gradient method (1.93) and the heavy ball method (1.94) for the unconstrained case \(X = \mathbb{R}^n\) is to form an infinite sequence of components

\[
f_1, f_2, \ldots, f_m, f_1, f_2, \ldots, f_m, f_1, f_2, \ldots, \tag{1.96}
\]

and group together blocks of successive components. In particular, we may add \(p > 1\) preceding gradients to the current component gradient in iteration (1.94), thus iterating according to

\[
x_{k+1} = x_k - \alpha_k \sum_{\ell=0}^{p} \nabla f_{k-\ell}(x_{k-\ell}) + \beta_k(x_k - x_{k-1}). \tag{1.97}
\]

Here \(f_{ik}\) is the component function selected for iteration \(k\) using the order of the sequence (1.96). This essentially amounts to reformulating the problem by redefining the components as sums of \(p + 1\) successive components and applying an approximation of the incremental heavy ball method (1.94). The advantage of the method (1.97) over the aggregated gradient method is that it requires keeping in memory only \(p\) previous component gradients.

**Stochastic Subgradient Methods**

Incremental subgradient methods are related to methods that aim to minimize an expected value

\[
f(x) = E \{ F(x, w) \},
\]

where \(w\) is a random variable, and \(F(\cdot, w) : \mathbb{R}^n \mapsto \mathbb{R}\) is a convex function for each possible value of \(w\). The stochastic subgradient method for minimizing \(f\) over a closed convex set \(X\) is given by

\[
x_{k+1} = P_X (x_k - \alpha_k g(x_k, w_k)), \tag{1.98}
\]

where \(w_k\) is a sample of \(w\) and \(g(x_k, w_k)\) is a subgradient of \(F(\cdot, w_k)\) at \(x_k\). This method has a rich theory and a long history, particularly for the case where \(F(\cdot, w)\) is differentiable for each value of \(w\) (see [PoT73], [Lju77], [KuC78], [TBA86], [Pol87], [BeT89], [BeT96], [BeT00], [Pfl96], [KuY97], [Mey07], [Bor08], [NJL09]). It is strongly related with an important iterative algorithmic approach known as stochastic approximation.
If we view the expected value cost $E\{F(x, w)\}$ as a weighted sum of cost function components, we see that the stochastic subgradient method (1.98) is related to the incremental subgradient method

$$x_{k+1} = P_X(x_k - \alpha_k g_i,k)$$  \hspace{1cm} (1.99)

for minimizing a finite sum $\sum_{i=1}^{m} f_i$ of components, when randomization is used for component selection [cf. Eq. (1.89)]. An important difference is that the former method involves sequential sampling of cost components $F(x, w)$ from an infinite population under some statistical assumptions, while in the latter the set of cost components $f_i$ is predetermined and finite. However, it is possible to view the incremental subgradient method (1.99), with uniform randomized selection of the component function $f_i$ (i.e., with $i_k$ chosen to be any one of the indexes 1, ..., $m$, with equal probability $1/m$, and independently of preceding choices), as a stochastic subgradient method.

Despite the apparent similarity of the incremental and the stochastic subgradient methods, the view that the problem

$$\text{minimize } f(x) = \sum_{i=1}^{m} f_i(x)$$
$$\text{subject to } x \in X,$$  \hspace{1cm} (1.100)

can simply be treated as a special case of the problem

$$\text{minimize } f(x) = E\{F(x, w)\}$$
$$\text{subject to } x \in X,$$  \hspace{1cm} (1.101)

is flawed. One reason is that the finite-component problem (1.100) is often genuinely deterministic, and to view it as a stochastic problem at the outset may mask some of its important characteristics, such as the number $m$ of cost components, or the sequence in which the components are ordered and processed. These characteristics may potentially be algorithmically exploited. For example, with insight into the problem’s structure, one may be able to discover a special deterministic or partially randomized order of processing the component functions that is superior to a uniform randomized order.

**Example 1.6.4:**

Consider the one-dimensional problem

$$\text{minimize } f(x) = \frac{1}{2} \sum_{i=1}^{m} (x - w_i)^2$$
$$\text{subject to } x \in \mathbb{R},$$
where the scalars \( w_i \) are given by

\[
  w_i = \begin{cases} 
    1 & \text{if } i: \text{odd}, \\ 
    -1 & \text{if } i: \text{even}.
  \end{cases}
\]

Assuming that \( m \) is an even number, the optimal solution is \( x^* = 0 \).

An incremental gradient method with the commonly used diminishing stepsize \( \alpha_k = 1/(k + 1) \) chooses a component index \( i_k \) at iteration \( k \), and updates \( x_k \) according to

\[
  x_{k+1} = x_k - \frac{1}{k + 1}(x_k - w_{i_k}),
\]

starting with some initial iterate \( x_0 \). It is then easily verified by induction that

\[
  x_k = \frac{x_0}{k} + \frac{w_{i_0} + \cdots + w_{i_{k-1}}}{k}, \quad k = 1, 2, \ldots.
\]

Thus the iteration error, which is \( x_k \) (since \( x^* = 0 \)), consists of two terms. The first is the error term \( x_0/k \), which is independent of the method of selecting \( i_k \), and the second is the error term

\[
  e_k = \frac{w_{i_0} + \cdots + w_{i_{k-1}}}{k},
\]

which depends on the selection method for \( i_k \).

If \( i_k \) is chosen by independently randomizing with equal probability 1/2 over the odd and even cost components, then \( e_k \) will be a random variable whose variance can be calculated to be \( 1/2k \). Thus the standard deviation of the error \( x_k \) will be of order \( O(1/\sqrt{k}) \). If on the other hand \( i_k \) is chosen by the deterministic order, which alternates between the odd and even components, we will have \( e_k = 1/k \) for the odd iterations and \( e_k = 0 \) for the even iterations, so the error \( x_k \) will be of order \( O(1/k) \), much smaller than the one for the randomized order. Of course, this is a favorable deterministic order, and we may obtain much worse results with an unfavorable deterministic order (such as selecting first all the odd components and then all the even components). However, the point here is that if we take the view that we are minimizing an expected value, we are disregarding at the outset information about the problem’s structure that could be algorithmically useful.

Let us also note that in Chapter 5 we will compare more formally various component selection orders in incremental methods. Our analysis will indicate that in the absence of problem-specific knowledge that can be exploited to select a favorable deterministic order, a uniform randomized order (each component \( f_i \) chosen with equal probability 1/m at each iteration, independently of preceding choices) has superior worst-case complexity.

Still, however, experimentation shows that by suitably mixing the deterministic and the stochastic order selection methods we may produce better practical results. As an example, a popular technique for incremental
methods, called random reshuffling, is to process the component functions \( f_i \) in cycles, with each component selected once in each cycle, and to reorder randomly the components after each cycle. This alternative order selection scheme has the nice property of allocating exactly one computation slot to each component in an \( m \)-slot cycle (\( m \) incremental iterations). By comparison, choosing components by uniform sampling allocates one computation slot to each component on the average, but some components may not get a slot while others may get more than one. A nonzero variance in the number of slots that any fixed component gets within a cycle, may be detrimental to performance, and suggests that reshuffling randomly the order of the component functions after each cycle works better. While it seems difficult to establish this fact analytically, a justification is suggested by the view of the incremental gradient method as a gradient method where the gradient is computed with error. The error has apparently greater variance in the uniform sampling method than in the random reshuffling method. Heuristically, if the variance of the error is larger, the direction of descent deteriorates, suggesting slower convergence.

Our conclusion is that it may be beneficial to search for a favorable order for processing the component functions \( f_i \) in an incremental method, exploiting whatever problem-specific information may be available, rather than ignore all prior information and apply a uniform randomized order of the type commonly used in stochastic gradient methods.

**Incremental Newton Methods**

We will now consider an incremental version of Newton’s method for unconstrained minimization of an additive cost function of the form

\[
f(x) = \sum_{i=1}^{m} f_i(x),
\]

(1.102)

where the functions \( f_i : \mathbb{R}^n \to \mathbb{R} \) are convex and twice continuously differentiable. Let us introduce a function that represents the quadratic approximation \( \tilde{f}_i \) of a function \( f_i \) at a vector \( \psi \in \mathbb{R}^n \), which is the second order Taylor expansion of \( f_i \) at \( \psi \):

\[
\tilde{f}_i(x; \psi) = \nabla f_i(\psi)'(x - \psi) + \frac{1}{2}(x - \psi)' \nabla^2 f_i(\psi)(x - \psi), \quad \forall \ x, \psi \in \mathbb{R}^n.
\]

Similar to Newton’s method, which minimizes a quadratic approximation at the current point of the cost function [cf. Eq. (1.75)], the incremental form of Newton’s method minimizes a sum of quadratic approximations of components. Similar to the incremental gradient method, we view an iteration as a cycle of \( m \) subiterations, each involving a single component \( f_i \), and its gradient and Hessian at the current point within the cycle. In particular, if \( x_k \) is the vector obtained after \( k \) cycles, the vector \( x_{k+1} \) obtained after one more cycle is

\[
x_{k+1} = \psi_{m,k},
\]

(1.103)
where starting with $\psi_{0,k} = x_k$, we obtain $\psi_{m,k}$ after the $m$ steps†

$$
\psi_{i,k} = \arg \min_{x \in \mathbb{R}^n} \sum_{\ell=1}^{i} \tilde{f}_{\ell}(x; \psi_{\ell-1,k}), \quad i = 1, \ldots, m.
$$

(1.104)

If all the functions $f_i$ are quadratic, it can be seen that the method finds the solution in a single cycle. The reason is that when $f_i$ is quadratic, each $f_i(x)$ differs from $\tilde{f}_i(x; \psi)$ by a constant, which does not depend on $x$. Thus the difference

$$
\sum_{i=1}^{m} f_i(x) - \sum_{i=1}^{m} \tilde{f}_i(x; \psi_{i-1,k})
$$

is a constant that is independent of $x$, and minimization of either sum in the above expression gives the same result.

As an example, consider a linear least squares problem, where

$$
f_i(x) = \frac{1}{2}(c_i'x - b_i)^2, \quad i = 1, \ldots, m.
$$

Then the $i$th subiteration within a cycle minimizes

$$
\sum_{\ell=1}^{i} f_{\ell}(x),
$$

and when $i = m$, the solution of the problem is obtained (see Fig. 1.6.8). This convergence behavior should be compared with the one for the Kaczmarz method (cf. Fig. 1.6.6).

† Here we assume that the quadratic minimizations to generate $\psi_{m,k}$ have a solution, and when there are multiple solutions, we choose the solution of minimum norm (a, so called, pseudoinverse solution). There are several ways to bypass this requirement. One possibility is to add to $f_1$ a small positive definite quadratic term, such as $\frac{1}{2}\|x - x_k\|^2$. Another possibility is to lump together several of the component functions for which the sum of the quadratic approximations at $x_k$ is positive definite.
Sec. 1.6 Iterative Descent Algorithms

It is important to note that the computations of Eq. (1.104) can be carried out efficiently. For simplicity, let as assume that \( \tilde{f}_1(x; \psi) \) is a positive definite quadratic, so that for all \( i, \psi_{i,k} \) is well defined as the unique solution of the minimization problem in Eq. (1.104). We will show that the incremental Newton method (1.104) can be implemented in terms of the incremental update formula

\[
\psi_{i,k} = \psi_{i-1,k} - D_{i,k} \nabla f_i(\psi_{i-1,k}),
\]

where \( D_{i,k} \) is given by

\[
D_{i,k} = \left( \sum_{\ell=1}^{i} \nabla^2 f_i(\psi_{\ell-1,k}) \right)^{-1},
\]

and is generated iteratively as

\[
D_{i,k} = \left( D_{i-1,k}^{-1} + \nabla^2 f_i(\psi_{i,k}) \right)^{-1}.
\]

Indeed, from the definition (1.104), the quadratic function \( \sum_{\ell=1}^{i} \tilde{f}_\ell(x; \psi_{\ell-1,k}) \) is minimized by \( \psi_{i-1,k} \) and its Hessian matrix is \( D_{i-1,k}^{-1} \), so we have

\[
\sum_{\ell=1}^{i-1} \tilde{f}_\ell(x; \psi_{\ell-1,k}) = \frac{1}{2} (x - \psi_{i-1,k})' D_{i-1,k}^{-1} (x - \psi_{i-1,k}) + \text{constant}.
\]

Thus, by adding \( \tilde{f}_i(x; \psi_{i-1,k}) \) to both sides of this expression, we obtain

\[
\sum_{\ell=1}^{i} \tilde{f}_\ell(x; \psi_{\ell-1,k}) = \frac{1}{2} (x - \psi_{i-1,k})' D_{i-1,k}^{-1} (x - \psi_{i-1,k}) + \text{constant}
\]

\[
+ \frac{1}{2} (x - \psi_{i-1,k})' \nabla^2 f_i(\psi_{i-1,k})(x - \psi_{i-1,k}) + \nabla f_i(\psi_{i-1,k})' (x - \psi_{i-1,k}).
\]

Since by definition \( \psi_{i,k} \) minimizes this function, we obtain Eqs. (1.105)-(1.107).

The recursion (1.107) for the matrix \( D_{i,k} \) can often be efficiently implemented by using convenient formulas for the inverse of the sum of two matrices. In particular, if \( f_i \) is given by

\[
f_i(x) = h_i(a_i' x - b_i),
\]

for some twice differentiable convex function \( h_i : \mathbb{R} \mapsto \mathbb{R} \), vector \( a_i \), and scalar \( b_i \), we have

\[
\nabla^2 f_i(\psi_{i-1,k}) = \nabla^2 h_i(\psi_{i-1,k}) a_i a_i'.
\]
and the recursion (1.107) can be written as
\[ D_{i,k} = D_{i-1,k} - \frac{D_{i-1,k} a_i a'_i D_{i-1,k}}{\nabla^2 h_i(\psi_{i-1,k}) + a'_i D_{i-1,k} a_i}; \]
this is the well-known Sherman-Morrison formula for the inverse of the sum of an invertible matrix and a rank-one matrix.

We have considered so far a single cycle of the incremental Newton method. One algorithmic possibility for cycling through the component functions multiple times, is to simply create a larger set of components by concatenating multiple copies of the original set, that is, by forming what we refer to as the extended set of components
\[ f_1, f_2, \ldots, f_m, f_1, f_2, \ldots, f_m, f_1, f_2, \ldots \] (1.108)
The incremental Newton method, when applied to the extended set, asymptotically resembles a scaled incremental gradient method with diminishing stepsize of the type described earlier. Indeed, from Eq. (1.106), the matrix \( D_{i,k}^{-1} \) grows roughly in proportion to \( k + 1 \). From this it follows that the asymptotic convergence properties of the incremental Newton method are similar to those of an incremental gradient method with diminishing stepsize of order \( O(1/k) \)(see [Ber14]).

The following example provides some insight regarding the behavior of the method when the cost function \( f \) has a very large number of cost components, as is the case when \( f \) is defined as the average of a very large number of random samples.

**Example 1.6.5: (Infinite Number of Cost Components)**

Consider the problem
\[
\begin{align*}
\text{minimize} & \quad f(x) \overset{\text{def}}{=} \lim_{m \to \infty} \frac{1}{m} \sum_{i=1}^{m} F(x, w_i) \\
\text{subject to} & \quad x \in \mathbb{R}^n,
\end{align*}
\]
where \( \{w_k\} \) is a given sequence from some set, and each function \( F(\cdot, w_i) : \mathbb{R}^n \to \mathbb{R} \) is positive semidefinite quadratic. We assume that \( f \) is well-defined (i.e., the limit above exists for each \( x \in \mathbb{R}^n \)), and is a positive definite quadratic. This type of problem arises in linear regression models (cf. Example 1.3.1) involving an infinite amount of data that is obtained through random sampling.

The natural extension of the incremental Newton’s method, applied to the infinite set of components
\[ F(\cdot, w_1), F(\cdot, w_2), \ldots \]
generates the sequence \( \{x_k^*\} \) where
\[
x_k^* \in \arg \min_{x \in \mathbb{R}^n} f_k(x) \overset{\text{def}}{=} \frac{1}{k} \sum_{i=1}^{k} F(x, w_i).
\]

Since \( f \) is positive definite and the same is true for \( f_k \), when \( k \) is large enough, we have \( x_k^* \to x^* \), where \( x^* \) is the unique minimum of \( f \). The rate of convergence is determined strictly by the rate at which the vectors \( x_k^* \) approach \( x^* \), or equivalently by the rate at which \( f_k \) approaches \( f \).

By contrast, if we were to apply the natural extension of the incremental gradient method to this problem, the convergence rate could be much worse. There would be an error due to the difference \( (x_k^* - x^*) \), but also an additional error due to the difference \( (x_k^* - x_k) \) between \( x_k^* \) and the \( k \)th iterate \( x_k \) of the incremental gradient method, which is generally diminishing quite slowly, possibly more slowly than \( (x_k^* - x^*) \).

Generally, with proper implementation, the incremental Newton method is often substantially faster than the incremental gradient method, in terms of numbers of iterations. However, in addition to computation of second derivatives, it involves greater overhead per iteration, so it is suitable only for problems where \( n \), the dimension of \( x \), is much smaller than \( m \), the number of components \( f_i \).

**Incremental Gauss-Newton Method - The Extended Kalman Filter**

We will next consider an algorithm that operates similar to the incremental Newton method, but is specialized for the nonlinear least squares problem
\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m} \left\| g_i(x) \right\|^2 \\
\text{subject to} & \quad x \in \mathbb{R}^n,
\end{align*}
\]

where \( g_i : \mathbb{R}^n \mapsto \mathbb{R}^{n_i} \) are some nonlinear functions [cf. Eq. (1.44)]. As noted earlier, this is a common problem in practice.

We introduce a function \( \tilde{g}_i \) that represents a linear approximation of \( g_i \) at a vector \( \psi \in \mathbb{R}^n \):
\[
\tilde{g}_i(x; \psi) = \nabla g_i(\psi)'(x - \psi) + g_i(\psi), \quad \forall x, \psi \in \mathbb{R}^n,
\]

where \( \nabla g_i(\psi) \) is the \( n \times n_i \) gradient matrix of \( g_i \) evaluated at \( \psi \). Similar to the incremental gradient and Newton methods, we view an iteration as a cycle of \( m \) subiterations, involving a single component \( \left\| g_i(x) \right\|^2 \) and its gradient at the current point within the cycle. In particular, if \( x_k \) is the vector obtained after \( k \) cycles, the vector \( x_{k+1} \) obtained after one more cycle is
\[
x_{k+1} = \hat{\psi}_{m,k},
\]

where \( \hat{\psi}_{m,k} \) is the updated estimate at the current cycle.
where starting with $\psi_{0,k} = x_k$, we obtain $\psi_{m,k}$ after the $m$ steps

$$
\psi_{i,k} = \arg \min_{x \in \mathbb{R}^n} \sum_{\ell=1}^{i} \| \tilde{g}_\ell(x; \psi_{\ell-1,k}) \|^2, \quad i = 1, \ldots, m. \quad (1.111)
$$

If all the functions $g_i$ are linear, we have $\tilde{g}_\ell(x; \psi) = g_\ell(x)$, and the method solves the problem exactly in a single cycle; it then becomes identical to the incremental Newton method.

When the functions $g_i$ are nonlinear the algorithm differs from the incremental Newton method because it does not involve second derivatives of $g_i$. It may be viewed instead as an incremental version of the Gauss-Newton method, a classical nonincremental scaled gradient method for solving nonlinear least squares problems (see e.g., [Ber99], Section 1.5). It is also known as the Extended Kalman Filter, and has found extensive application in state estimation and control of dynamic systems, where it was introduced in the mid-60s (it was also independently proposed in [Dav76]).

The implementation issues of the Extended Kalman Filter are similar to the ones of the incremental Newton method. This is because both methods solve similar linear least squares problems at each iteration [cf. Eqs. (1.104) and (1.111)]. The convergence behaviors of the two methods are also similar: they asymptotically operate as scaled forms of incremental gradient methods with diminishing stepsize. Both methods are primarily well-suited for problems where the dimension of $x$ is much smaller than the number of components in the additive cost function. Their transient behavior is often much faster than the one of the incremental gradient methods. We refer to [Ber96], [MYF03] for details of the convergence analysis.

### 1.7 APPROXIMATION METHODS

As noted earlier, approximation methods for minimizing a convex function $F: \mathbb{R}^n \to \mathbb{R}$ over a convex set $X$, are based on replacing at iteration $k$, $f$ and $X$ by approximations $F_k$ and $X_k$, respectively, and finding

$$
x_{k+1} \in \arg \min_{x \in X_k} F_k(x).
$$

At the next iteration, $F_{k+1}$ and $X_{k+1}$ are generated by refining the earlier approximation, possibly based on $x_{k+1}, \ldots, x_0$, and repeating the process. Of course such a method makes sense only if the approximating problems are simpler than the original. There is a great variety of approximation methods, with different aims, and suitable for different circumstances. The present section provides a brief overview and orientation, while Chapters 3-5 provide a detailed analysis.
1.7.1 Polyhedral Approximation

In polyhedral approximation methods, $F_k$ is a polyhedral function that approximates $f$ and $X_k$ is a polyhedral set that approximates $X$. The idea is that the approximate problem is polyhedral, so it may be easier to solve than the original problem. The methods include mechanisms for progressively refining the approximation, thereby obtaining a solution of the original problem in the limit.

In Chapter 3, we will discuss the two main approaches for polyhedral approximation: outer linearization (also called the cutting plane approach) and inner linearization (also called the simplicial decomposition approach). As the name suggests, outer linearization approximates $\text{epi}(f)$ and $X$ from without, $F_k(x) \leq f(x)$ for all $x$, and $X_k \supset X$, using intersections of finite numbers of halfspaces. By contrast, inner linearization approximates $\text{epi}(f)$ and $X$ from within, $F_k(x) \geq f(x)$ for all $x$, and $X_k \subset X$, using convex hulls of finite numbers of halflines or points. Figure 1.7.1 illustrates outer and inner linearization of convex sets and functions.

We will show in Sections 3.3 and 3.4 that these two approaches are intimately connected by conjugacy and duality: the dual of an outer approximating problem is an inner approximating problem involving the conjugate of $F_k$ and the indicator function of $X_k$, and reversely. In fact, using this duality, outer and inner approximations may be combined in the same
1.7.2 Penalty, Augmented Lagrangian, and Interior Point Methods

Generally in optimization problems, the presence of constraints complicates the algorithmic solution, and limits the range of available algorithms. For this reason it is natural to try to eliminate constraints by using approximation of the corresponding indicator functions. In particular, we may replace constraints by penalty functions that prescribe a high cost for their violation. We discussed in Section 1.5 such an approximation scheme, which uses exact nondifferentiable penalty functions. In this section, we focus on differentiable penalty functions, which are not necessarily exact.

To illustrate this approach, let us consider the equality constrained problem

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in X, \quad a_i'x = b_i, \quad i = 1, \ldots, r.
\end{align*}
\]

We replace this problem with a penalized version

\[
\begin{align*}
\text{minimize} & \quad f(x) + c_k \sum_{i=1}^{r} P(a_i'x - b_i) \\
\text{subject to} & \quad x \in X,
\end{align*}
\]

where \( P(\cdot) \) is a scalar penalty function satisfying

\[
P(u) = 0 \quad \text{if} \quad u = 0,
\]

and

\[
P(u) > 0 \quad \text{if} \quad u \neq 0,
\]

The scalar \( c_k \) is a positive penalty parameter, so by increasing \( c_k \) to \( \infty \), the solution \( x_k \) of the penalized problem tends to decrease the constraint violation, and provide an increasingly accurate approximation to the original problem. An important practical point here is that \( c_k \) should be increased gradually, using the optimal solution of each approximating problem to start the algorithm that solves the next approximating problem. Otherwise serious numerical problems (often referred to as ill-conditioning) occur.

A common choice for \( P \) is the quadratic penalty function

\[
P(u) = \frac{1}{2} u^2,
\]

in which case the penalized problem (1.113) takes the form

\[
\begin{align*}
\text{minimize} & \quad f(x) + \frac{c_k}{2} \|Ax - b\|^2 \\
\text{subject to} & \quad x \in X,
\end{align*}
\]
where $Ax = b$ is a vector representation of the system of equations $a'_i x = b_i$, $i = 1, \ldots, r$.

An important enhancement of the penalty function approach is the augmented Lagrangian method where we add a linear term to $P(t)$, involving a multiplier vector $\lambda_k \in \mathbb{R}^r$. Then in place of problem (1.114), we solve the problem

$$\begin{align*}
\text{minimize} & \quad f(x) + \lambda'_k (Ax - b) + \frac{c_k}{2} \|Ax - b\|^2 \\
\text{subject to} & \quad x \in X.
\end{align*}$$

(1.115)

After a minimizing vector $x_k$ is obtained, the multiplier vector $\lambda_k$ is updated by some formula that aims to approximate an optimal dual solution. A common choice that we will discuss in Chapter 4 is

$$\lambda_{k+1} = \lambda_k + c_k (Ax_k - b).$$

(1.116)

This is known as the first order augmented Lagrangian method (also called first order method of multipliers). It is a major general purpose constrained optimization method, which applies to nonconvex problems as well. In the convex programming setting of this book, augmented Lagrangian methods embody additional favorable structure, which manifests itself for example in that for convergence it is not necessary that $c_k \to \infty$, as is typically the case for penalty methods that do not involve multiplier updates.

Generally, penalty and augmented Lagrangian methods can be used for inequality as well as equality constraints. The penalty function is modified to reflect penalization for violation of inequalities. For example the inequality constraint analog of the quadratic penalty $P(u) = \frac{1}{2} u^2$ is

$$P(u) = \frac{1}{2} \left( \max\{0, u\} \right)^2.$$

We will discuss these possibilities in greater detail in Chapter 4.

The penalty methods just discussed are known as exterior penalty methods: they approximate the indicator function of the constraint set from without. Another type of algorithm involves approximation from within, which leads to the so called interior point methods. These are important methods that find application in a broad variety problems, including linear programming. They will be discussed in Chapter 5.

### 1.7.3 Proximal Algorithm and Tikhonov Regularization

We note that the augmented Lagrangian method bears a (Fenchel) duality relation with the proximal algorithm, discussed in Section 1.6.2 and in much more detail in Chapter 4:

$$x_{k+1} \in \arg \min_{x \in \mathbb{R}^n} \left\{ f(x) + \frac{1}{2c_k} ||x - x_k||^2 \right\},$$

(1.117)
[cf. Eq. (1.82)], where \( x_0 \) is an arbitrary starting point and \( c_k \) is a positive scalar parameter. The proximal algorithm aims to minimize a general closed proper convex function \( f : \mathbb{R}^n \rightarrow (-\infty, \infty] \), and we will show in Chapter 4 that when \( f \) is the dual function of the constrained optimization problem, it becomes essentially equivalent to the multiplier iteration of the augmented Lagrangian method [cf. Eq. (1.116)]. When properly generalized, the proximal algorithm is closely related and contains as a special case another important algorithm, the so-called alternating direction method of multipliers, which is a variant of the augmented Lagrangian method (see Chapter 4, and for more detailed treatments, [BeT89], [EcB92], [BPC11]).

The connection of the proximal algorithm with the approximation approach may also be seen from Eq. (1.117). As the parameter \( c_k \) tends to \( \infty \), the quadratic regularization term becomes insignificant and the proximal minimization (1.117) approximates more closely the minimization of \( f \).

Another approximation approach that bears similarity to the proximal algorithm is Tikhonov regularization, which approximates the minimization of \( f \) with the minimization

\[
x_{k+1} \in \arg \min_{x \in \mathbb{R}^n} \left\{ f(x) + \frac{1}{2c_k} \|x\|^2 \right\}, \tag{1.118}
\]

The quadratic regularization term makes the cost function of the above problem strictly convex, and guarantees that it has a unique minimum. Sometimes the quadratic term in Eq. (1.118) is scaled and a term \( \|Sx\|^2 \) is used instead, where \( S \) is a suitable scaling matrix. The difference with the proximal algorithm (1.117) is that \( x_k \) does not enter directly the minimization to determine \( x_{k+1} \), so the method relies for its convergence on increasing \( c_k \) to \( \infty \). By contrast this is not necessary for the proximal algorithm, which is generally convergent even when \( c_k \) is left constant, as we will see in Chapter 4.

### 1.7.4 Smoothing of Nondifferentiable Problems

Generally speaking, differentiable cost functions are preferable to nondifferentiable ones, because algorithms for the former are better developed and are more effective than algorithms for the latter. Thus there is an incentive to eliminate nondifferentiabilities by “smoothing” their corners. It turns out that penalty functions and smoothing are closely related, reflecting the fact that constraints and nondifferentiabilities are also closely related. As an example of this connection, the unconstrained minimax problem

\[
\begin{align*}
\text{minimize} & \quad \max \{ f_1(x), \ldots, f_m(x) \} \\
\text{subject to} & \quad x \in \mathbb{R}^n,
\end{align*}
\]
Sec. 1.7 Approximation Methods

Approximation Methods

Figure 1.7.2. Illustration of smoothing of the function \( f(x) = \max\{0, x\} \). As \( c \to \infty \), we have \( f_{c,\lambda}(x) \to f(x) \) for all \( x \in \mathbb{R} \).

where \( f_1, \ldots, f_m \) are differentiable functions can be converted to the differentiable constrained problem

\[
\begin{align*}
\text{minimize} & \quad z \\
\text{subject to} & \quad f_j(x) \leq z, \quad j = 1, \ldots, m,
\end{align*}
\]

where \( z \) is an artificial scalar variable.

We will now describe a general technique to obtain smoothing approximations. Let \( f : \mathbb{R}^n \to (-\infty, \infty] \) be a closed proper convex function with conjugate denoted by \( f^\star \). For each \( x \in \mathbb{R}^n \), define

\[
f_{c,\lambda}(x) = \inf_{u \in \mathbb{R}^n} \left\{ f(x - u) + \lambda^\prime u + \frac{c}{2} \|u\|^2 \right\}, \tag{1.119}
\]

where \( c \) is a positive scalar, and \( \lambda \) is a vector in \( \mathbb{R}^n \). By calculating the conjugates of \( f(x - u) \) and \( \lambda^\prime u + \frac{c}{2} \|u\|^2 \), viewed as functions of \( u \), and by using the Fenchel Duality Theorem (Prop. 1.2.1), we have

\[
f_{c,\lambda}(x) = \sup_{y \in \mathbb{R}^n} \left\{ x^\prime y - f^\star(y) - \frac{1}{2c} \|y - \lambda\|^2 \right\}. \tag{1.120}
\]

It can be seen that \( f_{c,\lambda} \) approximates \( f \) in the sense that

\[
\lim_{c \to \infty} f_{c,\lambda}(x) = f^{**}(x) = f(x), \quad \forall \ x, \lambda \in \mathbb{R}^n;
\]

the double conjugate \( f^{**} \) is equal to \( f \) by the conjugacy theorem (Prop. 1.6.1/App. B). Furthermore, it can be shown using Prop. 1.2.1(c) (see also [Ber77]) that \( f_{c,\lambda} \) is convex and differentiable as a function of \( x \) for fixed \( c \) and \( \lambda \), and that the gradient \( \nabla f_{c,\lambda}(x) \) at any \( x \in \mathbb{R}^n \) can be obtained in two ways:

(i) As the vector \( \lambda + cu \), where \( u \) is the unique vector attaining the infimum in Eq. (1.119).

(ii) As the unique vector \( y \) that attains the supremum in Eq. (1.120).
An example consider smoothing a common source of nondifferentiability in the cost function or the constraints of nondifferentiable optimization problems: the scalar function

\[ f(x) = \max\{0, x\}, \quad x \in \mathbb{R}. \]

It can be verified that

\[
 f_{c,\lambda}(x) = \begin{cases} 
 x - \frac{(1 - \lambda)^2}{2c} & \text{if } \frac{1 - \lambda}{c} \leq x, \\
 \lambda x + \frac{c}{2}x^2 & \text{if } -\frac{\lambda}{c} \leq x \leq \frac{1 - \lambda}{c}, \\
 -\frac{\lambda^2}{2c} & \text{if } x \leq -\frac{\lambda}{c};
\end{cases}
\]

see Fig. 1.7.2. To use this smoothing approximation, we may replace each occurrence of \( \max\{0, x\} \) in the cost or constraint functions by \( f_{c,\lambda}(x) \) (each approximation using possibly different \( \lambda \) and \( c \)), thereby obtaining a differentiable problem that approximates the original. Also the function \( f(x) = \max\{0, x\} \) may be used as a building block to construct more complicated nondifferentiable functions, such as for example

\[ \max\{x_1, x_2\} = x_1 + \max\{0, x_1 - x_2\}; \]

see [Ber82a], Ch. 3.

**Smoothing and Augmented Lagrangians**

The smoothing technique just described can also be combined with the augmented Lagrangian method. As an example, let \( f : \mathbb{R}^n \rightarrow (-\infty, \infty] \) be a closed proper convex function with conjugate denoted by \( f^* \). Let \( F : \mathbb{R}^n \rightarrow \mathbb{R} \) be another convex function, and let \( X \) be a closed convex set. Consider the problem

\[
 \text{minimize} \quad F(x) + f(x) \\
 \text{subject to} \quad x \in X,
\]

and the equivalent problem

\[
 \text{minimize} \quad F(x) + f(x - u) \\
 \text{subject to} \quad x \in X, \ u = 0.
\]

Applying the augmented Lagrangian method (1.115)-(1.116) to the latter problem leads to minimizations of the form

\[
 (x_{k+1}, u_{k+1}) = \arg \min_{x \in X, \ u \in \mathbb{R}^n} \left\{ F(x) + f(x - u) + \lambda_k u + \frac{c_k}{2} \|u\|^2 \right\}.
\]
By first minimizing over \( u \in \mathbb{R}^n \), these minimizations yield
\[
x_{k+1} \in \text{arg min}_{x \in X} \{ F(x) + f_{c_k, \lambda_k}(x) \},
\]
where \( f_{c_k, \lambda_k} \) is the smoothed function
\[
f_{c_k, \lambda_k}(x) = \inf_{u \in \mathbb{R}^n} \left\{ f(x - u) + \lambda_k' u + \frac{c_k}{2} \| u \|^2 \right\},
\]
[cf. Eq. (1.119)]. The corresponding multiplier update (1.116) is
\[
\lambda_{k+1} = \lambda_k + c_k u_{k+1},
\]
where
\[
u_{k+1} \in \text{arg min}_{u \in \mathbb{R}^n} \left\{ f(x_k + 1 - u) + \lambda_k' u + \frac{c_k}{2} \| u \|^2 \right\}.
\]

The preceding technique can be extended so that it applies to general convex/concave minimax problems. Let \( Z \) be a nonempty convex subset of \( \mathbb{R}^m \), respectively, and \( \phi : \mathbb{R}^n \times Z \rightarrow \mathbb{R} \) is a function such that \( \phi(\cdot, z) : \mathbb{R}^n \rightarrow \mathbb{R} \) is convex for each \( z \in Z \), and \( -\phi(x, \cdot) : Z \rightarrow \mathbb{R} \) is convex and closed for each \( x \in \mathbb{R}^n \). Consider the problem
\[
\text{minimize} \quad \sup_{z \in Z} \phi(x, z)
\]
subject to \( x \in X \),
where \( X \) is a nonempty closed convex subset of \( \mathbb{R}^n \). Consider also the equivalent problem
\[
\text{minimize} \quad H(x, y)
\]
subject to \( x \in X, \ y = 0 \),
where \( H \) is the function
\[
H(x, y) = \sup_{z \in Z} \{ \phi(x, z) - y' z \}, \quad x \in \mathbb{R}^n, \ y \in \mathbb{R}^m.
\]
The augmented Lagrangian minimization (1.115) for this problem takes the form
\[
x_{k+1} \in \text{arg min}_{x \in X} h_{c_k, \lambda_k}(x),
\]
where \( h_{c, \lambda} : \mathbb{R}^n \rightarrow \mathbb{R} \) is the differentiable function given by
\[
h_{c, \lambda}(x) = \min_{y \in \mathbb{R}^m} \left\{ H(x, y) - \lambda' y + \frac{c}{2} \| y \|^2 \right\}, \quad x \in \mathbb{R}^n.
\]
The corresponding multiplier update (1.116) is
\[
\lambda_{k+1} = \lambda_k - c_k y_{k+1},
\]
where
\[
y_{k+1} \in \arg \min_{y \in \mathbb{R}^m} \left\{ H(x_{k+1}, y) - \lambda_k y + \frac{c_k}{2} \|y\|^2 \right\}.
\]

For further discussion of the relations and combination of smoothing with the augmented Lagrangian method, see [Ber75b], [Ber77], [Pap81], and for a detailed textbook analysis, [Ber82a], Ch. 3. There have also been many variations of smoothing ideas and applications in different contexts; see [Ber73], [Geo77], [Pol79], [Pol88], [BeT89b], [PiZ94], [Nes05], [Che07]. In Chapter 5, we will also see an application of smoothing as an analytical device, in the context of complexity analysis.

**Exponential Smoothing**

We have used so far a quadratic penalty function as the basis for smoothing. It is also possible to use other types of penalty functions, such as an exponential, which will be discussed in Chapter 5. As an example, a smooth approximation of the function

\[
f(x) = \max\{f_1(x), \ldots, f_m(x)\}
\]
is

\[
f_{c,\lambda}(x) = \frac{1}{c} \ln \left\{ \sum_{i=1}^{m} \lambda_i e^{c f_i(x)} \right\},
\]

where \(c > 0\), and \(\lambda = (\lambda_1, \ldots, \lambda^m)\) is a vector with

\[
\sum_{i=1}^{m} \lambda^i = 1, \quad \lambda^i > 0, \quad \forall \ i = 1, \ldots, m.
\]

There is an augmented Lagrangian method associated with this approximation. It involves minimizing over \(x \in \mathbb{R}^n\) the function \(f_{c_k,\lambda_k}(x)\) for a given \(c_k\) and \(\lambda_k\) to obtain an approximation \(x_k\) to the minimum of \(f\). This approximation is refined by setting \(c_{k+1} \geq c_k\) and

\[
\lambda_k^i = \frac{\lambda_k^i e^{c_k f_i(x_k)}}{\sum_{j=1}^{m} \lambda_k^j e^{c_k f_j(x_k)}}, \quad i = 1, \ldots, m,
\]
and by repeating the process. The generated sequence \(\{x_k\}\) can be shown to converge to the minimum of \(f\) under mild assumptions, based on general convergence properties of augmented Lagrangian methods that use nonquadratic penalty functions; see [Ber82a], Ch. 5, which also describes other nonquadratic penalty functions with favorable second order differentiability properties. The advantage of the exponential function is that, contrary to the quadratic, it produces twice differentiable approximating functions. This may be significant when Newton’s method is used to solve the smoothed problem.
Example 1.7.1: (Smoothed $\ell_1$ Regularization)

Consider the $\ell_1$-regularized least squares problem

$$\min_{\gamma} \gamma \sum_{j=1}^{n} |x_j| + \frac{1}{2} \sum_{i=1}^{m} (a_i'x - b_i)^2$$

subject to $x \in \mathbb{R}^n$, 

(1.123)

where $a_i$ and $b_i$ are given vectors and scalars, respectively (cf. Example 1.3.1). The nondifferentiable $\ell_1$ penalty may be smoothed by writing each term $|x_j|$ as $\max\{x_j, -x_j\}$ and by smoothing it using Eq. (1.121), i.e., replace it by

$$R_{c,\lambda_j}(x_j) = \frac{1}{c} \ln \left\{ \lambda_j e^{cx_j} + (1 - \lambda_j)e^{-cx_j} \right\},$$

(1.124)

where $c$ and $\lambda_j$ are scalars satisfying $c > 0$ and $\lambda_j \in (0, 1)$ (see Fig. 1.7.3). We may then consider an exponential type of augmented Lagrangian method, whereby we minimize over $\mathbb{R}^n$ the twice differentiable function

$$\gamma \sum_{j=1}^{n} R_{c_k,\lambda_{j_k}}(x_j) + \frac{1}{2} \sum_{i=1}^{m} (a_i'x - b_i)^2,$$

(1.125)

to obtain an approximation $x_k$ to the optimal solution. This approximation is refined by setting $c_{k+1} \geq c_k$ and

$$\lambda_{j_{k+1}} = \frac{\lambda_j e^{c_kx_{j_k}}}{\lambda_j e^{c_kx_{j_k}} + (1 - \lambda_j)e^{-c_kx_{j_k}}}, \quad j = 1, \ldots, n,$$

(1.126)

[cf. Eq. (1.122)], and by repeating the process. Note that the minimization of the exponentially smoothed cost function (1.125) can be carried out efficiently by incremental methods, such as the incremental gradient and Newton methods of Section 1.6.3.

As Fig. 1.7.3 suggests, the adjustment of $\lambda_j$ can selectively reduce the error $|x_j| - R_{c,\lambda_j}(x_j)$ depending on whether good approximation for positive or negative $x_j$ is desired. For this reason it is not necessary to increase $c_k$ to infinity; the multiplier iteration (1.126) is sufficient for convergence even with $c_k$ kept constant at some positive value (see [Ber82a], Ch. 5).

1.8 \hspace{1cm} NOTES AND SOURCES

There is a very extensive literature on convex optimization, and we will restrict ourselves to noting some books and research monographs,
Figure 1.7.3. Illustration of the exponentially smoothed version

\[ R_{c,\lambda}(x) = \frac{1}{c} \ln \left\{ \lambda e^{cx} + (1-\lambda)e^{-cx} \right\} \]

of the absolute value function \(|x|\). The approximation becomes asymptotically exact as \(c \to \infty\) for any fixed value of \(\lambda \in (0,1)\). Also by adjusting the multiplier \(\lambda\) within the range \((0,1)\), we can attain better approximation for \(x\) positive or negative. As \(\lambda \to 1\) (or \(\lambda \to 0\)) the approximation becomes asymptotically exact for \(x \geq 0\) (or \(x \leq 0\), respectively).

which in turn contain a lot of other references. Books relating primarily to duality theory are: Rockafellar [Roc70], Stoer and Witzgall [StW70], Ekeland and Temam [EkT76], Bonnans and Shapiro [BoS00], Zalinescu [Zal02], Auslender and Teboulle [AuT03], Bertsekas [Ber98], Bauschke and Combettes [BaC11]. Books that have among others, a substantial focus on algorithms are: Rockafellar [Roc84], Hiriart-Urruty and Lemarechal [HiL93], Bertsekas [Ber98]. The books by Rockafellar and Wets [RoW98], Borwein and Lewis [BoL00], and Bertsekas, Nedić, and Ozdaglar [BNO03] straddle the boundary between convex and variational analysis, a broad spectrum of topics that integrate classical analysis, convexity, and optimization of both convex and nonconvex (possibly nonsmooth) functions. The book by Ben-Tal and Nemirovski [BeN01] focuses on conic and semidefinite programming [see also the 2005 class notes by Nemirovski (on line)]. The book by Wolkowicz, Saigal, and Vanderbergue [WSV00] contains a collection of survey articles on semidefinite programming. The book by Boyd and Vanderbergue [BoV04] describes many applications, and contains a lot of material and references. The book by Ben-Tal, El Ghaoui, and Nemirovski [BGN09] focuses on robust optimization. The books by Hastie, Tibshirani, and Friedman [HTF09], and Sra, Nowozin, and Wright [SNW12] describe a lot of applications of convex optimization in machine learning.
1.1 (Support Vector Machines and Duality)

Consider the problem associated with a support vector machine,

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2}\|x\|^2 + \beta \sum_{i=1}^{m} \max \left\{ 0, 1 - b_i (c'_i x + y) \right\} \\
\text{subject to} & \quad x \in \mathbb{R}^n, \ y \in \mathbb{R},
\end{align*}
\]

with quadratic regularization, where \( \beta \) is a positive regularization parameter (cf. Example 1.3.2).

(a) Write the problem in the equivalent form

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2}\|x\|^2 + \beta \sum_{i=1}^{m} \xi_i \\
\text{subject to} & \quad x \in \mathbb{R}^n, \ y \in \mathbb{R}, \ 0 \leq \xi_i, \ 1 - b_i (c'_i x + y) \leq \xi_i, \ i = 1, \ldots, m.
\end{align*}
\]

Associate Lagrange multipliers \( \mu_i \geq 0 \) with the constraints \( 1 - b_i (c'_i x + y) \leq \xi_i \), and show that the dual function is given by

\[
q(\mu) = \begin{cases} 
\hat{q}(\mu) & \text{if } \sum_{j=1}^{m} \mu_j b_j = 0, \ 0 \leq \mu_i \leq \beta, \ i = 1, \ldots, m, \\
-\infty & \text{otherwise,}
\end{cases}
\]

where

\[
\hat{q}(\mu) = \sum_{i=1}^{m} \mu_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} b_i b_j c'_i c_j \mu_i \mu_j.
\]

Does the dual problem, viewed as the equivalent quadratic program

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} b_i b_j c'_i c_j \mu_i \mu_j - \sum_{i=1}^{m} \mu_i \\
\text{subject to} & \quad \sum_{j=1}^{m} \mu_j b_j = 0, \ 0 \leq \mu_i \leq \beta, \ i = 1, \ldots, m,
\end{align*}
\]

always have a solution? Is the solution unique? Note: The dual problem may have high dimension, but it has a generally more favorable structure than the primal. The reason is the simplicity of its constraint set, which makes it suitable for special types of quadratic programming methods, as well as the two-metric projection methods discussed in Section 1.6.1.
(b) Consider an alternative formulation where the variable $y$ is set to 0, leading to the problem

$$\text{minimize } \frac{1}{2} \|x\|^2 + \beta \sum_{i=1}^{m} \max \left\{ 0, 1 - b_i c_i^t x \right\}$$
subject to $x \in \mathbb{R}^n$.

Show that the dual problem should be modified so that the constraint $\sum_{j=1}^{p} \mu_j b_j = 0$ is not present, thus leading to a bound-constrained quadratic dual problem.

1.2 (Minimizing the Sum or the Maximum of Norms [LVB98])

Consider the problems

$$\text{minimize } \sum_{i=1}^{p} \|F_i x + g_i\|$$
subject to $x \in \mathbb{R}^n$, \hspace{1cm} (1.127)

and

$$\text{minimize } \max_{i=1, \ldots, p} \|F_i x + g_i\|$$
subject to $x \in \mathbb{R}^n$,

where $F_i$ and $g_i$ are given matrices and vectors, respectively. Convert these problems to second order cone form and derive the corresponding dual problems.

1.3 (Complex $l_1$ and $l_\infty$ Approximation [LVB98])

Consider the complex $l_1$ approximation problem

$$\text{minimize } \|Ax - b\|_1$$
subject to $x \in \mathbb{C}^n$,

where $\mathbb{C}^n$ is the set of $n$-dimensional vectors whose components are complex numbers. Show that it is a special case of problem (1.127) and derive the corresponding dual problem. Repeat for the complex $l_\infty$ approximation problem

$$\text{minimize } \|Ax - b\|_\infty$$
subject to $x \in \mathbb{C}^n$. 
1.4

The purpose of this exercise is to show that the SOCP can be viewed as a special case of SDP.

(a) Show that a vector \( x \in \mathbb{R}^n \) belongs to the second order cone if and only if the matrix
\[
\begin{pmatrix}
0 & 0 & \cdots & 0 & x_1 \\
0 & 0 & \cdots & 0 & x_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & x_{n-1} \\
x_1 & x_2 & \cdots & x_{n-1} & 0
\end{pmatrix}
\]
is positive semidefinite. *Hint:* We have that for any positive definite \( n \times n \) matrix \( A \), vector \( b \in \mathbb{R}^n \), and scalar \( d \), the matrix
\[
\begin{pmatrix}
A & b \\
b' & c
\end{pmatrix}
\]
is positive definite if and only if \( c - b' A^{-1} b > 0 \).

(b) Use part (a) to show that the primal SOCP can be written in the form of the dual SDP.

1.5 (Explicit Form of a Second Order Cone Problem)

Consider the SOCP (1.33).

(a) Partition the \( n_i \times (n + 1) \) matrices \( (A_i \ b_i) \) as
\[
( A_i \ b_i ) = \begin{pmatrix} D_i & d_i \\ p'_i & q_i \end{pmatrix}, \quad i = 1, \ldots, m,
\]
where \( D_i \) is an \( (n_i - 1) \times n \) matrix, \( d_i \in \mathbb{R}^{n_i-1} \), \( p_i \in \mathbb{R}^n \), and \( q_i \in \mathbb{R} \). Show that
\[
A_i x - b_i \in C_i \quad \text{if and only if} \quad \| D_i x - d_i \| \leq p_i' x - q_i,
\]
so we can write the SOCP (1.33) as
\[
\begin{align*}
\text{minimize} \quad & c' x \\
\text{subject to} \quad & \| D_i x - d_i \| \leq p_i' x - q_i, \quad i = 1, \ldots, m.
\end{align*}
\]

(b) Similarly partition \( \lambda_i \) as
\[
\lambda_i = \begin{pmatrix} \mu_i \\ \nu_i \end{pmatrix}, \quad i = 1, \ldots, m,
\]
where $\mu_i \in \mathbb{R}^{n_i-1}$ and $\nu_i \in \mathbb{R}$. Show that the dual problem (1.34) can be written in the form

$$\text{maximize } \sum_{i=1}^{m} (d'_i \mu_i + q_i \nu_i)$$

subject to $\sum_{i=1}^{m} (D'_i \mu_i + \nu_i p_i) = c, \quad \|\mu_i\| \leq \nu_i, \quad i = 1, \ldots, m$. \hfill (1.128)

(c) Show that the primal and dual interior point conditions for strong duality (Prop. 1.2.4) hold if there exist primal and dual feasible solutions $\bar{x}$ and $(\bar{\mu}_i, \bar{\nu}_i)$ such that

$$\|D_i x - d_i\| < p'_i x - q_i, \quad i = 1, \ldots, m,$$

and

$$\|\bar{\mu}_i\| < \bar{\nu}_i, \quad i = 1, \ldots, m,$$

respectively.

1.6 (Separable Conic Problems)

Consider the problem

$$\text{minimize } \sum_{i=1}^{m} f_i(x_i)$$

subject to $x \in S \cap C,$

where $x = (x_1, \ldots, x_m)$ with $x_i \in \mathbb{R}^{n_i}, i = 1, \ldots, m,$ and $f_i : \mathbb{R}^{n_i} \mapsto (-\infty, \infty]$ is a proper convex function for each $i$, and $S$ and $C$ are a subspace and a cone of $\mathbb{R}^{n_1 + \cdots + n_m}$, respectively. Show that a dual problem is

$$\text{maximize } \sum_{i=1}^{m} q_i(\lambda_i)$$

subject to $\lambda \in \hat{C} + S^\perp,$

where $\lambda = (\lambda_1, \ldots, \lambda_m), \hat{C}$ is the dual cone of $C$, and

$$q_i(\lambda_i) = \inf_{z_i \in \mathbb{R}} \{ f_i(z_i) - \lambda'_i z_i \}, \quad i = 1, \ldots, m.$$
1.7 (Convergence of Steepest Descent with Constant Stepsize)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable convex function such that for some constant $L > 0$, we have

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$ 

Show that every limit point of the generated sequence $\{x_k\}$ by the steepest descent iteration

$$x_{k+1} = x_k - \alpha \nabla f(x_k),$$

where $0 < \alpha < \frac{2}{L}$, minimizes $f$. *Hint:* Show that the cost function is reduced at each iteration according to

$$f(x_{k+1}) \leq f(x_k) - \alpha \left(1 - \frac{\alpha L}{2}\right) \|\nabla f(x_k)\|^2.$$

1.8 (A Stepsize Reduction Rule)

Consider minimization of a differentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$, using the iteration

$$x_{k+1} = x_k + \alpha_k d_k,$$

where $d_k$ is a descent direction and the stepsize $\alpha_k$ is determined by the following rule:

$$\alpha_{k+1} = \begin{cases} \alpha_k & \text{if } \nabla f(x_{k+1})'d_k \leq 0, \\ \beta \alpha_k & \text{otherwise}, \end{cases}$$

where $\beta \in (0, 1)$ is a fixed scalar and $\alpha_0$ is any positive scalar. Assume that for some constant $L > 0$, we have

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n,$$

and that there exist positive scalars $c_1, c_2$ such that for all $k$ we have

$$c_1 \|\nabla f(x_k)\|^2 \leq -\nabla f(x_k)'d_k, \quad \|d_k\|^2 \leq c_2 \|\nabla f(x_k)\|^2.$$

(a) Show that the stepsize is reduced after iteration $k$ if and only if the interval $I_k$ connecting $x_k$ and $x_{k+1}$ contains in its interior all the vectors $x \in I_k$ that minimize $f(x)$ over $x \in I_k$.

(b) Show that the stepsize will be constant after a finite number of iterations. *Hint:* The conditions on $\nabla f(x)$ and $d_k$ imply that

$$\nabla f(x_{k+1})'d_k \leq \nabla f(x_k)'d_k + \|\nabla f(x_{k+1}) - \nabla f(x_k)\| \cdot \|d_k\| \leq \nabla f(x_k)'d_k + \alpha_k L \|d_k\|^2 \leq -(c_1 - c_2 \alpha_k L) \|\nabla f(x_k)\|^2.$$

When the stepsize becomes small enough so that $c_1 - c_2 \alpha_k L \geq 0$ for some $k$, then $\nabla f(x_{k+1})'d_k \leq 0$ for all $k \geq k$ and no further reduction will ever be needed.

(c) Show that every limit point of the generated sequence $\{x_k\}$ minimizes $f$. *Hint:* Modify the inequality in the hint of Exercise 1.7.
1.9 (Steepest Descent Direction [BeM71])

Let \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) be a convex function, and let us view the steepest descent direction at \( x \) as the solution of the problem

\[
\begin{align*}
\text{minimize} & \quad f'(x; d) \\
\text{subject to} & \quad \|d\| \leq 1.
\end{align*}
\]

(1.129)

Show that this direction is \(-g^*\), where \( g^* \) is the vector of minimum norm in \( \partial f(x) \).

**Abbreviated Solution:** From Prop. 5.4.8/App. B (see also Section 2.1), \( f'(x; d) \) is the support function of the nonempty and compact subdifferential \( \partial f(x) \),

\[
f'(x; d) = \max_{g \in \partial f(x)} d'g, \quad \forall \, x, d \in \mathbb{R}^n.
\]

Since the sets \( \{d \mid \|d\| \leq 1\} \) and \( \partial f(x) \) are convex and compact, and the function \( d'g \) is linear in each variable when the other variable is fixed, by Prop. 5.5.3/App. B, we have

\[
\min_{\|d\| \leq 1} \max_{g \in \partial f(x)} d'g = \max_{g \in \partial f(x)} \min_{\|d\| \leq 1} d'g,
\]

and a saddle point exists. For any saddle point \((d^*, g^*)\), \( g^* \) maximizes the function \( \min_{\|d\| \leq 1} d'g = -\|g\| \) over \( \partial f(x) \), so \( g^* \) is the unique vector of minimum norm in \( \partial f(x) \). Moreover, \( d^* \) minimizes \( \max_{g \in \partial f(x)} d'g \) or equivalently \( f'(x; d) \) [by Eq. (1.129)] subject to \( \|d\| \leq 1 \) (so it is a direction of steepest descent), and minimizes \( d'g^* \) subject to \( \|d\| \leq 1 \), so it has the form

\[
d^* = -\frac{g^*}{\|g^*\|}
\]

[except if \( 0 \in \partial f(x) \), in which case \( d^* = 0 \)].

1.10 (Two-Metric Projection Methods for Bound Constraints [Ber82a], [Ber82b])

Let \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) be a differentiable convex function, and \( X \) be the set

\[
X = \left\{ (x^1, \ldots, x^n) \mid \underline{b}^i \leq x^i \leq \bar{b}^i, \quad i = 1, \ldots, n \right\},
\]

where \( \underline{b}^i \) and \( \bar{b}^i \), \( i = 1, \ldots, n \), are given scalars. Consider a scaled projection method of the form

\[
x_{k+1} = P_X \left( x_k - \alpha_k S_k \nabla f(x_k) \right),
\]

for minimization of \( f \) over \( X \), where \( S_k \) is a positive definite symmetric matrix.

(a) Construct an example of convex \( f \) and \( X \), and a positive definite \( S_k \), where \( x_k \) does not minimize \( f \) over \( X \), and \( f(x_{k+1}) > f(x_k) \) for all positive values of \( \alpha_k \).

(b) Let \( I_k = \{ i \mid x^i_k = \underline{b}^i \text{ or } x^i_k = \bar{b}^i \} \). Show that the iteration yields cost function descent if \( S_k \) is diagonal with respect to \( I_k \) in the sense that \( (S_k)_{ij} = 0 \) for all \( i, j \in I_k \) with \( i \neq j \). In particular, show that if \( x_k \) is not optimal, there exists \( \bar{\alpha}_k > 0 \) such that \( f(x_{k+1}) < f(x_k) \) for all \( \alpha_k \in (0, \bar{\alpha}_k] \).
1.11 (Proximal Gradient Method, $\ell_1$-Regularization, and the Shrinkage Operation)

Consider the problem

$$\minimize f(x) + \gamma \|x\|_1$$

subject to $x \in \mathbb{R}^n$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable convex function, $\| \cdot \|_1$ is the $\ell_1$ norm, and $\gamma > 0$. The proximal gradient iteration (1.84) is given by the gradient step

$$z_k = x_k - \alpha \nabla f(x_k),$$

followed by the proximal step

$$x_{k+1} \in \arg \min_{x \in \mathbb{R}^n} \left\{ \gamma \|x\|_1 + \frac{1}{2\alpha} \|x - z_k\|^2 \right\};$$

[cf. Eq. (1.85)]. Show that the proximal step can be performed separately for each coordinate $x_i$ of $x$, and is given by the so-called shrinkage operation:

$$x_{k+1}^i = \begin{cases} 
    z_k^i - \alpha \gamma & \text{if } z_k^i > \alpha \gamma, \\
    0 & \text{if } |z_k^i| \leq \alpha \gamma, \\
    z_k^i + \alpha \gamma & \text{if } z_k^i < -\alpha \gamma,
\end{cases} \quad i = 1, \ldots, n.$$

Note: Since the proximal step tends to set many coordinates $x_{k+1}^i$ to 0, it tends to produce "sparse" iterates.

1.12 (Incremental Methods - Computational Exercise)

This exercise relates to the (perhaps approximate) solution of a system of linear inequalities

$$c_i^i x \leq b_i, \quad i = 1, \ldots, m,$$

where $c_i$ are given vectors in $\mathbb{R}^n$ and $b_i$ are given scalars.

(a) Consider a variation of the Kaczmarz algorithm that operates in cycles as follows. At the end of cycle $k$, we set $x_{k+1} = \psi_{m,k}$, where $\psi_{m,k}$ is obtained after the $m$ steps

$$\psi_{1,k} = \psi_{i-1,k} - \frac{\alpha_k}{\|c_i\|^2} \max \left\{ 0, c_i^i \psi_{i-1,k} - b_i \right\} c_i, \quad i = 1, \ldots, m,$$

starting with $\psi_{0,k} = x_k$. Show that the algorithm can be viewed as an incremental gradient method for a suitable differentiable cost function. Show that if the given system of inequalities has a nonempty solution set $X^*$ and $\alpha_k \equiv 1$, then $\min_{x \in X^*} \|x - x^*\| \to 0$. What happens if the solution set is empty?

(b) Implement the algorithm of (a) for two examples where $n = 2$ and $m = 100$. In the first example, the vectors $c_i$ have the form $c_i = (\xi_i, \zeta_i)$,
where \( \xi, \zeta, \) as well as \( b_i, \) are chosen randomly and independently from \([-1, 1]\) according to a uniform distribution. In the second example, the vectors \( c_i \) have the form \( c_i = (\xi, \zeta, b_i), \) where \( \xi, \zeta, \) as well as \( b_i, \) are chosen randomly and independently within \([-1, 1]\) according to a uniform distribution, while \( b_i \) is chosen randomly and independently within \([0, 1]\) according to a uniform distribution. Experiment with different starting points and stepsize choices, and deterministic and randomized orders of selection of the indexes \( i \) for iteration. Explain your experimental results in terms of the theoretical behavior described in Section 1.6.

(c) Consider the minimization of

\[
f(x) = R(x) + \frac{1}{2} \sum_{i=1}^{m} \left( \max\{0, c'_i x - b_i\} \right)^2,
\]

where \( R(x) \) is a positive definite quadratic regularization function. Implement the incremental Newton method of Section 1.6 for the two examples of part (b), and compare with the results of the algorithm of part (a). (Note that for this problem, the Hessian of the cost function is not continuous, which tends to interfere with the operation of Newton’s method.)

(d) Consider the cost function

\[
f(x) = \gamma \| x \|_1 + \frac{1}{2} \sum_{i=1}^{m} \max\{0, c'_i x - b_i\}.
\]

Implement the exponential smoothing approach of Section 1.7.4 and the incremental Newton method of Section 1.6 for the two examples of part (b).

1.13 (Convergence Rate of the Kaczmarz Algorithm with Random Projection [StV09])

Consider a consistent system of linear equations \( c'_i x = b_i, \ i = 1, \ldots, m, \) and assume for convenience that the vectors \( c_i \) have been scaled so that \( \| c_i \| = 1 \) for all \( i. \) A randomized version of the Kaczmarz method is given by

\[
x_{k+1} = x_k - (c'_{i_k} x - b_{i_k}) c_{i_k},
\]

where \( i_k \) is an index randomly chosen from the set \( \{1, \ldots, m\} \) with equal probabilities \( 1/m, \) independently of previous choices. Let \( P(x) \) denote the Euclidean projection of a vector \( x \in \mathbb{R}^n \) onto the set of solutions of the system, and let \( C \) be the matrix whose rows are \( c_1, \ldots, c_m. \) Show that

\[
E[\| x_{k+1} - P(x_{k+1}) \|^2] \leq \left( 1 - \frac{\lambda_{\text{min}}}{m} \right) E[\| x_k - P(x_k) \|^2],
\]

where \( \lambda_{\text{min}} \) is the minimum eigenvalue of the matrix \( C'C. \) Hint: Show that

\[
\| x_{k+1} - P(x_{k+1}) \|^2 \leq \| x_{k+1} - P(x_k) \|^2 = \| x_k - P(x_k) \|^2 - (c'_{i_k} x_{i_k} - b_{i_k})^2,
\]

and take conditional expectation of both sides to show that

\[
E[\| x_{k+1} - P(x_{k+1}) \|^2 | x_k] \leq \| x_k - P(x_k) \|^2 - \frac{1}{m} \| C x_k - b \|^2
\]

\[
\leq \left( 1 - \frac{\lambda_{\text{min}}}{m} \right) \| x_k - P(x_k) \|^2.
\]
1.14 (Limit Cycle of Incremental Gradient Method [Luo91])

Consider the scalar least squares problem

\[
\begin{align*}
    &\text{minimize } \frac{1}{2}\left\{(b_1 - x)^2 + (b_2 - x)^2\right\} \\
    &\text{subject to } x \in \mathbb{R},
\end{align*}
\]

where \(b_1\) and \(b_2\) are given scalars, and the incremental gradient algorithm that generates \(x_{k+1}\) from \(x_k\) according to

\[
x_{k+1} = \psi_k - \alpha(\psi_k - b_2),
\]

where

\[
\psi_k = x_k - \alpha(x_k - b_1),
\]

and \(\alpha\) is a positive stepsizes. Assuming that \(\alpha < 1\), show that \(\{x_k\}\) and \(\{\psi_k\}\) converge to limits \(x(\alpha)\) and \(\psi(\alpha)\), respectively. However, unless \(b_1 = b_2\), \(x(\alpha)\) and \(\psi(\alpha)\) are neither equal to each other, nor equal to the least squares solution \(x^* = (b_1 + b_2)/2\). Verify that

\[
\lim_{\alpha \to 0} x(\alpha) = \lim_{\alpha \to 0} \psi(\alpha) = x^*.
\]

1.15 (Convergence of Incremental Gradient Method for Linear Least Squares problems [BeT96], [Ber99])

Consider the linear least squares problem of minimizing

\[
f(x) = \frac{1}{2}\sum_{i=1}^{m} \|g_i(x)\|^2
\]

over \(x \in \mathbb{R}^n\), where \(g_i : \mathbb{R}^n \to \mathbb{R}^{n_i}\) is the linear function

\[
g_i(x) = z_i - C_ix, \quad i = 1, \ldots, m.
\]

The incremental gradient method operates in cycles with \(x_k\), the vector at the start of cycle \(k\), given by

\[
x_{k+1} = x_k + \alpha_k \sum_{i=1}^{m} C_i'(z_i - C_i\psi_{i-1}),
\]

where \(\psi_0 = x_0\) and

\[
\psi_i = \psi_{i-1} + \alpha_k C_i'(z_i - C_i\psi_{i-1}), \quad i = 1, \ldots, m.
\]

Assume that \(\sum_{i=1}^{m} C_i' C_i\) is a positive definite matrix and let \(x^*\) be the optimal solution. Then:

(a) There exists \(\alpha > 0\) such that if \(\alpha_k\) is equal to some constant \(\alpha \in (0, \alpha]\) for all \(k\), \(\{x_k\}\) converges to some vector \(x(\alpha)\). Furthermore, the error \(\|x_k - x(\alpha)\|\) converges to 0 linearly. In addition, we have \(\lim_{\alpha \to 0} x(\alpha) = x^*\). Hint: Show that the mapping that produces \(x_{k+1}\) starting from \(x_k\) is a contraction mapping for \(\alpha\) sufficiently small.

(b) If \(\alpha_k > 0\) for all \(k\), and

\[
\alpha_k \to 0, \quad \sum_{k=0}^{\infty} \alpha_k = \infty,
\]

then \(\{x_k\}\) converges to \(x^*\). Hint: Use Prop. A.4.2 of Appendix A.
1.16 (Neighborhood Convergence of Incremental Gradient Method [Ber99], [NeB00])

Consider the gradient method
\[ x_{k+1} = x_k - \alpha \nabla f_k(x_k) \quad k = 0, 1, \ldots, \]

where \( f_0, f_1, \ldots, \) are quadratic functions with eigenvalues lying within some interval \([\gamma, \Gamma]\), where \( \gamma > 0 \). Suppose that for a given \( \epsilon > 0 \), there is a vector \( x^* \) such that
\[ \| \nabla f_k(x^*) \| \leq \epsilon, \quad \forall \ k = 0, 1, \ldots. \]

Show that for all \( \alpha \) with \( 0 < \alpha \leq 2/(\gamma + \Gamma) \), we have
\[ \limsup_{k \to \infty} \| x_k - x^* \| \leq \frac{2\epsilon}{\gamma}. \]

**Hint:** Let \( Q_k \) be the positive definite symmetric matrix corresponding to \( f_k \), and write
\[ x_{k+1} - x^* = (I - \alpha Q_k)(x_k - x^*) - \alpha \nabla f_k(x^*). \]

Use this relation to show that
\[ \| x_k - x^* \| > \frac{2\epsilon}{\gamma} \Rightarrow \| x_{k+1} - x^* \| < \left( 1 - \frac{\alpha \gamma}{2} \right) \| x_k - x^* \|. \]

while
\[ \| x_k - x^* \| \leq \frac{2\epsilon}{\gamma} \Rightarrow \| x_{k+1} - x^* \| \leq \frac{2\epsilon}{\gamma}. \]
REFERENCES


