22

Algorithms for Constrained Optimization

22.1 INTRODUCTION

In Part II we discussed algorithms for solving *unconstrained* optimization problems. This chapter is devoted to a treatment of some simple algorithms for solving special *constrained* optimization problems. The methods here build on those of Part II.

We begin our presentation in the next section with a discussion of *projected methods*, including a treatment of projected gradient methods for problems with linear equality constraints. We then consider *penalty methods*. This chapter is intended as an introduction to some basic ideas underlying methods for solving constrained optimization problems. For an in-depth coverage of the subject, we refer the reader to [8].

22.2 PROJECTIONS

The optimization algorithms considered in Part II have the general form

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)},$$

where $\boldsymbol{d}^{(k)}$ is typically a function of $\nabla f(\boldsymbol{x}^{(k)})$. The value of $\boldsymbol{x}^{(k)}$ is not constrained to lie inside any particular set. Such an algorithm is not immediately applicable to solving constrained optimization problems in which the decision variable is required to lie within a prespecified constraint set.

Consider the optimization problem

$$\text{minimize} \qquad f(\boldsymbol{x})$$

$$\text{subject to} \quad x \in \Omega.$$

If we use the algorithm above to solve this constrained problem, the iterates $x^{(k)}$ may not satisfy the constraints. Therefore, we need to modify the algorithms to take into account the presence of the constraints. A simple modification involves the introduction of a *projection*. The idea is as follows. If $x^{(k)} + \alpha_k d^{(k)}$ is in Ω, then we set $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$ as usual. If, on the other hand, $x^{(k)} + \alpha_k d^{(k)}$ is not in Ω, then we "project" it back into Ω before setting $x^{(k+1)}$.

To illustrate the projection method, consider the case where the constraint set $\Omega \subset \mathbb{R}^n$ is given by

$$\Omega = \{x : l_i \le x_i \le u_i,\ i = 1, \ldots, n\}.$$

In this case, Ω is a "box" in $\mathbb{R}^n$. Given a point $x \in \mathbb{R}^n$, define $y = \Pi[x] \in \mathbb{R}^n$ by

$$y_i = \begin{cases} u_i & \text{if } x_i > u_i \\ x_i & \text{if } l_i \le x_i \le u_i \\ l_i & \text{if } x_i < l_i \end{cases}.$$

The point $\Pi[x]$ is called the *projection* of x onto Ω. Note that $\Pi[x]$ is actually the "closest" point in Ω to x. Using the projection operator Π, we can modify the previous unconstrained algorithm as follows:

$$x^{(k+1)} = \Pi[x^{(k)} + \alpha_k d^{(k)}].$$

Note that the iterates $x^{(k)}$ now all lie inside Ω. We call the above algorithm a *projected* algorithm.

In the more general case, we can define the projection onto Ω:

$$\Pi[x] = \arg\min_{z \in \Omega} \|z - x\|.$$

In this case, $\Pi[x]$ is again the "closest" point in Ω to x. This projection operator is well defined only for certain types of constraint sets—for example, closed convex sets. For some sets Ω, the "arg min" above is not well defined. If the projection Π is well defined, we can similarly apply the projected algorithm

$$x^{(k+1)} = \Pi[x^{(k)} + \alpha_k d^{(k)}].$$

In some cases, there is a formula for computing $\Pi[x]$. For example, if Ω is a "box" constraint set as described above, then the formula given previously can be used. Another example is where Ω is a linear variety (plane), which is discussed in the next section. In general, even if the projection Π is well defined, the computation of $\Pi[x]$ given x may not be easy. Often, the projection $\Pi[x]$ may have to be computed numerically. However, the numerical computation of $\Pi[x]$ itself entails solving an optimization algorithm. Indeed, the computation of $\Pi[x]$ may be as difficult as the original optimization problem, as is the case in the following example:

$$\begin{array}{ll} \text{minimize} & \|x\|^2 \\ \text{subject to} & x \in \Omega. \end{array}$$

Note that the solution to the problem in this case can be written as $\Pi[0]$. Therefore, if $0 \notin \Omega$, the computation of a projection is equivalent to solving the given optimization problem.

22.3 PROJECTED GRADIENT METHODS

In this section, we consider optimization problems of the form

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b, \end{array}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $A \in \mathbb{R}^{m \times n}$, $m < n$, $\operatorname{rank} A = m$, $b \in \mathbb{R}^m$. We assume throughout that $f \in \mathcal{C}^1$. In the above problem, the constraint set is $\Omega = \{x : Ax = b\}$. The specific structure of the constraint set allows us to compute the projection operator Π using the *orthogonal projector* (see Section 3.3). Specifically, $\Pi[x]$ can be defined using the orthogonal projector matrix P given by

$$P = I_n - A^T(AA^T)^{-1}A$$

(see Example 12.4). Two important properties of the orthogonal projector P that we use in this section are (see Theorem 3.5):

1. $P = P^T$; and
2. $P^2 = P$.

Another property of the orthogonal projector that we need in our discussion is given in the following lemma.

Lemma 22.1 *Let $v \in \mathbb{R}^n$. Then, $Pv = 0$ if and only if $v \in \mathcal{R}(A^T)$. In other words, $\mathcal{N}(P) = \mathcal{R}(A^T)$. Moreover, $Av = 0$ if and only if $v \in \mathcal{R}(P)$, that is, $\mathcal{N}(A) = \mathcal{R}(P)$.* □

Proof. $\Rightarrow$: We have

$$\begin{aligned} Pv &= (I_n - A^T(AA^T)^{-1}A)v \\ &= v - A^T(AA^T)^{-1}Av. \end{aligned}$$

If $Pv = 0$, then

$$v = A^T(AA^T)^{-1}Av$$

and hence $v \in \mathcal{R}(A^T)$.

$\Leftarrow$: Suppose there exists $u \in \mathbb{R}^m$ such that $v = A^T u$. Then,

$$\begin{aligned} Pv &= (I_n - A^T(AA^T)^{-1}A)A^T u \\ &= A^T u - A^T(AA^T)^{-1}AA^T u \\ &= 0. \end{aligned}$$

Hence, we have proved that $\mathcal{N}(\boldsymbol{P}) = \mathcal{R}(\boldsymbol{A}^T)$.

Using a similar argument as above, we can show that $\mathcal{N}(\boldsymbol{A}) = \mathcal{R}(\boldsymbol{P})$. ∎

Recall that in unconstrained optimization, the first-order necessary condition for a point $\boldsymbol{x}^*$ to be a local minimizer is $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ (see Section 6.2). In optimization problems with equality constraints, the Lagrange condition plays the role of the first-order necessary condition (see Section 19.4). When the constraint set takes the form $\{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$, the Lagrange condition can be written as $\boldsymbol{P}\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$, as stated in the following proposition.

Proposition 22.1 *Let $\boldsymbol{x}^* \in \mathbb{R}^n$ be a feasible point. Then, $\boldsymbol{P}\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ if and only if $\boldsymbol{x}^*$ satisfies the Lagrange condition.* □

Proof. By Lemma 22.1, $\boldsymbol{P}\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ if and only if we have $\nabla f(\boldsymbol{x}^*) \in \mathcal{R}(\boldsymbol{A}^T)$. This is equivalent to the condition that there exists $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ such that $\nabla f(\boldsymbol{x}^*) + \boldsymbol{A}^T\boldsymbol{\lambda}^* = \boldsymbol{0}$, which, together with the feasibility equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, constitutes the Lagrange condition. ∎

In the remainder of this section, we discuss the projection method applied specifically to the gradient algorithm (see Chapter 8). Recall that the vector $-\nabla f(\boldsymbol{x})$ points in the direction of maximum rate of decrease of f at $\boldsymbol{x}$. This was the basis for gradient methods for unconstrained optimization, which have the form $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \nabla f(\boldsymbol{x}^{(k)})$, where α_k is the step size. The choice of the step size α_k depends on the particular gradient algorithm. For example, recall that in the steepest descent algorithm, $\alpha_k = \arg\min_{\alpha \ge 0} f(\boldsymbol{x}^{(k)} - \alpha\nabla f(\boldsymbol{x}^{(k)}))$.

The projected version of the gradient algorithm has the form

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{\Pi}[\boldsymbol{x}^{(k)} - \alpha_k \nabla f(\boldsymbol{x}^{(k)})].$$

We refer to the above as the *projected gradient algorithm*. It turns out that we can express the projection $\boldsymbol{\Pi}$ in terms of the matrix $\boldsymbol{P}$ as follows:

$$\boldsymbol{\Pi}[\boldsymbol{x}^{(k)} - \alpha_k \nabla f(\boldsymbol{x}^{(k)})] = \boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}),$$

assuming $\boldsymbol{x}^{(k)} \in \Omega$. Although the above formula can be derived algebraically (see Exercise 22.1), it is more insightful to derive the formula using a geometric argument, as follows. In our constrained optimization problem, the vector $-\nabla f(\boldsymbol{x})$ is not necessarily a feasible direction. In other words, if $\boldsymbol{x}^{(k)}$ is a feasible point and we apply the algorithm $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \nabla f(\boldsymbol{x}^{(k)})$, then $\boldsymbol{x}^{(k+1)}$ need not be feasible. This problem can be overcome by replacing $-\nabla f(\boldsymbol{x}^{(k)})$ by a vector that points in a feasible direction. Note that the set of feasible directions is simply the nullspace $\mathcal{N}(\boldsymbol{A})$ of the matrix $\boldsymbol{A}$. Therefore, we should first project the vector $-\nabla f(\boldsymbol{x})$ onto $\mathcal{N}(\boldsymbol{A})$. This projection is equivalent to multiplication by the matrix $\boldsymbol{P}$. In summary, in the projection gradient algorithm, we update $\boldsymbol{x}^{(k)}$ according to the equation

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}).$$

The projected gradient algorithm has the following property.

Proposition 22.2 *In a projected gradient algorithm, if $\boldsymbol{x}^{(0)}$ is feasible, then each $\boldsymbol{x}^{(k)}$ is feasible, that is, for each $k \ge 0$, $\boldsymbol{A}\boldsymbol{x}^{(k)} = \boldsymbol{b}$.* □

Proof. We proceed by induction. The result holds for $k = 0$ by assumption. Suppose now that $\boldsymbol{A}\boldsymbol{x}^{(k)} = \boldsymbol{b}$. We now show that $\boldsymbol{A}\boldsymbol{x}^{(k+1)} = \boldsymbol{b}$. To show this, first observe that $\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}) \in \mathcal{N}(\boldsymbol{A})$. Therefore,

$$\begin{aligned} \boldsymbol{A}\boldsymbol{x}^{(k+1)} &= \boldsymbol{A}(\boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)})) \\ &= \boldsymbol{A}\boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{A}\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}) \\ &= \boldsymbol{b}, \end{aligned}$$

which completes the proof. ∎

The projected gradient algorithm updates $\boldsymbol{x}^{(k)}$ in the direction of $-\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)})$. This vector points in the direction of maximum rate of decrease of f at $\boldsymbol{x}^{(k)}$ along the surface defined by $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, as described in the following argument. Let $\boldsymbol{x}$ be any feasible point and $\boldsymbol{d}$ a feasible direction such that $\|\boldsymbol{d}\| = 1$. The rate of increase of f at $\boldsymbol{x}$ in the direction $\boldsymbol{d}$ is $\langle \nabla f(\boldsymbol{x}), \boldsymbol{d}\rangle$. Next, we note that because $\boldsymbol{d}$ is a feasible direction, it lies in $\mathcal{N}(\boldsymbol{A})$ and hence by Lemma 22.1, we have $\boldsymbol{d} \in \mathcal{R}(\boldsymbol{P}) = \mathcal{R}(\boldsymbol{P}^T)$. So, there exists $\boldsymbol{v}$ such that $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{v}$. Hence,

$$\langle \nabla f(\boldsymbol{x}), \boldsymbol{d}\rangle = \langle \nabla f(\boldsymbol{x}), \boldsymbol{P}^T\boldsymbol{v}\rangle = \langle \boldsymbol{P}\nabla f(\boldsymbol{x}), \boldsymbol{v}\rangle.$$

By the Cauchy-Schwarz inequality,

$$\langle \boldsymbol{P}\nabla f(\boldsymbol{x}), \boldsymbol{v}\rangle \le \|\boldsymbol{P}\nabla f(\boldsymbol{x})\|\|\boldsymbol{v}\|$$

with equality if and only if the direction of $\boldsymbol{v}$ is parallel with the direction of $\boldsymbol{P}\nabla f(\boldsymbol{x})$. Therefore, the vector $-\boldsymbol{P}\nabla f(\boldsymbol{x})$ points in the direction of maximum rate of decrease of f at $\boldsymbol{x}$ among all feasible directions.

Following the discussion in Chapter 8 for gradient methods in unconstrained optimization, we suggest the following gradient method for our constrained problem. Suppose we have a starting point $\boldsymbol{x}^{(0)}$, which we assume is feasible, that is, $\boldsymbol{A}\boldsymbol{x}^{(0)} = \boldsymbol{b}$. Consider the point $\boldsymbol{x} = \boldsymbol{x}^{(0)} - \alpha\boldsymbol{P}\nabla f(\boldsymbol{x}^{(0)})$, where $\alpha \in \mathbb{R}$. As usual, the scalar α is called the step size. By the above discussion, $\boldsymbol{x}$ is also a feasible point. Using a Taylor series expansion of f about $\boldsymbol{x}^{(0)}$, and the fact that $\boldsymbol{P} = \boldsymbol{P}^2 = \boldsymbol{P}^T\boldsymbol{P}$, we get

$$\begin{aligned} f(\boldsymbol{x}^{(0)} - \alpha\boldsymbol{P}\nabla f(\boldsymbol{x}^{(0)})) &= f(\boldsymbol{x}^{(0)}) - \alpha\nabla f(\boldsymbol{x}^{(0)})^T\boldsymbol{P}\nabla f(\boldsymbol{x}^{(0)}) + o(\alpha) \\ &= f(\boldsymbol{x}^{(0)}) - \alpha\|\boldsymbol{P}\nabla f(\boldsymbol{x}^{(0)})\|^2 + o(\alpha). \end{aligned}$$

Thus, if $\boldsymbol{P}\nabla f(\boldsymbol{x}^{(0)}) \ne 0$, that is, $\boldsymbol{x}^{(0)}$ does not satisfy the Lagrange condition, then we can choose an α sufficiently small such that $f(\boldsymbol{x}) < f(\boldsymbol{x}^{(0)})$, which means that $\boldsymbol{x} = \boldsymbol{x}^{(0)} - \alpha\boldsymbol{P}\nabla f(\boldsymbol{x}^{(0)})$ is an improvement over $\boldsymbol{x}^{(0)}$. This is the basis for the projected gradient algorithm $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)})$, where the initial point $\boldsymbol{x}^{(0)}$ satisfies $\boldsymbol{A}\boldsymbol{x}^{(0)} = \boldsymbol{b}$, and α_k is some step size. As for unconstrained gradient methods, the choice of α_k determines the behavior of the algorithm. For small step sizes, the algorithm progresses slowly, while large step sizes may result in a zig-zagging path. A well-known variant of the projected gradient algorithm is the *projected steepest descent algorithm*, where α_k is given by

$$\alpha_k = \arg\min_{\alpha \geq 0} f(\boldsymbol{x}^{(k)} - \alpha \boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)})).$$

The following theorem states that the projected steepest descent algorithm is a descent algorithm, in the sense that at each step the value of the objective function decreases.

Theorem 22.1 *If $\{\boldsymbol{x}^{(k)}\}$ is the sequence of points generated by the projected steepest descent algorithm and if $\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}) \neq \boldsymbol{0}$, then $f(\boldsymbol{x}^{(k+1)}) < f(\boldsymbol{x}^{(k)})$.* □

Proof. First, recall that

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}),$$

where $\alpha_k \geq 0$ is the minimizer of

$$\phi_k(\alpha) = f(\boldsymbol{x}^{(k)} - \alpha \boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}))$$

over all $\alpha \geq 0$. Thus, for $\alpha \geq 0$, we have

$$\phi_k(\alpha_k) \leq \phi_k(\alpha).$$

By the chain rule,

$$\begin{aligned}\phi_k'(0) &= \frac{d\phi_k}{d\alpha}(0)\\ &= -\nabla f(\boldsymbol{x}^{(k)} - 0\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}))^T \boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)})\\ &= -\nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}).\end{aligned}$$

Using the fact that $\boldsymbol{P} = \boldsymbol{P}^2 = \boldsymbol{P}^T\boldsymbol{P}$, we get

$$\phi_k'(0) = -\nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{P}^T\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}) = -\|\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)})\|^2 < 0,$$

because $\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}) \neq \boldsymbol{0}$ by assumption. Thus, there exists $\bar{\alpha} > 0$ such that $\phi_k(0) > \phi_k(\alpha)$ for all $\alpha \in (0, \bar{\alpha}]$. Hence,

$$f(\boldsymbol{x}^{(k+1)}) = \phi_k(\alpha_k) \leq \phi_k(\bar{\alpha}) < \phi_k(0) = f(\boldsymbol{x}^{(k)})$$

and the proof of the theorem is completed. ∎

In the above theorem we needed the assumption that $\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}) \neq \boldsymbol{0}$ to prove that the algorithm possesses the descent property. If for some k, we have $\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}) = \boldsymbol{0}$, then by Proposition 22.1 the point $\boldsymbol{x}^{(k)}$ satisfies the Lagrange condition. This condition can be used as a stopping criterion for the algorithm. Note that in this case, $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$. For the case where f is a convex function, the condition $\boldsymbol{P}\nabla f(\boldsymbol{x}^{(k)}) = \boldsymbol{0}$ is, in fact, equivalent to $\boldsymbol{x}^{(k)}$ being a global minimizer of f over the constraint set $\{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$. We show this in the following proposition.

Proposition 22.3 *The point $\boldsymbol{x}^* \in \mathbb{R}^n$ is a global minimizer of a convex function f over $\{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$ if and only if $\boldsymbol{P}\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$.* □

Proof. We first write $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$. Then, the constraints can be written as $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}$, and the problem is of the form considered in previous chapters. Note that $D\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{A}$. Hence, $\boldsymbol{x}^* \in \mathbb{R}^n$ is a global minimizer of f if and only if the Lagrange condition holds (see Theorem 21.7). By Proposition 22.1, this is true if and only if $\boldsymbol{P}\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$, and the proof is completed. ∎

For an application of the projected steepest descent algorithm to minimum fuel and minimum amplitude control problems in linear discrete systems, see [57].

22.4 PENALTY METHODS

In this section, we consider constrained optimization problems of the form

$$\begin{aligned}\text{minimize} \quad & f(\boldsymbol{x})\\ \text{subject to} \quad & g_1(\boldsymbol{x}) \leq 0\\ & g_2(\boldsymbol{x}) \leq 0\\ & \quad\vdots\\ & g_p(\boldsymbol{x}) \leq 0,\end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, p$. Considering only inequality constraints is not restrictive, because an equality constraint of the form $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}$ is equivalent to the inequality constraint $\|\boldsymbol{h}(\boldsymbol{x})\|^2 \leq 0$ (however, see Exercise 20.21 for a caveat). We now discuss a method for solving the above constrained optimization problem using techniques from unconstrained optimization. Specifically, we approximate the constrained optimization problem above by an unconstrained optimization problem

$$\text{minimize } f(\boldsymbol{x}) + \gamma P(\boldsymbol{x}),$$

where $\gamma \in \mathbb{R}$ is a positive constant, and $P : \mathbb{R}^n \to \mathbb{R}$ is a given function. We then solve the associated unconstrained optimization problem, and use the solution as an approximation to the minimizer of the original problem. The constant γ is called the *penalty parameter*, and the function P is called the *penalty function*. Formally, we define a penalty function as follows.

Definition 22.1 A function $P : \mathbb{R}^n \to \mathbb{R}$ is called a *penalty function* for the above constrained optimization problem if it satisfies the following three conditions:

1. P is continuous;