Chapter 2
Numerical Optimization

2.1 Mathematical Background

In this chapter, we present the necessary mathematical background on numerical optimization [11], which is used in this book as a fundamental tool for extremum seeking control. We first review concepts related to continuity, differentiability, and optimality. These concepts will then allow us to present the line search and trust-region unconstrained optimization methods.

Definition 2.1.1 (Sequence) A sequence of real numbers $\{x_k^s | k = 1, 2, \dots\}$ (also represented as $\{x_k^s\}$) is said to *converge* to a limit $x \in \mathbb{R}$ if for every $\varepsilon > 0$ there exists some positive integer K (that depends on ε) such that, for every $k \geq K$, we have $|x_k^s - x| < \varepsilon$. For such a convergent sequence we may also write $\lim_{k\to\infty} x_k^s = x$.

Similarly, a sequence $\{x_k^s\}$ of vectors $x_k^s \in \mathbb{R}^n$ is said to converge to a limit $x \in \mathbb{R}^n$ if the ith coordinate of x_k^s converges to the ith coordinate of x for $1 \leq i \leq n$. In this case, the notation $\lim_{k\to\infty} x_k^s = x$ is employed as well.

Definition 2.1.2 (Continuously Differentiable Functions) Consider the function $f : \mathbb{R} \to \mathbb{R}$. This function is said to be *continuously differentiable* if its derivative f' exists and is continuous. Alternatively, one can say that f *belongs to class* C^1, or $f \in C^1$.

Similarly, if f', f'', $\dots$, $f^{(k)}$ exist and are continuous, then f is said to belong to *class* C^k, or $f \in C^k$. Finally, if f has continuous derivatives of all orders, then it is said to be *smooth*, or $f \in C^\infty$.

For a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$, it is said to belong to class C^k if each of its derivatives up to kth order exists and is continuous.

In addition to these continuity concepts, there is another type of continuity we often need, called Lipschitz continuity. Lipschitz continuity is a smoothness condition, stronger than regular continuity, that imposes a limit on the function's growth rate.

C. Zhang, R. Ordóñez, *Extremum-Seeking Control and Applications*,
Advances in Industrial Control,
DOI 10.1007/978-1-4471-2224-1_2, © Springer-Verlag London Limited 2012

Definition 2.1.3 (Lipschitz Continuity) Consider the function $f : \mathbb{R}^n \to \mathbb{R}^m$. This function is said to be *Lipschitz continuous* if there exists a constant $L > 0$ such that

$$\left\| f(x) - f(y) \right\| \leq L \|x - y\|. \tag{2.1}$$

The following cases can be distinguished:

1. If condition (2.1) is satisfied for every $x, y \in \mathbb{R}^n$ and with *the same* constant L, then f is said to be *globally Lipschitz continuous*, or simply *Lipschitz*.
2. If condition (2.1) is satisfied for every $x, y \in D$, where $D \in \mathbb{R}^n$ and with the same constant L, then f is said to be *Lipschitz in* D.
3. If condition (2.1) is satisfied for every $x, y \in D$, where $D \in \mathbb{R}^n$ but not with the same constant L, which can depend on x, then f is said to be *locally Lipschitz in* D.

For the study of nonlinear systems it is often useful to define functions that map one set to another, with the purpose of normalizing the form of a system's dynamics. To this end, we require the following concept.

Definition 2.1.4 (Diffeomorphism) Consider a function $T : D \to M$ mapping the domain $D \in \mathbb{R}^n$ to the domain $M \in \mathbb{R}^n$. The function T is called a *diffeomorphism* if T is continuously differentiable, and its inverse T^{-1} exists and is continuously differentiable.

Definition 2.1.5 (Matrix Concepts) An $n \times n$ matrix A with real elements is *symmetric* if it is equal to its transpose, or $A = A^\top$. The symmetric matrix A is said to be *positive definite* if $x^\top Ax > 0$ for all $x \in \mathbb{R}^n$, $x \neq 0$. A is called *positive semi-definite* if $x^\top Ax \geq 0$. A is *negative definite* if $-A$ is positive definite.

Theorem 2.1.6 *The following statements hold for any non-zero symmetric matrix* $A \in \mathbb{R}^{n \times n}$ [8]:

1. *Its eigenvalues are all real-valued, and the corresponding* n *eigenvectors are real, non-zero and mutually orthogonal.*
2. A *is positive definite* if and only if *all its eigenvalues are strictly positive.*
3. A *is positive semi-definite* if and only if *all its eigenvalues are non-negative.*

Given a function $J : \mathbb{R}^n \to \mathbb{R}$ with $J \in C^1$, denote

$$g(x) = \nabla J(x) = \left[\frac{\partial J(x)}{\partial x_1}, \ldots, \frac{\partial J(x)}{\partial x_n} \right]^\top$$

as the *gradient* of J evaluated at the point $x \in \mathbb{R}^n$. In the case of $x = x(t)$ for $t \in \mathbb{R}^m$, by using the chain rule we have

$$\nabla J\big(x(t)\big) = \sum_{i=1}^{n} \frac{\partial J}{\partial x_i} \nabla x_i(t) = \sum_{i=1}^{n} \frac{\partial J}{\partial x_i} \left[\frac{\partial x_i(t)}{\partial t_1}, \ldots, \frac{\partial x_i(t)}{\partial t_m} \right]^\top.$$

If $J \in C^2$, let $H(x) = \nabla^2 J(x)$ be the *Hessian matrix*. The (i, j)th component of H is given by $\partial^2 J(x)/\partial x_i \partial x_j$, $1 \le i, j \le n$. The square matrix H is symmetric.

Theorem 2.1.7 (Taylor's Theorem [11]) *Let $J : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable and $p \in \mathbb{R}^n$. Then*

$$J(x + p) = J(x) + \nabla J^\top (x + \alpha p) p,$$

for some $\alpha \in [0, 1]$. Moreover, if J is twice continuously differentiable then

$$\nabla J(x + p) = \nabla J(x) + \left(\int_0^1 \nabla^2 J(x + \tau p)\, d\tau \right) p,$$

and the following three statements hold:

1. *There exists some $\alpha \in [0, 1]$ such that, for any $p \in \mathbb{R}^n$,*

$$J(x + p) = J(x) + \nabla J(x)^\top p + \frac{1}{2} p^\top \nabla^2 J(x + \alpha p) p.$$

2. *For any $p \in \mathbb{R}^n$,*

$$J(x + p) = J(x) + \nabla J(x)^\top p + \frac{1}{2} p^\top \nabla^2 J(x) p + O\big(\|p\|^2\big).$$

3. *For any $p \in \mathbb{R}^n$,*

$$J(x + p) = J(x) + \nabla J(x)^\top p + \frac{1}{2} p^\top \left(\int_0^1 \left(\int_0^t \nabla^2 J(x + \tau y)\, d\tau \right) dt \right) p.$$

Note that a set $S \subseteq \mathbb{R}^n$ is *convex* if for any $x, y \in S$ we have $\beta x + (1 - \beta) y \in S$ for all $\beta \in [0, 1]$. The set S is *closed* if it contains all of its limit points, and it is *bounded* if all its elements have coordinates whose magnitude is less than some finite $d > 0$. Further, S is *compact* if every sequence of elements of S has a subsequence that converges to an element of S. The set S is compact if and only if it is closed and bounded.

A function J is a *convex function* if its domain is convex and if for any x, y in this domain we have $J(\beta x + (1 - \beta) y) \le \beta J(x) + (1 - \beta) J(y)$ for all $\beta \in [0, 1]$.

Definition 2.1.8 (Global Minimizer) Let J be defined on $S \subseteq \mathbb{R}^n$. The point $x^* \in S$ is a *global minimizer* of J if $J(x^*) \le J(x)$ for all $x \in S$; it is a *strict global minimizer* of J if $J(x^*) < J(x)$ for all $x \in S$, $x \ne x^*$. Correspondingly, we say that $J(x^*)$ is a (*strict*) *global minimum* of J.

Definition 2.1.9 (Local Minimizer) Let J be defined on $S \subseteq \mathbb{R}^n$. The point $x^* \in S$ is a *local minimizer* of J if there exists an open neighborhood B of x^* such that $J(x^*) \le J(x)$ for all $x \in B \cap S$; it is a *strict local minimizer* if $J(x^*) < J(x)$ for all $x \in B \cap S$, $x \ne x^*$. Correspondingly, we say that $J(x^*)$ is a (*strict*) *local minimum* of J.

Definition 2.1.10 (Stationary Point) We say that x^* is a *stationary point* of J defined on $S \subseteq \mathbb{R}^n$ if $\nabla J(x^*) = 0$.

Definition 2.1.11 (Level Set) Let J be defined on $\mathbb{R}^n$ and $\gamma > 0$. We define the *level set with respect to* γ as

$$\mathcal{L}_\gamma = \left\{x \in \mathbb{R}^n : J(x) \leq \gamma\right\}.$$

Computational procedures that minimize (maximize) J, that is, search for its minima (maxima), are referred to as *optimization methods*. The optimization is often achieved via different *iterative algorithms.* The algorithms begin with a *initial guess* x_0^s and generate a *sequence* $\{x_k^s\}$ leading to a possible solution, that is, a stationary point, a local minimizer or a global minimizer.

Definition 2.1.12 (Algorithm Convergence) Let $S \subseteq \mathbb{R}^n$ and $\{x_k^s\} \subseteq S$ be a sequence generated by an optimization algorithm. If $\lim_{k\to\infty} x_k^s = x^* \in S$ for any $x_0^s \in S$, then we say that the algorithm is *globally convergent*. If such a convergence only exists for some $x_0^s \in S$, then we say the algorithm is *locally convergent*.

Definition 2.1.13 (q-order Convergence) Let $\{x_k^s\}$ be a locally convergent sequence in $S \subseteq \mathbb{R}^n$. We say that $\{x_k^s\}$ is *q-order convergent* if

$$\lim_{k\to\infty} \frac{\|x_{k+1}^s - x^*\|}{\|x_k^s - x^*\|^q} = M$$

exists for some $q, M > 0$. In particular, we say that $\{x_k^s\}$ is *linearly convergent* if $q = 1$; and *superlinearly* or *quadratically convergent* if $1 < q < 2$ or $q = 2$, respectively.

The following standard *stopping criteria* are frequently employed in optimization computations. In the case when $x_k^s \neq 0$ and $J(x_k^s) \neq 0$ for sufficiently large k, computation terminates when

$$\left\|x_{k+1}^s - x_k^s\right\| / \left\|x_k^s\right\| \leq \varepsilon,$$

or

$$\left|J\left(x_k^s\right) - J\left(x_{k+1}^s\right)\right| / \left|J\left(x_k^s\right)\right| \leq \varepsilon.$$

Otherwise, the optimization computation may be terminated when $\|x_{k+1}^s - x_k^s\| \leq \varepsilon$, or $|J(x_k^s) - J(x_{k+1}^s)| \leq \varepsilon$, where $\varepsilon > 0$ is a controlling parameter. More sophisticated stopping criteria may also be considered.

Since most optimization algorithms are iterative, there has been a fundamental trade-off between their efficiency and robustness [14]. In general, algorithms designed to be very efficient on one type of problem tend to be brittle in the sense that

they may not be ideally used for other types of problem. Such a lack of a universally best algorithm is a manifestation of the so-called *No Free Lunch* (*NFL*) *theorems* [16]. The NFL theorems serve as a fundamental barrier to exaggerated claims of the power and efficiency of any specific algorithm in numerical optimizations. A way to cope with negative implications of the barrier is to restrict an algorithm to a particular class of problems and to design the algorithm structures only for the anticipated class. This has become a general principle in optimization.

2.2 Unconstrained Optimization

Let $J : \mathbb{R}^n \to \mathbb{R}$ be a sufficiently smooth objective function. Consider

$$y^* = \min_{x \in \mathbb{R}^n} J(x). \tag{2.2}$$

The above function J is referred as an *objective function*. We do not consider the maximization optimization problem due to the fact that $\max J(x) = -\min(-J(x))$, for $x \in S \subseteq \mathbb{R}^n$. The existence of a global minimizer for (2.2) has been shown in cases where the level sets of J are compact for certain γ [12].

2.2.1 Optimality Conditions

Theorem 2.2.1 (First-order Necessary Conditions, [11]) *If x^* is a local minimizer and J is continuously differentiable in an open neighborhood of x^*, then $\nabla J(x^*) = 0$.*

Theorem 2.2.2 (Second-order Necessary Conditions, [11]) *If x^* is a local minimizer and $\nabla^2 J$ is continuous in an open neighborhood of x^*, then $\nabla J(x^*) = 0$ and $\nabla^2 J(x^*)$ is positive semi-definite.*

Theorem 2.2.3 (Second-order Sufficient Conditions, [11]) *If $\nabla J(x^*) = 0$, $\nabla^2 J$ is continuous in an open neighborhood of x^* and $\nabla^2 J(x^*)$ is positive definite, then x^* is a strict local minimizer of J.*

Theorem 2.2.4 [11] *If J is convex, then any local minimizer x^* is a global minimizer of J. If in addition J is differentiable, then any stationary point x^* is a global minimizer of J.*

The above conditions provide a basis for the developments and analysis of various algorithms. In particular, any well-defined algorithm should verify whether putative solutions satisfy certain optimality conditions, and detect if a minimizer has been satisfactorily approximated. To determine a global minimizer of a given problem is in general difficult, therefore many algorithms used can only guarantee the

convergence to a stationary point. The optimality and algorithms for constrained optimization are based on the results of unconstrained optimization, and additional techniques such as penalty methods and barrier methods are used to address equality and inequality constraints [2, 3, 11].

2.2.2 Line Search Methods

Each iteration in a line search method starts from x_k^s, computes a *search direction* p_k, and then decides *how far* to move along that direction. This iterative process can be illustrated by

$$x_{k+1}^s = x_k^s + \alpha_k p_k,$$

where the positive scalar α_k is called the *step length*. Most line search methods require p_k to be a *descent direction*, that is, $p_k^\top \nabla J(x_k^s) < 0$, to guarantee that the objective function value is reduced along that direction if the step length is sufficiently small. This is understood by using Taylor's theorem, which offers $J(x_k^s + \alpha_k p_k) = J(x_k^s) + \alpha_k p_k^\top \nabla J(x_k^s) + O(\alpha_k^2)$. Since the term $\alpha_k p_k^\top \nabla J(x_k^s)$ dominates $O(\alpha_k^2)$ for small α_k, it follows that $J(x_k^s + \alpha_k p_k) < J(x_k^s)$ for all positive but sufficiently small α_k if $p_k^\top \nabla J(x_k^s) < 0$. The *steepest-descent direction*,

$$p_k = -\nabla J\big(x_k^s\big),$$

is the most obvious choice for search direction. It is chosen among all the directions we could select from x_k^s, and it ensures that J decreases most rapidly.

According to Taylor's theorem, the rate of change in J along p_k at x_k^s is $p_k^\top \nabla J(x_k^s)$. Thus, if the condition $\|p_k\| = 1$ is imposed, the most rapid decrease is given by the solution of the problem

$$\min_{p_k} \big(p_k^\top \nabla J\big(x_k^s\big)\big).$$

Its solution can be found to be $p_k = -\nabla J(x_k^s)/\|\nabla J(x_k^s)\|$, which yields $p_k^\top \nabla J(x_k^s) = -\|\nabla J(x_k^s)\|$.

Other frequently used search directions include the *Newton direction*,

$$p_k = -\big(\nabla^2 J\big(x_k^s\big)\big)^{-1} \nabla J\big(x_k^s\big),$$

which can be adopted in a line search method when the Hessian matrix is positive definite. The basic idea here is to minimize

$$J_k(x) = J\big(x_k^s\big) + \big(x - x_k^s\big)^\top \nabla J\big(x_k^s\big) + \frac{1}{2}\big(x - x_k^s\big)^\top \nabla^2 J\big(x_k^s\big)\big(x - x_k^s\big),$$

the quadratic approximation of J at x_k^s instead of the objective function J itself. Setting the derivative of $J_k(x)$ equal to zero, one obtains

$$\nabla J\big(x_k^s\big) + \nabla^2 J\big(x_k^s\big)\big(x - x_k^s\big) = 0.$$

Therefore $p_k^\top \nabla J(x_k^s) = -p_k^\top \nabla^2 J(x_k^s) p_k \leq -\sigma_k \|p_k\|$ for some $\sigma_k > 0$. Unless the gradient $\nabla J(x_k^s)$ is zero, $p_k^\top \nabla J(x_k^s) < 0$. Therefore, a Newton direction is a descent direction and a normalized unit step length is often utilized.

Needless to mention that calculations of the Hessian matrix may involve large amounts of computation. A *quasi-Newton method* is designed to avoid this disadvantage via features of $J(x_k^s)$ and $\nabla J(x_k^s)$. Their curvature information is used to construct the matrix B_k, an approximation of the Hessian matrix. The standard Quasi-Newton search routine is

$$p_k = -B_k^{-1} \nabla J\left(x_k^s\right).$$

A popular formula for obtaining B_k is the BFGS formula, named after Broyden, Fletcher, Goldfarb, and Shanno:

$$B_k = B_{k-1} - \frac{B_{k-1}^\top s_{k-1}^\top s_{k-1} B_{k-1}}{s_{k-1}^\top B_{k-1} s_{k-1}} + \frac{y_{k-1} y_{k-1}^\top}{y_{k-1}^\top s_{k-1}},$$

where $B_0 = I$, $s_{k-1} = x_k^s - x_{k-1}^s$ and $y_{k-1} = \nabla J(x_k^s) - \nabla J(x_{k-1}^s)$. Factorizations of B_k can be achieved through updating the inverse of B_{k-1} [11].

As yet another alternative, a *conjugate gradient direction* is computed by

$$p_k = -\nabla J\left(x_k^s\right) + \beta_k p_{k-1},$$

where $p_0 = -\nabla J(x_0^s)$, and β_k can either by computed via the Fletcher–Reeves formula,

$$\beta_k = \frac{\nabla J^\top(x_k^s) \nabla J(x_k^s)}{\nabla J^\top(x_{k-1}^s) \nabla J(x_{k-1}^s)},$$

or the Dixon formula,

$$\beta_k = -\frac{\nabla J^\top(x_k^s) \nabla J(x_k^s)}{p_{k-1}^\top \nabla J(x_{k-1}^s)}.$$

The Dixon formula β_k ensures p_k and p_{k+1} are conjugate, a concept originally developed for solutions of linear systems.

2.2.2.1 Step Length

Typical step-length selection algorithms consist of two phases: a *bracketing phase* and a *selection phase*. The former finds an interval $[a, b]$ containing acceptable step lengths, while the latter zooms in the interval to locate the final step length.

The second phase can be implemented by approximating solutions of the following scalar minimization problem:

$$\min_{\alpha>0} \phi(\alpha) = \min_{\alpha>0} J\left(x_k^s + \alpha p_k\right), \quad \alpha > 0, \tag{2.3}$$

which brings the name "line search." An exact solution for the above is called exact line search, which is expensive and frequently not necessary. More practical strategies suggest an *inexact line search* to determine a step size that makes an adequate reduction in J at minimal costs. To achieve this, often used conditions include

$$J\left(x_k^s + \alpha_k p_k\right) \leq J\left(x_k^s\right) + c_1 \alpha_k p_k^\top \nabla J\left(x_k^s\right), \tag{2.4}$$

which prevents steps that are too long via a sufficient decrease criterion, and

$$p_k^\top \nabla J\left(x_k^s + \alpha_k p_k\right) \geq c_2 p_k^\top \nabla J\left(x_k^s\right), \tag{2.5}$$

which prevents steps that are too short via a curvature criterion, for $0 < c_1 < c_2 < 1$. Condition (2.4) is sometimes called the *Armijo condition,* while (2.5) is called the *Wolfe condition*. Moreover, in order to avoid poor choices of descent directions, an *angle condition* [9] can be introduced to enforce a uniformly lower bound on the angle θ_k between p_k and $-\nabla J(x_k^s)$:

$$\cos\theta_k = \frac{-p_k^\top \nabla J(x_k^s)}{\|p_k\| \|\nabla J(x_k^s)\|} \geq c_3 > 0, \tag{2.6}$$

where c_3 is independent of k. The above holds naturally in the method of steepest descent.

2.2.2.2 Convergence and Rate of Convergence

Definition 2.2.5 (First-order Convergence) First-order convergence of an optimization algorithm means that one (or some, or all) of the limit points of the iterate sequence is a stationary point of $J(x)$.

A standard first-order global convergence result for line search methods is

Theorem 2.2.6 [9] *Let $J : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable and bounded from below. Further, let ∇J be Lipschitz continuous with constant $L > 0$, that is,*

$$\left\|\nabla J(y) - \nabla J(x)\right\| \leq L\|y - x\| \quad \textit{for all } x, y \in \mathbb{R}^n.$$

If the sequence $\{x_k^s\}$ satisfies conditions (2.4), (2.5) and (2.6), then

$$\lim_{k\to\infty} \left\|\nabla J\left(x_k^s\right)\right\| = 0.$$

We can relax the assumptions in this theorem, where instead of requiring J to be bounded from below and continuously differentiable on $\mathbb{R}^n$, we only do so within an open set $\mathcal{N}$ containing the level set $\{x | J(x) \leq J(x_0^s)\}$, where x_0^s is the starting point of the iteration. And the gradient ∇J is only required to be Lipschitz continuous on $\mathcal{N}$ [11].

Furthermore, the following theorem shows the linear convergence rate of the steepest-descent algorithm.

Theorem 2.2.7 [7] *Let $J : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable, and assume the Hessian matrix is positive definite. If the sequence $\{x_k^s\}$ is generated by a steepest-descent method with exact line search and it converges to x^*, then*

$$J\left(x_{k+1}^s\right) - J(x^*) \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)\left[J\left(x_k^s\right) - J\left(x^*\right)\right],$$

where $0 < \lambda_1 \leq \cdots \leq \lambda_n$ are the eigenvalues of the Hessian matrix of J.

It has been shown that numerical methods using Newton directions have a fast rate of local convergence, typically quadratic. Their main drawback, however, is the need of the Hessian matrix. There have been numerous recent discussions about the simplification of the underlying computation procedures. In the particularly practical case of the quasi-Newton method, if its search direction approximates the Newton direction accurately enough, then the unit step length can satisfy the Wolfe conditions as the iterates converge to a minimizer. Further, if for the search direction it holds that $\lim_{k\to\infty} \|\nabla J(x_k^s) + \nabla^2 J(x_k^s) p_k\| / \|p_k\| = 0$, then the quasi-Newton method offers a superlinearly convergent iteration. It is also known that for any quadratic objective function, a conjugate gradient method terminates with an optimal solution within n steps.

The following lemma will be used in the robustness analysis for line search methods.

Lemma 2.2.8 (Descent Lemma [2]) *Let $J : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable on $\mathbb{R}^n$. Suppose that ∇J is Lipschitz continuous with constant L. Then for $x, y \in \mathbb{R}^n$,*

$$J(x + y) \leq J(x) + y^\top \nabla J(x) + \frac{L}{2}\|y\|^2.$$

Lemma 2.2.9 *Let $J : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable on $\mathbb{R}^n$. Suppose that ∇J is Lipschitz continuous with constant L. Let α_k, p_k be the step length and descent direction. Then*

$$J\left(x_k^s + \alpha_k p_k\right) - J\left(x_k^s\right) \leq -\frac{c}{2L}\left\|\nabla J\left(x_k^s\right)\right\|^2 \cos^2\theta_k,$$

where $c = 1$ for exact line search, and $c = 2c_1(1 - c_2)$ for inexact line search satisfying conditions (2.4) and (2.5), and θ_k represents the angle between vector p_k and $-\nabla J(x_k^s)$.

Proof First, for exact line search, α_k is the solution of (2.3). From the Descent Lemma 2.2.8, we have $J(x_k^s + \alpha p_k) \leq J(x_k^s) + \alpha p_k^\top \nabla J(x_k^s) + \frac{\alpha^2}{2} L\|p_k\|^2$ valid for all $\alpha > 0$. Letting $\bar{\alpha} = -\frac{p_k^\top \nabla J(x_k^s)}{L\|p_k\|^2} > 0$, it follows that

$$\begin{aligned} J\left(x_k^s + \alpha_k p_k\right) - J\left(x_k^s\right) &\leq J\left(x_k^s + \bar{\alpha} p_k\right) - J\left(x_k^s\right) \quad \text{(exact line search)} \\ &\leq \bar{\alpha} p_k^\top \nabla J\left(x_k^s\right) + \frac{\bar{\alpha}^2}{2} L\|p_k\|^2 \quad \text{(Descent Lemma 2.2.8)} \end{aligned}$$

$$= -\frac{(p_k^\top \nabla J(x_k^s))^2}{L\|p_k\|^2} + \frac{L\|p_k\|^2}{2}\frac{(p_k^\top \nabla J(x_k^s))^2}{(L\|p_k\|^2)^2}$$
$$= -\frac{1}{2L}\left\|\nabla J\left(x_k^s\right)\right\|^2 \cos^2\theta_k.$$

Second, for inexact line search, α_k satisfies conditions (2.4) and (2.5). From the Lipschitz condition, we have $p_k^\top[\nabla J(x_k^s + \alpha_k p_k) - \nabla J(x_k^s)] \le \|p_k\|\|\nabla J(x_k^s + \alpha_k p_k) - \nabla J(x_k^s)\| \le \alpha_k L\|p_k\|^2$. Then from (2.5), we have $-\alpha_k L\|p_k\|^2 \le p_k^\top[\nabla J(x_k^s) - \nabla J(x_k^s + \alpha_k p_k)] \le (1 - c_2)p_k^\top \nabla J(x_k^s)$. That is, $-\alpha_k\|p_k\| \le -\frac{1-c_2}{L}\|\nabla J(x_k^s)\|\cos\theta_k$. Finally, from (2.4),

$$\begin{aligned} J\left(x_k^s + \alpha_k p_k\right) - J\left(x_k^s\right) &\le c_1\alpha_k p_k^\top \nabla J\left(x_k^s\right) \\ &= -c_1\alpha_k\|p_k\|\left\|\nabla J\left(x_k^s\right)\right\|\cos\theta_k \\ &\le -\frac{c}{2L}\left\|\nabla J\left(x_k^s\right)\right\|^2\cos^2\theta_k, \end{aligned}$$

where $c = 2c_1(1 - c_2)$. □

Since $0 < c_1 < c_2 < 1$ is required to ensure the feasibility of inexact line search, we will have $c = 2c_1(1 - c_2) < 1$. This observation is consistent with the upper bound results in the above lemma. That is, we always expect that the exact line search achieves more decrease along the search direction than the inexact line search.

2.2.2.3 Example: Minimization of the Rosenbrock's Function with Line Search Method

The Rosenbrock's function,

$$J(x) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2, \quad x \in \mathbb{R}^2, \tag{2.7}$$

also known as the "banana function," is a benchmark function in unconstrained optimization due to its curvature bends around the origin. The only global minimizer occurs at $x^* = [1, 1]^\top$, where $J(x^*) = 0$. A sequence $\{x_k^s\}$ obtained via the steepest-descent method with inexact line search staring from $x_0^s = [-1.9, 0]^\top$ is shown in Fig. 2.1, where the Armijo condition (2.4) is used and $c_1 = 0.4$. Due to the very slow curvature change of the banana function inside its "valley," the steepest-descent algorithm takes more than one thousand steps to converge.

2.2.3 Trust-Region Methods

At each iteration of a trust-region method, we consider the minimization of a *model function* m_k instead of the objective function J at the current iterate x_k^s. Because the

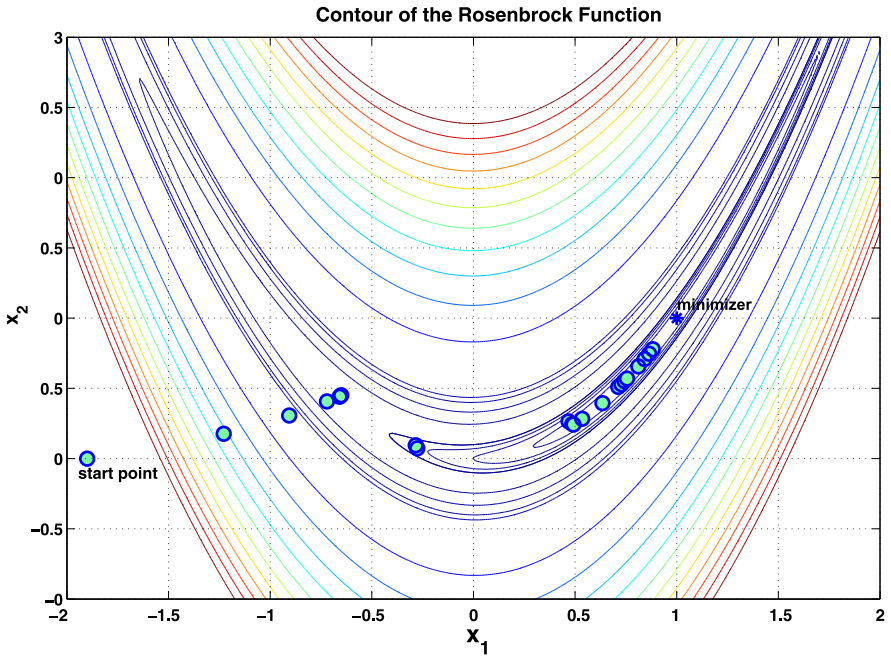

Fig. 2.1 Steepest-descent method on Rosenbrock's function

model function may not be a good approximation of J when x is far away from x_k^s, we have to restrict the search for a minimizer of m_k to a local region around x_k^s. Such a region is called a *trust region*. A trust-region method is defined as

$$\min_{\|p\| \le \Delta_k} m_k\big(x_k^s + p\big). \tag{2.8}$$

Let p_k be the minimizer obtained, and Δ_k the current size of the trust region. The current iterate is then updated to be $x_k^s + p_k$. If the achieved objective function reduction is sufficient compared with the reduction predicted by the model, the trial point is accepted as the new iterate and the trust region is centered at the new point and possibly enlarged. On the other hand, if the achieved reduction is poor compared with the predicted one, the current iterate is typically left unchanged and the trust region is reduced. This process is then repeated until convergence occurs.

Define the ratio

$$\rho_k = \frac{J(x_k^s) - J(x_k^s + p_k)}{m_k(x_k^s) - m_k(x_k^s + p_k)}. \tag{2.9}$$

The following algorithm [11] describes the process.

2.2.3.1 Trust-Region Algorithm

Step 0 Given $\bar{\Delta} > 0$, initialize the trust-region size to $\Delta_0 \in (0, \bar{\Delta})$, and $\eta \in [0, \frac{1}{4})$. Set $k = 0$.
Step 1 Approximately solve the trust-region problem (2.8) to obtain p_k.
Step 2 Evaluate ρ_k from (2.9).
Step 3 If $\rho_k < \frac{1}{4}$, $\Delta_{k+1} = \frac{1}{4}\|p_k\|$; if $\rho > \frac{3}{4}$ and $\|p_k\| = \Delta_k$, $\Delta_{k+1} = \min(2\Delta_k, \bar{\Delta})$; else $\Delta_{k+1} = \Delta_k$.
Step 4 If $\rho_k > \eta$, $x_{k+1}^s = x_k^s + p_k$, else $x_{k+1}^s = x_k^s$. Set $k = k + 1$. Go to Step 1.

Quadratic approximations of J are often used for constructing m_k. In this case, m_k in (2.8) can be formed as

$$m_k(p) = J\left(x_k^s\right) + g_k^\top p + \frac{1}{2} p^\top B_k p. \tag{2.10}$$

The vector g_k is either the gradient $\nabla J(x_k^s)$ or an approximation of it, and the matrix B_k is either the Hessian matrix $\nabla^2 J(x_k^s)$ or an approximation of it. Thus, such construction of m_k still requires gradient information. However, the trust-region framework provides large flexibility in designing derivative-free optimization methods. This compares very favorable with most line search methods which do require gradient measurements of the objective function. Derivative-free trust-region algorithms proposed in [4, 5, 13] use multivariate interpolation to construct the model function m_k, where only an interpolation set Y containing the interpolating nodes and their objective function values are needed. Overall, trust-region methods retain the quadratic convergence rate while being globally convergent. The following is a global convergence result for trust-region methods [11].

Theorem 2.2.10 *Let $J : \mathbb{R}^n \to \mathbb{R}$ be Lipschitz, continuously differentiable and bounded below on the level set $\{x \in \mathbb{R}^n | J(x) \le J(x_0^s)\}$. Further, let $\eta > 0$ in the trust-region algorithm. Suppose that $\|B_k\| \le \beta$ for some constant β, and that all approximate solutions of (2.8) satisfy the inequality*

$$m_k(0) - m_k(p_k) \ge c_t \left\|\nabla J\left(x_k^s\right)\right\| \min\left(\Delta_k, \frac{\|\nabla J(x_k^s)\|}{\|B_k\|}\right)$$

for some constant $c_t \in (0, 1]$, and $\|p_k\| \le \gamma \Delta_k$ for some constant $\gamma \ge 1$. Then

$$\lim_{k\to\infty} \left\|\nabla J\left(x_k^s\right)\right\| = 0.$$

2.2.3.2 Example: Minimization of the Rosenbrock's Function with Trust-Region Method

Again we use the banana function to illustrate the trust-region method. A sequence $\{x_k^s\}$ obtained via the trust-region method starting from $x_0^s = [-1.9, 0]^\top$ is shown in Fig. 2.2, where a quadratic approximation (2.10) is used and the measurements of exact gradient and Hessian are assumed. The trust region $\Delta_0 = 0.5$ and the iterates converge to x^* in 18 steps.

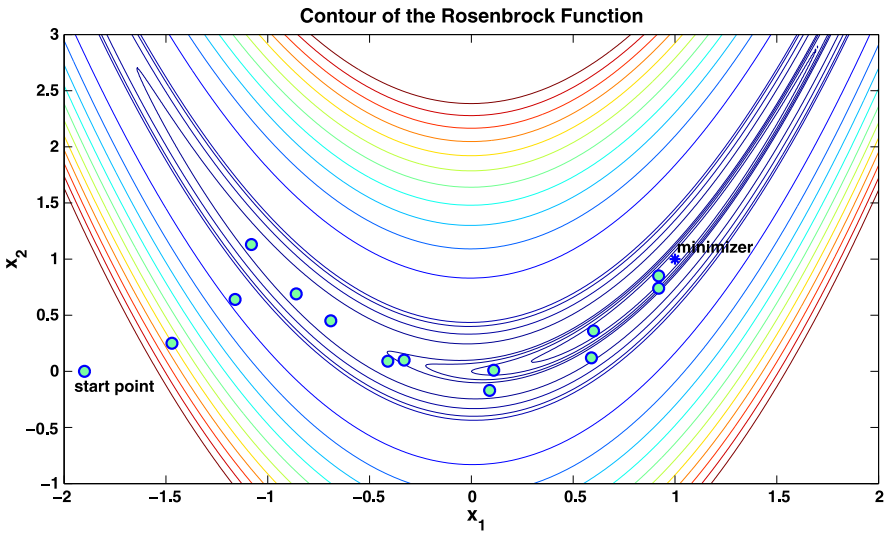

Fig. 2.2 Trust-region method on Rosenbrock's function

2.2.4 Direct Search Methods

Direct search methods are one of the best known methods within the family of derivative-free unconstrained optimization. In the past decades, these methods have seen a revival of interest due to the appearance of rigorous mathematical analysis [1, 15], as well as in parallel and distributed computing. Such features make direct search applicable to the problem of extremum seeking control design. And as direct search does not need derivative information, it can apply to non-smooth objective functions as well. Overall, direct search methods are slower than line search methods, such as steepest-descent method. A systematic review of direct search methods can be found in [9].

The well-known Simplex algorithm of Nelder and Mead [10] is one of the direct search methods. Compass search is one of the earlier version of two dimensional direct search, and it can be summarized as follows: Try steps to the East, West, North, and South. If one of these steps yields a reduction in the function, the improved point becomes the new iterate. If none of these steps yields improvement, try again with steps half as long. By revisiting compass search in a more analytically rigorous manner, it has been named "generating set search" or "pattern search method" [9].

2.2.4.1 Generating Set Search Algorithm

Step 0 Let x_0 be the initial guess. Set $\Delta_{tol} > 0$ as the tolerance used for convergence, and let $\Delta_0 > \Delta_{tol}$ be the initial value of the step-length control parameter.

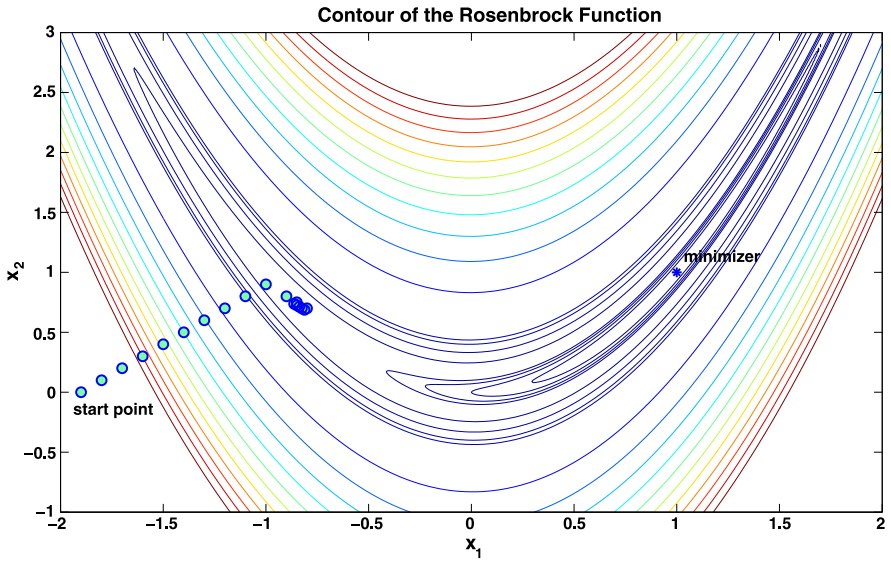

Fig. 2.3 Direct search method on Rosenbrock's function

Step 1 Let D_s be the coordinate direction set (or generator set)

$$D_s = \{e_1, e_2, \ldots, e_n, -e_1, -e_2, \ldots, -e_n\},$$

where e_i is the ith unit coordinate vector in $\mathbb{R}^n$.

Step 2 If there exists $d_k \in D_s$ such that $f(x_k + \Delta_k d_k) < f(x_k)$, then set $x_{k+1} = x_k + \Delta_k d_k$ and $\Delta_{k+1} = \Delta_k$. Set $k = k + 1$ and go to Step 1.

Step 3 Otherwise, if $f(x_k + \Delta_k d) \geq f(x_k)$ for all $d \in D_s$, set $x_{k+1} = x_k$ and $\Delta_{k+1} = \Delta_k/2$. If $\Delta_{k+1} < \Delta_{tol}$, then terminate; otherwise set $k = k + 1$ and go to Step 1.

As depicted for two dimensional compass search, it is easy to see that at each iteration, at least one of the four coordinate directions will be a descent direction. In fact, it is true for any dimension n: given any $x \in \mathbb{R}^n$ for which $\Delta f(x) \neq 0$, at least one of coordinate directions must be a descent direction.

We choose the generator set D_s as $\{e_1, e_2, \ldots, e_n, -e_1, -e_2, \ldots, -e_n\}$ in the above algorithm. In general, it can be any positive spanning set [6]. That is, for n dimensional optimization problem, the minimum number of vectors in the generator set is $n + 1$, which will guarantee a descent direction can be found in the generator set.

Direct search can be thought of as being related to trust-region methods, although in direct search no attempt is done to approximate the objective function nor its gradient, as trust-region methods do. Thus, direct search methods are best suited to problems for which no derivative information is available; in particular, to problems where the objective function is non-smooth.

2.2.4.2 Example: Minimization of the Rosenbrock's Function with Direct Search Method

Again we use the banana function to illustrate the direct search method. A sequence $\{x_k^s\}$ is obtained via the direct search method, starting from $x_0^s = [-1.9, 0]^\top$. The resulting sequence is shown in Fig. 2.3, where the generator set

$$D_s = \{(1, 1), (1, -1), (-1, 1), (-1, 1)\}$$

is used and no derivative information is employed. The initial step length is $\Delta_0 = 0.1$. Only the first 20 steps of the simulation are shown, where it can be seen in Fig. 2.3 that the sequence does converge to the neighborhood of the minimum, but at a very slow rate due to the small gradient change near the minimum, which is located inside an almost flat "valley."

References

1. Audet, C., Dennis, J.E.: Analysis of generalized pattern searches. SIAM J. Optim. **13**(3), 889–903 (2002)
2. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, Belmont (1995)
3. Blowey, J.F., Craig, A.W., Shardlow, T.: Frontiers in Numerical Analysis. Springer, Berlin (2002)
4. Conn, A.R., Scheinberg, K., Toint, P.L.: On the convergence of derivative-free methods for unconstrained optimization. Approximation Theory and Optimization: Tributes to M.J.D. Powell, pp. 83–108 (1997)
5. Conn, A.R., Scheinberg, K., Toint, P.L.: Recent progress in unconstrained nonlinear optimization without derivatives. Math. Program. **79**, 397–414 (1997)
6. Davis, C.: Theory of positive linear dependence. Am. J. Math. **76**(4), 733–746 (1954)
7. Fletcher, R.: Practical Methods of Optimization. Wiley, New York (1987)
8. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press, Cambridge (1985)
9. Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: new perspectives on some classical and modern methods. SIAM Rev. **45**(3), 385–482 (2003)
10. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**(4), 308–313 (1965)
11. Nocedal, J., Wright, S.: Numerical Optimization. Springer, Berlin (1999)
12. Pardalos, P.M., Resende, M.G.C.: Handbook of Applied Optimization. Oxford University Press, New York (2002)
13. Powell, M.J.D.: UOBYQA: Unconstrained optimization by quadratic approximation. Technical Report NA2000/14, DAMTP. University of Cambridge (2000)
14. Spall, J.C.: Introduction to Stochastic Search and Optimization. Wiley-Interscience, New Jersey (2003)
15. Torczon, V.: On the convergence of pattern search algorithms. SIAM J. Optim. **7**(1), 1–25 (1997)
16. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**, 67–82 (1997)

Springer

http://www.springer.com/978-1-4471-2223-4

Extremum-Seeking Control and Applications
A Numerical Optimization-Based Approach

Zhang, C.; Ordóñez, R.

2012, XVIII, 201 p., Hardcover
ISBN: 978-1-4471-2223-4