Numerical Methods for Nonlinear Optimization and Optimal Control

Dominique Orban
dominique.orban@polymtl.ca

Département de Mathématiques et Génie Industriel

Revision 35 of November 5, 2008

"To know and not to do is not to know."
Ancient Chinese Proverb.

Abstract

These are preliminary notes for the three courses available from the Mathematics and Industrial Engineering Department at École Polytechnique de Montréal, *Mathematical Programming I* (MTH6403), *Numerical Methods for Optimization and Control* (MTH6408) and *Mathematical Programming II* (MTH6413).

The purposes of the three classes differ but we hope that this document can serve both as class notes and also as a reference for further readings and more advanced subjects. By definition, MTH6408 is more applied and less technical than the other two classes. It should not present any difficulty to work through these notes, skip some technical discussions and still have a useful reference manual. A section on control problems will be introduced for MTH6408 and students in the other two classes may find it interesting to browse.

These notes expand as the classes progress and will hopefully be enriched by input from the students.

Contents

Introduction

Chapter 1

Preliminaries

In this chapter, we start with a quick look at a few important classes of optimization problems, show how they relate to each other and how convexity is an important additional property that each may possess.

A simple, but sufficient for most practical purposes, tree showing the different classes of optimization problems is shown in Fig. I.1.

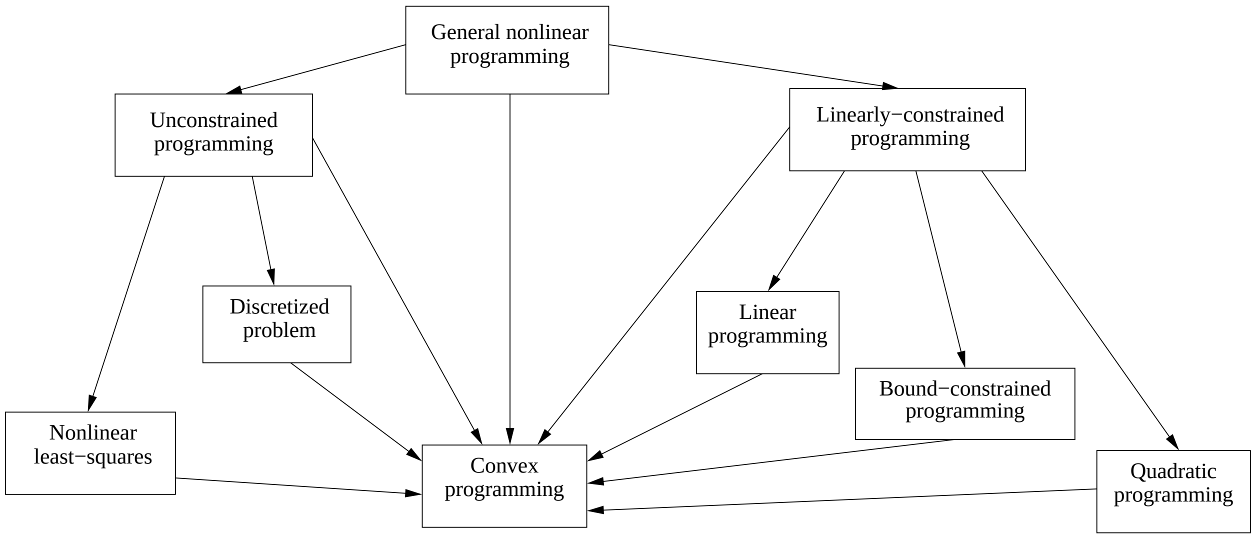

General nonlinear programming
Unconstrained programming
Linearly−constrained programming
Discretized problem
Linear programming
Bound−constrained programming
Nonlinear least−squares
Convex programming
Quadratic programming

Figure I.1: A practical optimization tree.

In the remainder of this section, we will address the following questions.

(a) Why classify optimization problems?

(b) Convexity seems to be central. Why is it important?

(c) If convexity is important, is it possible to detect it? How?

(d) Is it not possible to break optimization problems into many more classes? Why not do it?

(e) What do we do once a problem has been classified?

The tree of Fig. I.1 might be called *structural* because its nodes refer to the particular structure of an optimization problem. Detecting and knowing this structure is key to successfully solving the problem. A feature of this tree is that the farthest a node is from the root, the more specialized the structure. Note that convexity, rather than a structural property of its own, should instead be seen as an additional characteristic that a problem if any class may have. This is the reason why all nodes have the convexity node as successor.

We start by giving in more detail the structure of the problems represented by each node of the tree. Unless stated otherwise, all functions are supposed to be twice-continuously differentiable. Note that below, inequalities between vectors are to be understood componentwise.

General Nonlinear Program. This is the most general class. Problems in this class either do not fit in any more specialized class, or have an unknown structure. They have the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & c_{\mathcal{E}}(x) = 0 \\ & c_{\mathcal{I}}(x) \geq 0, \end{aligned} \tag{I.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $\mathcal{E} = \{1, \ldots, n_{\mathcal{E}}\}$, $\mathcal{I} = \{1, \ldots, n_{\mathcal{I}}\}$, $c_{\mathcal{E}} : \mathbb{R}^{n_{\mathcal{E}}} \to \mathbb{R}$ and $c_{\mathcal{I}} : \mathbb{R}^{n_{\mathcal{I}}} \to \mathbb{R}$. The functions f, $c_{\mathcal{E}}$ and $c_{\mathcal{I}}$ are either nonlinear or nothing is known about their structure. This problem is nonconvex in general.

Unconstrained Program. This case corresponds to $n_{\mathcal{E}} = n_{\mathcal{I}} = 0$ or equivalently $\mathcal{E} = \mathcal{I} = \emptyset$. The problem formulation is

$$\underset{x}{\text{minimize}} \ f(x) \tag{I.2}$$

over $\mathbb{R}^n$. For this problem to make sense, one usually assumes that f is bounded below on $\mathbb{R}^n$. Given the absence of constraints, algorithms for (I.2) usually have a rather simple form and provide good insights when designing algorithms for the constrained case. This is a convex problem if and only if f is convex over $\mathbb{R}^n$.

Nonlinear Least-Squares Problem. This is a special case of (I.2) in which the objective function takes the very particular form

$$f(x) = \sum_{i=1}^{p} f_i(x)^2, \tag{I.3}$$

where the functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, p$, are twice-continuously differentiable. This type of problem typically appears when minimizing an error. A classic example is linear regression. If all the f_i are nonnegative and convex, f is convex. The converse is not true.

Discretized Problem. Problems in this class result from the discretization of infinite-dimensional problems, usually involving the integral of a functional, a system of ordinary or partial differential equations. When designing algorithms for such problems, the relation between the finite and the infinite-dimensional problems is paramount. This importance calls for very specialized algorithms, which often involve discretizing the problem several times. We will return to this class later.

Linearly-Constrained Program. In this class, the constraint functions are linear but the objective function may be nonlinear. The general form of the problem is

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & A_{\mathcal{E}}x - b_{\mathcal{E}} = 0 \\ & A_{\mathcal{I}}x - b_{\mathcal{I}} \geq 0, \end{aligned} \tag{I.4}$$

where $A_{\mathcal{E}} \in \mathbb{R}^{n_{\mathcal{E}} \times n}$, $A_{\mathcal{I}} \in \mathbb{R}^{n_{\mathcal{I}} \times n}$, $b_{\mathcal{E}} \in \mathbb{R}^{n_{\mathcal{E}}}$ and $b_{\mathcal{I}} \in \mathbb{R}^{n_{\mathcal{I}}}$. Usually, regularity assumptions are made to ensure consistency; the matrices $A_{\mathcal{E}}$ and $A_{\mathcal{I}}$ are required to have full row rank. Given the form of the constraints, specialized algorithms may be derived. Note that the feasible set is always polyhedral, hence convex. This problem is therefore convex if and only if f is convex over the feasible set.

Linear Program. This class is so special that entire books are devoted to its exploration and over the years, algorithms have become extremely efficient and able to solve very large-scale problems quickly.

Algorithms for linear programs have also formed the basis of algorithms for convex quadratic programs and for semidefinite programs. The problem has the form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & A_{\mathcal{E}} x - b_{\mathcal{E}} = 0 \\ & A_{\mathcal{I}} x - b_{\mathcal{I}} \geq 0, \end{array} \tag{I.5}$$

where $c \in \mathbb{R}^n$ is often referred to as the *cost vector*. The value of the objective function $f(x) = c^T x$ is sometimes called the *cost*, a terminology that somewhat carried over to other problem classes. Problems of the form (I.5) are always convex.

Bound-Constrained Program. This class features the simplest constraints. Bound constraints are sometimes referred to as *box constraints* because of their geometry. The problem has the form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & x_{\text{L}} \leq x \leq x_{\text{U}}, \end{array} \tag{I.6}$$

where $x_{\text{L}}, x_{\text{U}} \in \mathbb{R}^n$. Given the relative simplicity of these constraints, very efficient specialized algorithms have been developed for (I.6). Note that the feasible set is always convex. This program is therefore convex if and only if f is convex over the bounds.

Quadratic Program. This is probably the class that received the most attention after the linear programs class. It benefits from particularly strong properties but even though quadratic programs are only the next more complicated problems after (I.5), a number of fairly tricky issues come into play. The problem has the general form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2} x^T H x - b^T x \\ \text{subject to} & A_{\mathcal{E}} x - b_{\mathcal{E}} = 0 \\ & A_{\mathcal{I}} x - b_{\mathcal{I}} \geq 0, \end{array} \tag{I.7}$$

where $b \in \mathbb{R}^n$ and $H \in \mathbb{R}^{n \times n}$ is a *symmetric* matrix. The problem is convex if and only if H is positive semidefinite and strictly convex if and only if H is positive definite. Under convexity, it is striking to see that some algorithms for linear programming can be readily adapted.

In the next few sections, we give some elements of answers to the questions asked above.

Why Classify Optimization Problems?

Computer programming textbooks and computer programmers themselves repeatedly advise that no programming language is the best tool for all programming tasks. The same goes in optimization. No algorithm is the best solution to all problems, or even to all problems of a given category. Optimization problems come in much more variety than suggested by our practical tree of Fig. I.1—they include nondifferentiable problems, noisy problems, problems with integral data, problems with categorical variables, network problems, infinite-dimensional problems, stochastic problems and more. It is almost unconceivable to write an algorithm that will even remotely treat *any* type of problem. Of course, given the nature of our tree, it can be argued with reason that if algorithm $\mathcal{A}$ is able to solve the most general problems, it can also solve the more specialized ones. However, an algorithm specifically targeted at convex quadratic programs will be able to employ some linear algebra methods which would not make sense in a general setting, for instance a symmetric positive definite solver. It seems reasonable to say that in general, a specialized solver should perform better on a specialized problem than a general-purpose solver.

Classifying problems helps make the distinction between those features that problems have which call for different algorithmic techniques. It helps make an educated choice when the time comes to pick a solver or building blocks with which a solver will be written.

Why is Convexity Important?

Convexity is important for several reasons. A theoretical reason comes from the result stating that if our problem is convex, any local minimizer is also global. This means that we have been able to find the overall best solution. Generally speaking, in large-scale nonlinear optimization, finding a global minimizer is hopeless. Convex problems form a class apart in which that may be achieved. A second reason is that if we know a priori that the problem is convex, or if we can detect it, then it narrows down the spectrum of solvers from which to choose. A third reason is that convexity allows us to employ specific linear algebra tools in the course of the iterations. If our problem is convex, we will probably only need to solve symmetric positive semidefinite systems, and we can use for that, for instance, a Cholesky factorization, which will be more efficient than an LU decomposition in general.

Convexity Detection

Convexity, or equivalently, concavity, is a complex property to asses given a nonlinear function. Depending on how the functions are coded internally, it will be more or less easy to assess convexity. For instance, in modeling languages such as AMPL [FGK02], nonlinear functions are represented in reverse Polish notation and subsequently as a directed acyclic graph (a "DAG") which is a convenient data structure to both evaluate and differentiate the function. By recursively walking this graph, it is possible to infer properties such as monotonicity, convexity, quadraticity, polynomiality, etc. We will not go into the details here. More information is given in [FO07, FMN+04].

Enhancements to the Tree

It is of course possible to greatly expand the tree of Fig. I.1 to include all, or most, classes of optimization problems. We will refrain in this document for two reasons. The first is that we are only examining continuous problems with no stochastic data, which already eliminates a number of classes. The second reason is that although we could furthe specialize some classes—for instance problems with a linear objective and bound constraints, problems with a polynomial objective and quadratic convex constraints, etc.—there is no advantage to do so if no algorithms exist for those new classes. There exist algorithms for problems with polynomial objective and constraints, for problems with quadratic constraints, and so forth, but for the sake of generality, we will not examine those fairly specialized classes. A good resource for solvers tailored to different problems is the NEOS server at `http://neos.mcs.anl.gov`.

Choosing a Solver

A reason why we want to classify problems is to help choose an appropriate solver. There exist several guides to assist in this choice. The Optimization Software Guide [MW93] gives an overview of the software available up until 1993 and the classes of problems for which each solver is designed. To update the book, the online NEOS guide was created, at the address `www.mcs.anl.gov/otc/Guide`.

As an addition to solver choice, the NEOS server also lets you try and benchmark solvers online on a problem of your choice.

As a different means of automatically suggesting appropriate solvers for a given problem structure, the author designed a relational database pairing solvers with the structural features of the problems which they are designed to solve. The DrAmpl tool [FO07] automatically examines models written in the AMPL [FGK02] modeling language to gather these structural features. It then issues a query towards the database to obtain a list of recommended solvers.

Chapter 2 Implementation of Algorithms in the Present Notes

This chapter describes the language and framework in which the algorithms presented in these notes are implemented for the purpose of an illustration.

To better understand the concepts introduced in the next chapters, and to gain insight into the various algorithms for nonlinear optimization, it is important to realize that they are tangible and seeing them in action is probably the best way to achieve this.

We believe it is important to implement an algorithm in order to fully grasp its mechanisms. There is nothing like comparing two algorithms in action to point out the differences between them, their forces and their weaknesses.

There are a myriad of programming languages, most of which are reasonably well suited to the programming of optimization algorithms, and a choice must be made.

Low-level languages, often compiled, such as C, C++, Fortran 77 or Fortran 90/95 are natural candidates and are indeed very popular in numerical computing circles, rightfully so. They present however the difficulty that they have a steep learning curve, and the programmer must mind issues unrelated to the optimization algorithm, such as memory allocation, the proper declaration of variables, and so forth.

On the other hand, higher-level languages, often interpreted, offer the advantage of being easy to learn and to put to work but also may be sluggish and have difficulties to treat larger problems, for lack of available memory or for having too much overhead.

Our opinion is that a language which would fall in between the two categories would be optimal for learning, experimenting with ideas, prototyping new algorithms, and also solving reasonably large problems.

It appears unfortunately that there is no such language. However, it turns out that it is rather easy to use a combination of low and high-level languages to take advantage of the strenghts of both while at the same time avoiding their downsides. This can also be done without sacrificing execution speed or memory. The combination we have retained in this document is composed of

(a) C and Fortran as low-level languages to implement the numerically-intensive tasks, such as matrix-vector products, linear system solving, subproblem solving, etc.,

(b) Python as high-level language in which algorithms can be easily programmed without being concerned about the details not pertaining to the optimization.

This system is organized in three layers. In the first layer, computation-intensive numerical tasks are implemented in C or Fortran. Often, these routines are reference implementations that constitute all the necessary building blocks that are used to make an optimization algorithm. This includes solving symmetric positive-definite systems, symmetric indefinite systems, obtaining a Cholesky factorization, performing a Krylov-type iteration, and so forth.

The second layer is an interface between the low and high-level languages. It is constituted of code written partly in C and partly in Python. This part of the code can be considered *wrappers* around the low-level subroutines of the first layer but are at the same time providing convenient interfaces that are used by the third layer.

The third layer is the one most visible to the user, and the only one most users will need to use. This layer is constituted of programs written purely in Python—and hence short, easy to read and easy to modify—in which simple transparent function calls refer to the intensive task-performing subroutines of the first layer through the interface of the second layer.

The advantage of this design choice is that users can concentrate on the logic of the optimization algorithm and need not mind computer-specific details. Moreover, since Python is available on so many platforms, all of which have also fully supported C and Fortran for many years, users need not change anything when moving from a platform or an operating system to another. Since all complex tasks are implemented in low level languages, they are executed efficiently, memory efficient and fast. Finally, Python is an open source programming language and the GNU implementations of C and Fortran are also available, and open source, for many different platforms.

The resulting system is named `nlpy`, for *nonlinear programming in Python.* It can be obtained, along with more information on its design, examples, updated, from its main website `nlpy.sf.net`.

It is not all to be able to program optimization algorithms, we also need a convenient way to represent optimization *problems.* This can be done in essentially two ways:

(a) Represent the problem by a set of subroutines or functions, to implement the objective function, the constraints and their first and possibly second derivatives if needed by the algorithm. This may seem simple for small-scale problems but quicky becomes burdensome and error-prone as the number of variables or constraints increase.

(b) Represent the problem in a *modeling language.*

Both are possible in `nlpy` although the second one is certainly more convenient.

Chapter 3

Unconstrained Optimization

Let $f : \mathbb{R}^n \to \mathbb{R}$ and consider the unconstrained problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \tag{III.1}$$

In this section, we take a close look at local solutions of (III.1) and identify a number of necessary and sufficient conditions for an instance of x to be a solution.

3.1 Necessary and Sufficient Optimality Conditions

We distinguish several types of solutions for (III.1).

Definition 3.1.1. Consider the unconstrained problem (III.1) and let $x^* \in \mathbb{R}^n$. We say that x^* is

(a) a *global unconstrained minimizer* of f if and only if for all $x \in \mathbb{R}^n$, $f(x) \geq f(x^*)$,

(b) a *local unconstrained minimizer* of f if and only if there exists $\epsilon > 0$ such that for all $x \in B(x^*, \epsilon)$, $f(x) \geq f(x^*)$,

(c) a *strict local unconstrained minimizer* of f if and only if there exists $\epsilon > 0$ such that for all $x \in B(x^*, \epsilon)$, $x \neq x^*$, $f(x) > f(x^*)$,

(d) an *isolated local unconstrained minimizer* of f if and only if there exists $\epsilon > 0$ such that x^* is the only local unconstrained minimizer of f in $B(x^*, \epsilon)$.

Examples of a local minimizers that are strict but not isolated appear in Fig. III.1. For both problems, $x = 0$ is a local minimizer but regardless of the value of $\epsilon > 0$, there always exists another minimizer in $]-\epsilon, \epsilon[$.

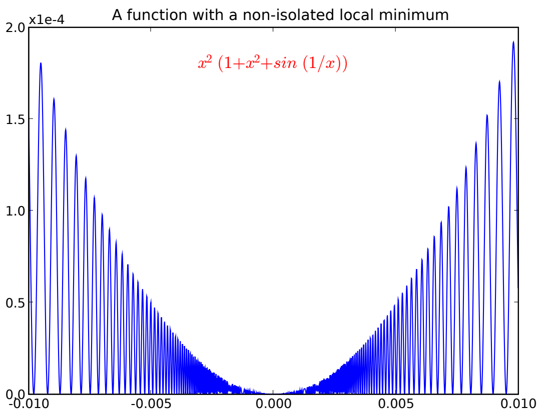
Note that an isolated minimizer is necessarily strict but that the converse is not true. In Fig. III.1, $x = 0$ is in fact a strict local minimizer. An even stronger example is shown in the rightmost picture of Fig. III.1 where 0 is a strict *global* minimizer that is not isolated.

Typical issues in unconstrained programming are

(a) does there exist a solution?

(b) if there is one, is it unique or are there several? Are they local or global solutions?

Except in some highly structured cases and typically small-scale, although not always, it is normally hopeless to find a global minimizer. We will thus be satisfied with local minimizers. Often, it is even difficult to guarantee that a point is indeed a minimizer and not only something that resembles a minimizer. To be able to identify candidate minimizers we need a set of conditions that all minimizers must satisfy. These are the necessary optimality conditions.



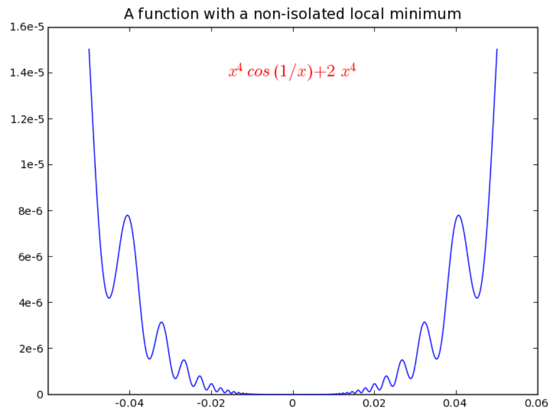


Figure III.1: The objective function on the left is $f(x) = x^2(1 + x^2 + \sin(1/x))$ and on the right, $f(x) = x^4 \cos(1/x)+2x^4$. Each has a strict local (and global) minimizer at $x^* = 0$, but this minimizer is not isolated.

Theorem 3.1.1 (First-order necessary conditions)**.** *Let $x^* \in \mathbb{R}^n$ be a local minimizer of (III.1) and let f be $\mathcal{C}^1$ over an open ball centered at x^*. Then $\nabla f(x^*) = 0$.*

Proof. Since x^* is a local minimizer, there is $\epsilon > 0$ such that for all $x \in B(x^*, \epsilon)$, $f(x) \geq f(x^*)$. Let $u \in \mathbb{R}^n$ be an arbitrary unit direction, i.e., $\|u\| = 1$ and let $d = \alpha u$ for fixed $\alpha > 0$. For sufficiently small values of $t > 0$ we have $x^* + td \in B(x^*, \epsilon)$. Therefore, by Taylor's theorem and because x^* is a local minimizer,

$$0 \leq f(x^* + td) - f(x^*) = t\nabla f(x^*)^T d + o(t\|d\|).$$

Dividing by $t\|d\|$, there remains

$$\nabla f(x^*)^T u + \frac{o(t\|d\|)}{t\|d\|} \geq 0.$$

Now we let $\alpha \to 0$ so that $d \to 0$ along the direction u and we obtain $\nabla f(x^*)^T u \geq 0$. We can repeat this reasoning with u replaced by $-u$ and similarly, we obtain $\nabla f(x^*)^T(-u) \geq 0$, i.e., $\nabla f(x^*)^T u = 0$. Since u was arbitrary, we have $\nabla f(x^*)^T u = 0$ for all $u \in \mathbb{R}^n$. Therefore, $\nabla f(x^*) = 0$. ☐

Typically, the negation of the necessary conditions given by Theorem 3.1.1 is used. If $\nabla f(x) \neq 0$ then x cannot be a local minimizer of f.

Definition 3.1.2. A point $x \in \mathbb{R}^n$ at which $\nabla f(x) = 0$ is said to be *first-order critical* or *stationary.*

When $n = 1$, these conditions reduce to the well-known $f'(x) = 0$. However, these are not sufficient conditions; it suffices to consider the univariate function $f(x) = -x^2$. The first derivative of f vanishes at $x = 0$ but this point is in fact a local maximum. Local maxima are ruled out in the following extension to Theorem 3.1.1.

Theorem 3.1.2 (Second-order necessary conditions)**.** *Let $x^* \in \mathbb{R}^n$ be a local minimizer of (III.1) and let f be $\mathcal{C}^2$ over an open ball centered at x^*. Then $\nabla^2 f(x^*)$ is positive semi-definite.*

Proof. Just as in the proof of Theorem 3.1.1, let $u \in \mathbb{R}^n$ be an arbitrary unit direction and $d = \alpha u$ for fixed $\alpha > 0$. For all sufficiently small values of $t \geq 0$,

$$0 \leq f(x^* + td) - f(x^*) = \frac{1}{2}t^2 d^T \nabla^2 f(x^*) d + o(\|td\|^2),$$

where we used the fact that $\nabla f(x^*) = 0$. Dividing by $\|td\|^2$,

$$\frac{1}{2}u^T \nabla^2 f(x^*)u + \frac{o(\|td\|^2)}{\|td\|^2} \geq 0.$$

We now let $\alpha \downarrow 0$. There remains $u^T \nabla^2 f(x^*)u \geq 0$. Since u was arbitrary, $\nabla^2 f(x^*)$ must be positive semi-definite. $\square$

When $n = 1$ the second-order necessary conditions become the familiar $f''(x^*) \geq 0$. They however remain necessary conditions only since for $f(x) = x^3$ we have $f'(0) = f''(0) = 0$ but $x = 0$ is still not a local minimizer. Only if $f''(x^*) > 0$ will we be sure that x^* is a local minimizer. This is stated in the next theorem.

Theorem 3.1.3 (Second-order sufficient conditions)**.** *Let $x^* \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ be $\mathcal{C}^2$ over an open ball centered at x^*. If $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite, x^* is an isolated local minimizer of f.*

Proof. By assumption, there is $\delta > 0$ such that for $y \in B(x^*, \delta)$, we may again apply Taylor's theorem to obtain

$$\frac{f(y) - f(x^*)}{\|y - x^*\|^2} = \frac{1}{2}\frac{(y - x^*)^T}{\|y - x^*\|}\nabla^2 f(x^*)\frac{(y - x^*)}{\|y - x^*\|} + \frac{o(\|y - x^*\|^2)}{\|y - x^*\|^2}. \tag{III.2}$$

The first term of the right-hand side being always positive, there is an $\epsilon > 0$, $\epsilon \leq \delta$, such that if $y \in B(x^*, \epsilon)$, the right-hand side of (III.2) is nonnegative. For all such y, we thus also have $f(y) - f(x^*) \geq 0$ and therefore x^* must be a local minimizer.

To see that x^* is isolated, we use the fact that since $f \in \mathcal{C}^2$, then $\nabla f \in \mathcal{C}^1$ and

$$\begin{aligned} \nabla f(x^* + d) &= \nabla f(x^*) + \nabla^2 f(x^*)d + \epsilon(\|d\|) \\ &= \nabla^2 f(x^*)d + \epsilon(\|d\|), \end{aligned}$$

where the function $\epsilon(\cdot)$ is such that $\lim_{d\to 0} \epsilon(\|d\|)/\|d\| = 0$. By taking norms and using the inverse triangle inequality and the fact that $\nabla^2 f(x^*)$ is positive definite,

$$\begin{aligned} \|\nabla f(x^* + d)\| &\geq \|\nabla^2 f(x^*)d\| - |\epsilon(\|d\|)| \\ &\geq \sigma_{\min}\|d\| - |\epsilon(\|d\|)|, \end{aligned} \tag{III.3}$$

where $\sigma_{\min} > 0$ is the smallest singular value of $\nabla^2 f(x^*)$. Dividing by $\|d\|$ we see that there is $\gamma > 0$ such that the right-hand side of (III.3) is positive whenever $\|d\| \leq \gamma$. Since d was arbitrary, the gradient of f cannot vanish in the open ball $B(x^*, \gamma)$ and x^* is therefore isolated. $\square$

This result eliminates the borderline case of $f(x) = x^3$ and $x = 0$. Note that the converse of Theorem 3.1.3 does not hold. Even if x^* is an isolated local minimizer, $\nabla^2 f(x^*)$ may be singular. Consider for instance $n = 1$ and $f(x) = x^4$ for which $x^* = 0$ is an isolated minimizer but $f''(x^*) = 0$.

There are three corresponding theorems for the case of a local maximizer. Their proofs are identical to the proofs above.

Theorem 3.1.4. *Let $x^* \in \mathbb{R}^n$ be a local maximizer of (III.1) and let f be $\mathcal{C}^1$ over an open ball centered at x^*. Then $\nabla f(x^*) = 0$.*

Theorem 3.1.5. *Let $x^* \in \mathbb{R}^n$ be a local maximizer of (III.1) and let f be $\mathcal{C}^2$ over an open ball centered at x^*. Then $\nabla^2 f(x^*)$ is negative semi-definite.*

Theorem 3.1.6. *Let $x^* \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ be $\mathcal{C}^2$ over an open ball centered at x^*. If $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is negative definite, x^* is an isolated local maximizer of f.*

Definition 3.1.3. A point $x^* \in \mathbb{R}^n$ such that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is indefinite is called a *saddle point* of f.

When $n = 1$, a saddle point is also sometimes called an inflexion point. For example, if $f(x) = x^3$, $x = 0$ is an inflexion point. If $f(x, y) = x^2 - y^2$, $(x, y) = (0, 0)$ is a saddle point. The terminology *saddle point* is justified by the last example, shown in Fig. III.2. We illustrate the above concepts on some examples.

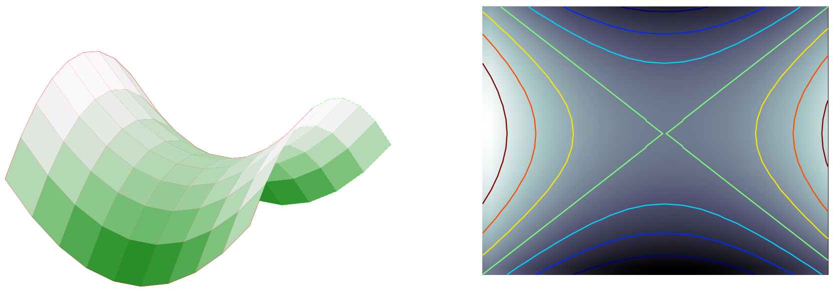


Figure III.2: The function $f(x, y) = x^2 - y^2$ has a saddle point at $(0, 0)$. On the left, the graph of the function is represented and on the right, its level curves, or *contours*.

Example 3.1.1. Let $f : \mathbb{R}^n \to \mathbb{R}$, $f(x) = \|x\|_2^2 = x_1^2 + x_2^2 + \dots x_n^2$. The only stationary point of f is $x = 0$, which is a local minimizer since $\nabla^2 f(0) = 2I$. $\square$

Example 3.1.2. Let $f : \mathbb{R}^2 \to \mathbb{R}$, $f(x, y) = 2x^3 - 3x^2 - 6xy(x - y - 1)$. We compute

$$\nabla f(x) = 6 \begin{bmatrix} x^2 - 2xy - x + y^2 + y \\ 2xy - x^2 + x \end{bmatrix}.$$

The only candidate minimizers are therefore $(0, 0)$, $(1, 0)$, $(-1, -1)$ and $(0, -1)$. By evaluating the Hessian matrix

$$\nabla^2 f(x) = 6 \begin{bmatrix} 2x - 2y - 1 & -2x + 2y + 1 \\ -2x + 2y + 1 & 2x \end{bmatrix},$$

we find that $(0, 0)$ is a saddle point, $(1, 0)$ is a local minimizer, $(-1, -1)$ is a local maximizer and $(0, -1)$ is a saddle point. $\square$

3.2 Quadratic Objectives

It quickly becomes apparent that problems with a quadratic objective play a central role in nonlinear optimization. According to Taylor's theorem, all $\mathcal{C}^2$ functions are locally virtually indistinguishable from their second-order expansion. It is also enlightening to examine the necessary and sufficient conditions of §3.1 in the case of a quadratic objective.

Let $f(x) = g^T x + \frac{1}{2} x^T H x$ where $g \in \mathbb{R}^n$ and $H \in \mathbb{R}^{n \times n}$ is symmetric. The first-order necessary conditions read

$$H x^* = -g, \tag{III.4}$$

which is a system of n linear equations in n unknowns.

The second-order necessary conditions are that H be positive semi-definite. Hence it is only for convex quadratic objectives that we can hope to find a local unconstrained minimizer. If H is indefinite, there can exist no local minimizer.

Example 3.2.1. The quadratic function $f(x, y) = x^2 - y^2$ possesses no local unconstrained minimizer. In fact, f is unbounded below. $\square$

The second-order sufficient conditions are that H be positive definite. In this case, H is nonsingular and the linear system (III.4) has a single solution $x^* = -H^{-1}g$, which is the unique global minimizer of f since then f is strictly convex.

Example 3.2.2. The Euclidian norm function of Example 3.1.1 is quadratic and strictly convex and hence, possesses a unique global minimizer. □

When H is positive semi-definite but not positive definite, it is singular and in this case, it may be that there exists a local minimizer or that there does not exist any. If there is one however, it cannot be isolated since there is a direction $d \neq 0$ such that $Hd = 0$. Thus all points of the form $x^* + d$ are also local minimizers and x^* cannot be isolated.

Example 3.2.3. For the quadratic objective $f(x, y) = x^2$, all points of the form $(0, y)$ with $y \in \mathbb{R}^n$ are stationary. Since $\nabla^2 f(x, y)$ is always positive semi-definite, they are all local minimizers but never isolated. □

3.3 Convexity in Unconstrained Programming

This section anticipates on more general convex programs to illustrate the power of convexity at this early stage. We first recall what we mean by saying that a function or a set is convex.

Definition 3.3.1. A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if for all $x, y \in \mathbb{R}^n$ and all $\lambda \in [0, 1]$, $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$. The function f is *concave* if $-f$ is convex. If f is both convex and concave, it is *affine*.

A convex function is one for which the chord between $f(x)$ and $f(y)$ lies above the image of the segment between x and y.

Definition 3.3.2. A subset $C \subseteq \mathbb{R}^n$ is said to be *convex* if for all $x, y \in C$ and all $\lambda \in [0, 1]$, the convex combination $\lambda x + (1-\lambda)y \in C$.

A convex set is one which contains entirely the line segment joining any two of its elements.

Convex functions and convex sets are tightly linked. An important set associated to a convex function $f : \mathbb{R}^n \to \mathbb{R}$ is its *epigraph*, the subset of $\mathbb{R}^n \times \mathbb{R}$ defined by

$$\text{epi}(f) = \{(x, \mu) \in \mathbb{R}^n \times \mathbb{R} \mid \mu \geq f(x)\}. \tag{III.5}$$

Using the epigraph of a function, it is possible to prove that f is a convex function if and only if its epigraph is a convex subset of $\mathbb{R}^n \times \mathbb{R}$.

Strictly speaking, a convex function need not be defined over the whole of $\mathbb{R}^n$. If instead the domain of f is $D \subseteq \mathbb{R}^n$, we require that D be a convex set and that Definition 3.3.1 be satisfied for all $x, y \in D$.

If f is also differentiable, the following result states that a function is convex if and only if its graph always lies above the tangent to the graph at any point.

Theorem 3.3.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable. Then f is convex if and only if for all $x, y \in \mathbb{R}^n$,*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x).$$

Example 3.3.1. Many common functions are convex. Here are some examples.

(a) e^{ax} is convex on $\mathbb{R}$ for any $a \in \mathbb{R}$,

(b) x^a is convex on $\mathbb{R}_0^+$ for $a \leq 0$ or $a \geq 1$ and concave on $\mathbb{R}_0^+$ for $0 < a < 1$,

(c) $|x|^p$ is convex on $\mathbb{R}$ for $p \geq 1$,

(d) $-\log(x)$ is convex on $\mathbb{R}_0^+$,

(e) $-\log(\det X)$ is convex on the cone of symmetric positive-definite matrices $\mathcal{S}_n^{++} = \{X \in \mathbb{R}^{n\times n} \mid X = X^T,\ X \succ 0\}$.

□

An important example of a convex set in optimization—even in non-convex optimization—is the *level set*.

Definition 3.3.3. If $f : \mathbb{R}^n \to \mathbb{R}$ and $\alpha \in \mathbb{R}$, we call the set $C_\alpha = \{x \in \mathbb{R}^n \mid f(x) \leq \alpha\}$ a level set, or the α-level set.

It is not difficult to see that if f is convex, C_α is convex for any $\alpha \in \mathbb{R}$. The converse is not true however. For example, the function $\sin(x)$ is not convex over $\mathbb{R}$ but its 1-level set is convex $\{x \in \mathbb{R} \mid \sin(x) \leq 1\} = \mathbb{R}$. For the function $-e^x$, *all* the level sets are convex, but the function is concave.

Consider the problem (III.1) where the function f is convex. The main result of this section shows the power of convexity. It is sufficient to identify a stationary point for this point is automatically a global minimizer.

Theorem 3.3.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable and let x be a stationary point for f. Then x is a global minimizer of f.*

Proof. The proof follows immediately from applying Theorem 3.3.1 and using the fact that $\nabla f(x) = 0$. □

3.4 Exercises

Exercise 3.4.1. Show that an isolated local minimizer is necessarily strict.

Exercise 3.4.2. Show that if f is convex, all its level sets are convex.

3.5 Globalization

To successfully tackle a smooth nonlinear nonconvex programming problem from a remote starting guess, the iteration must often be embedded into a *globalization* method. The two most popular globalization methods are the linesearch and the trust region. Their philosophies may be seen as dual; a linesearch strategy computes a step length along a predetermined direction, while a trust-region strategy considers all acceptable directions but limits the maximal step length.

As a general rule, algorithms do not explicitly use the second-order necessary and sufficient conditions. A reason for this is the computational cost of verifying positive definiteness and the indeterminacy associated to this computation due to finite precision. Instead we attempt to design algorithms which eventually satisfy the first-order necessary conditions and, as much as possible, are not attracted by saddle points. This is not always possible.

3.5.1 Linesearches

A linesearch algorithm is intuitive. Starting from an initial guess, an updated guess is obtained by searching for a point with a lower objective function value. Once this new guess obtained, the process repeats. The updated guess is found by first computing a direction along which we are confident that the objective decreases, at least initially. This direction is explored, we choose a certain step and move of that amount along the direction.

3.5.1.1 Descent Directions

According to Taylor's theorem, if f is differentiable, if our current point is x and if we consider points of the form $x + td$ for some direction d and step length $t \geq 0$,

$$f(x + td) - f(x) = t\nabla f(x)^T d + o(\|td\|).$$

For sufficiently small values of t, the little o is negligible and we can guarantee that $f(x + td) < f(x)$ by picking d so that $\nabla f(x)^T d < 0$. More precisely, there is an $\epsilon > 0$ such that for all $t \in]0, \epsilon[$, $f(x + td) < f(x)$.

Definition 3.5.1. If $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, a direction $d \in \mathbb{R}^n$ is called a *descent direction* for f at x if $\nabla f(x)^T d < 0$.

As long as $\nabla f(x) \neq 0$, there always exists a descent direction for f at x—for instance $d = -\nabla f(x)$. There exists no descent direction if $\nabla f(x) = 0$, but in this case we have found a stationary point.

The set of all descent directions for f at x form an open cone. In fact, they even form an open half-space.

3.5.1.2 The Steepest Descent Direction

Since the set of all descent directions is a cone, if d is a descent direction, so is λd for any $\lambda > 0$ and by letting $\lambda \to +\infty$, we can obtain inner products that are as negative as desired. For this reason, in this section, we limit the norm of the directions we consider.

Whenever $\nabla f(x) \neq 0$, there is a direction which produces the most negative inner product with the gradient. This direction solves the problem

$$\begin{array}{ll} \underset{d \in \mathbb{R}^n}{\text{minimize}} & \nabla f(x)^T d \\ \text{subject to} & \|d\| = 1. \end{array}$$

Using the Cauchy-Schwartz inequality, it is easy to find the unique solution to this problem since

$$\nabla f(x)^T d \geq -\|\nabla f(x)\| \|d\| = -\|\nabla f(x)\|$$

and this lower bound is attained for

$$d^{\mathrm{C}} = -\frac{\nabla f(x)}{\|\nabla f(x)\|}. \tag{III.6}$$

This direction d^{C} is called the *steepest descent direction* or sometimes the Cauchy direction, named after the french mathematician Augustin Cauchy.

3.5.1.3 The Newton Direction

Around x, let us approximate the real objective function f by a quadratic *model*, easier to manipulate, given by the second-order Taylor expansion of f about x

$$m(d) = f(x) + \nabla f(x)^T d + \frac{1}{2} d^T \nabla^2 f(x) d. \tag{III.7}$$

Imagine for a moment that $\nabla^2 f(x)$ is positive definite. Then the model m is a convex function of d and, from §3.3, we know that we can identify a global minimizer from

$$\nabla_d m(d) = \nabla f(x) + \nabla^2 f(x) d = 0.$$

The Newton direction is precisely this direction which can also be written

$$d^{\mathrm{N}} = -(\nabla^2 f(x))^{-1} \nabla f(x). \tag{III.8}$$

It is indeed a descent direction for f at x since the inverse Hessian is also positive definite and $\nabla f(x)^T d^{\mathrm{N}} = -\nabla f(x)^T (\nabla^2 f(x))^{-1} \nabla f(x) < 0$

When $f \in \mathcal{C}^2$, f and m differ by an amount of order $o(\|d\|^2)$ so that for small displacements, the model is accurate.

It is important to note that the Newton direction has an intrinsic *scaling*. Indeed if f *were* a convex quadratic, it would coincide with the model and $x^* = x + d$ would be a global minimizer of f. This property is no longer true if we scale d by some scalar λ. The steepest descent, on the other hand, has no such inherent scaling.

Whenever we do not have access to the Hessian matrix of f, or this matrix is not positive definite, the Newton direction is undefined.

3.5.1.4 Quasi-Newton Directions

When the Newton direction described in §3.5.1.3 is not defined, be it because we do not wish to compute the second-order derivatives, because the Hessian is not positive definite or because for some reason they are not available, it is commonplace to approximate them. To that end, the Hessian matrix in the model (III.7) is replaced by a symmetric positive definite matrix $B(x)$. Just as in §3.5.1.3, the corresponding direction is now

$$d^{\mathrm{QN}} = -B(x)^{-1} \nabla f(x) \tag{III.9}$$

and this direction is a descent direction for f at x. This direction, for its resemblance with the Newton direction, is called a quasi-Newton direction.

Exactly how the matrix $B(x)$ is constructed will determine how useful the direction d^{QN} is efficient in an algorithm. Methods using quasi-Newton directions can be very efficient if what second-order information is collected in the course of the iterations is incorporated into $B(x)$.

3.5.1.5 The Linesearch

Once a descent direction is chosen, a steplength along this direction must be computed. In an ideal world we would like to solve the univariate problem

$$\min_{t>0} \; f(x + td).$$

We know however that even for univariate problems, finding a global minimizer is out of the question and even a local minimizer will require a full-fledged algorithm. For those reasons, we will be satisfied if we can simply identify a value of $t > 0$ such that $f(x + td) < f(x)$. Identifying such a step length is referred to as *performing a linesearch*.

We obtain our first template for algorithms based on descent directions.

It is important to realize that the steplength must be wisely chosen and not randomly selected or selected based on some loose criterion.

Algorithm 3.5.1: [Template Descent Algorithm]

Step 0. Choose a starting point $x_0 \in \mathbb{R}^n$ and a stopping tolerance $\epsilon_0 > 0$. Initialize an iteration counter k to 0.

Step 1. If $\|\nabla f(x_k)\| \le \epsilon_0$, terminate successfully with solution x_k. Otherwise, go to Step 2.

Step 2. From x_k, select a descent direction d_k and compute a steplength t_k such that $f(x_k + t_k d_k) < f(x_k)$.

Step 3. Set $x_{k+1} = x_k + t_k d_k$, increment k by one and return to Step 1.

Example 3.5.1. Consider the function $f(x) = x^2$ and $x_0 = 2$. At each step we select the steepest descent direction which is in this case $d_k = (-1)^{k+1}$ and the steplength $t_k = 2 + 3/2^{k+1}$. Algorithm 3.5.1 generates the sequence $x_1 = -3/2$, $x_2 = 5/4$, $\ldots$, $x_k = (-1)^k(1 + 1/2^k)$. At each step we have $f(x_{k+1}) < f(x_k)$. The sequence $\{x_k\}$ does not converge, has the two limit points -1 and 1, neither of which is critical. The sequence $\{f(x_k)\} \to 1$ which is not a locally optimal value. This iteration is illustrated in Fig. III.3. The reason why things are going wrong in this example is that after a few iterations, we see that we are taking very long steps, of length around 2, for negligible decrease in the objective function. $\square$

Example 3.5.2. Consider again $f(x) = x^2$, $x_0 = 2$ and this time $d_k = -1$ and $t_k = 1/2^{k+1}$. The sequence generated is this time $x_1 = 3/2$, $x_2 = 5/4$, $\ldots$, $x_k = 1 + 1/2^k$. The sequence $\{x_k\}$ converges to 1 which is not critical, even though $f(x_{k+1}) < f(x_k)$ for all k. This iteration is illustrated in Fig. III.3. The reason why things are going wrong in this situation as well is that the steplength is converging to zero too quickly to allow sufficient progress. $\square$

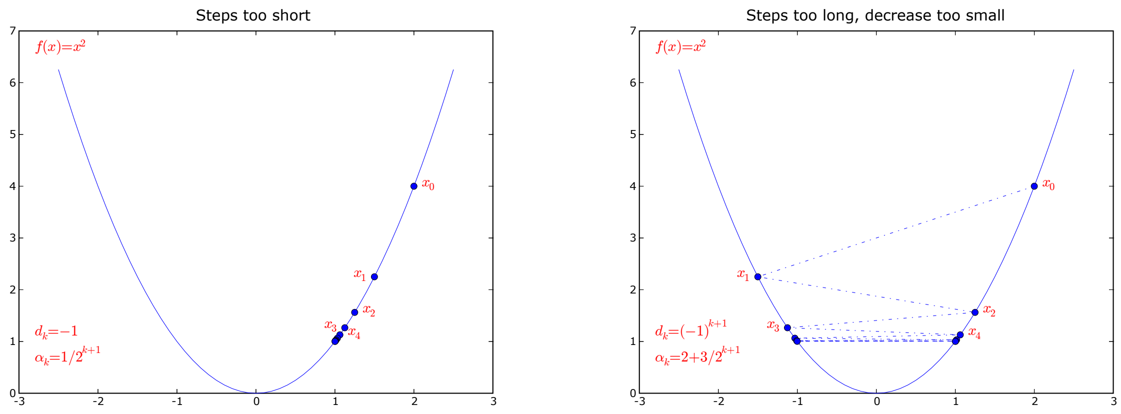


Figure III.3: Left: Steps too long for minimal objective decrease. Right: Steps converge to zero too fast.

In view of Examples 3.5.1 and 3.5.2, it is important that the steplength and the decrease achieved in the objective be in proportion. Example 3.5.2 also illustrate that our criterion $f(x + td) < f(x)$ is too loose. Instead we must specify that f decreases *sufficiently*. There are many ways to specify this.

3.5.1.5.1 The Armijo Condition.

We impose that at iteration k, the steplength t_k satisfies the Armijo condition

$$f(x_k + t_k d_k) \le f(x_k) + \alpha t_k \nabla f(x_k)^T d_k, \qquad \text{(III.10)}$$

where $\alpha \in]0,1[$ is fixed. Note that the second term on the right-hand side is negative because d_k is a descent direction. Therefore, the larger the steplength, the more demanding condition III.10 is on the decrease of f. On the other hand, for small steplengths, small decreases are acceptable.

On the left-hand side of (III.10), we recognize the restriction of f to the direction d_k. It is convenient to define the function

$$\phi : \mathbb{R}^+ \to \mathbb{R}, \qquad \phi(t) = f(x_k + td_k) \tag{III.11}$$

and to note that $\phi(0) = f(x_k)$, $\phi'(t) = \nabla f(x_k + td_k)^T d_k$ and $\phi'(0) = \nabla f(x_k)^T d_k < 0$.

Acceptable values of the steplength are illustrated on an imaginary function in Fig. III.4.

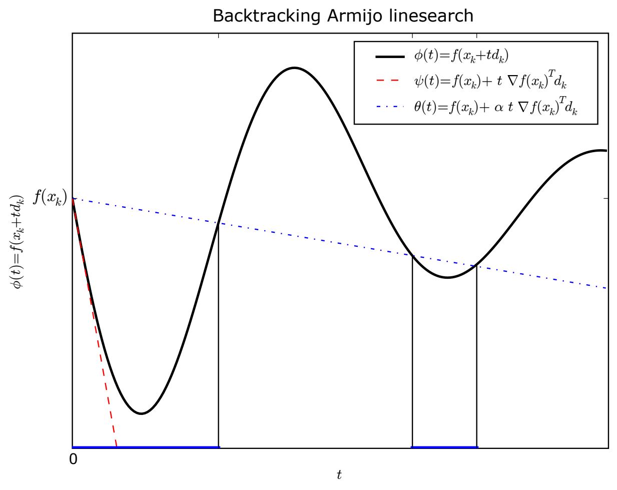


Figure III.4: Illustration of the Armijo condition. In this example, there are two intervals of acceptable steplengths, where the curve lies below the dotted blue line. In all cases, there is one of the form $]0, \epsilon]$.

There is always an interval of the form $]0, \epsilon]$ of steplengths that are acceptable according to the Armijo condition.

Theorem 3.5.1. *If d_k is a descent direction for f at x_k, there is $\epsilon > 0$ such that for all $t \in]0, \epsilon]$, (III.10) is satisfied.*

Proof. By definition of the derivative,

$$\nabla f(x_k)^T d_k = \lim_{t \to 0} \frac{f(x_k + td_k) - f(x_k)}{t}$$

and thus

$$\lim_{t \to 0} \frac{f(x_k + td_k) - f(x_k)}{t \nabla f(x_k)^T d_k} = 1.$$

Consequently, since $\alpha \in]0,1[$, there must be an $\epsilon > 0$ such that the general term in the limit exceeds α, which is precisely (III.10), since $\nabla f(x_k)^T d_k < 0$. $\square$

Theorem 3.5.1 eliminates the problem with steps that are too long. We need an additional safeguard to rule out steps that are too small. Thanks to Theorem 3.5.1 this can be done by means of a *backtracking*. A backtracking consists in choosing an initial value of t, for instance $t = 1$, and decreasing t by a constant factor until the Armijo condition is satisfied. It is specified more precisely in Algorithm 3.5.2.

Algorithm 3.5.2: [Backtracking Linesearch]

```
Step 0. Let $t_{(0)} = 1$, $\beta \in ]0,1[$ and set $j = 0$.
Step 1. If (III.10) is satisfied with $t_{(j)}$, exit successfully with $t_k = t_{(j)}$. Otherwise go to Step 2.
Step 2. Let $t_{(j+1)} = \beta t_{(j)}$, increment $j$ by one and return to Step 1.
```

The procedure of Algorithm 3.5.2 is *opportunistic*; the first acceptable steplength—i.e. the largest—is accepted. We thus avoid small steps. In practice, an optimistic value of α such as 10^{-4} is chosen so as to increase chances of accepting a step early.

To study the convergence of direction-based methods using an Armijo linesearch with backtracking, we start by strenghtening Theorem 3.5.1 under the assumption that $\nabla f(x)$ is *Lipschitz continuous*.

Theorem 3.5.2. *Assume $f \in \mathcal{C}^1$ about $x \in \mathbb{R}^n$ and that there exists a constant $L(x) > 0$ and $\epsilon > 0$ such that for all $y \in B(x,\epsilon)$, $\|\nabla f(x) - \nabla f(y)\| \le L(x)\|x-y\|$. Let d be a descent direction for f at x. Then, for all*

$$t \le \frac{2(\alpha-1)\nabla f(x)^T d}{L(x)\|d\|^2}, \tag{III.12}$$

the Armijo condition (III.10) is satisfied.

Proof. By the mean-value theorem and Lipschitz continuity, we successively have

$$\begin{aligned} f(x+td) - f(x) &= \int_0^1 \nabla f(x+\theta td)^T (td)\ \mathrm{d}\theta \\ &= t\nabla f(x)^T d + t\int_0^1 \left[\nabla f(x+\theta td) - \nabla f(x)\right]^T d\ \mathrm{d}\theta \\ &\le t\nabla f(x)^T d + t\int_0^1 \|\nabla f(x+\theta td) - \nabla f(x)\|\|d\|\ \mathrm{d}\theta \\ &\le t\nabla f(x)^T d + \frac{1}{2}t^2 L(x)\|d\|^2. \end{aligned}$$

This last expression is less than the right-hand side of (III.10) whenever

$$\frac{1}{2}L(x)t\|d\|^2 \le (\alpha-1)\nabla f(x)^T d,$$

which completes the proof. □

Note that the Lipschitz-continuity assumption is stronger than assuming that $f \in \mathcal{C}^2$.

Theorem 3.5.2 has an immediate corollary, yielding an upper bound on the step produced by Algorithm 3.5.2.

Corollary 3.5.1. *Under the same assumptions as Theorem 3.5.2, the step t_k produced by Algorithm 3.5.2 from x_k along the descent direction d_k satisfies*

$$t_k \ge \min\left(t_{(0)}, \beta\frac{2(\alpha-1)\nabla f(x_k)^T d_k}{L(x_k)\|d_k\|^2}\right). \tag{III.13}$$

The bound (III.12) will allow us to give a very general condition ensuring global convergence of linesearch-based algorithms for unconstrained programming.

An Armijo linesearch with backtracking is very easy and fast to implement. There are other types of requirements on the steplength which do not need to be supplemented by a backtracking procedure.

3.5.1.5.2 The Wolfe Conditions. Instead of backtracking, small steps are ruled out by imposing a *curvature* condition in addition to the Armijo condition. The two together are known as the *Wolfe conditions*

$$\begin{aligned} f(x_k + t_k d_k) &\leq f(x_k) + \alpha t_k \nabla f(x_k)^T d_k, && \text{(III.14)} \\ \nabla f(x_k + t_k d_k)^T d_k &\geq \gamma_1 \nabla f(x_k)^T d_k, && \text{(III.15)} \end{aligned}$$

where $\alpha \in]0,1[$ and $\gamma_1 \in]\alpha, 1[$.

Condition (III.15) imposes a restriction on the slope of the restriction of f at the trial point. In terms of the function (III.11), this condition can be restated as

$$\phi'(t_k) \geq \gamma_1 \phi'(0).$$

The Wolfe conditions are illustrated in Fig. III.5.

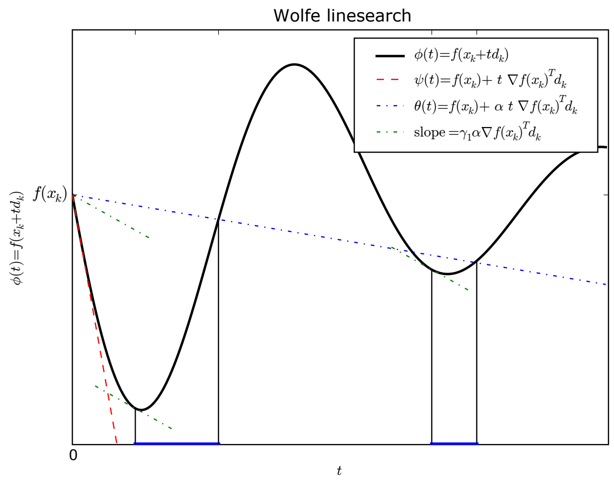


Figure III.5: Illustration of the Wolfe conditions. In this example, there are two intervals of acceptable steplengths, where the curve lies below the dotted blue line and where the curvature condition is met.

It is no longer true that the Wolfe conditions are satisfied over an interval of the form $]0, \epsilon]$ with $\epsilon > 0$. Identifying an acceptable steplength is more difficult and the standard method resorts to interpolation. We briefly describe the procedure here, without details.

(a) Start with an initial value $t_{(0)}$ and evaluate $\phi(t_{(0)}$. If the Wolfe conditions are met, terminate. Otherwise, build a quadratic interpolant $q(t)$ such that $q(0) = \phi(0)$, $q'(0) = \phi'(0)$ and $q(t_{(0)}) = \phi(t_{(0)})$. The $t_{(1)}$ which minimizes $q(t)$ can be found explicitely.

(b) Evaluate $\phi(t_{(1)})$. If the Wolfe conditions are still not met at $t_{(1)}$, build a cubic interpolant by adding the condition $q(t_{(1)}) = \phi(t_{(1)})$. Find $t_{(2)}$, the minimizer of this new interpolant.

(c) Repeat the procedure by successively maintaining cubic interpolants such that $q(0) = \phi(0)$, $q'(0) = \phi'(0)$ and q interpolates ϕ at the last two trial steplengths $t_{(j-1)}$ and $t_{(j-2)}$.

It is still possible to obtain a lower bound on the step size produced by a Wolfe linesearch, and we obtain a result which parallels Corollary 3.5.1.

Theorem 3.5.3. *Assume $f \in \mathcal{C}^1$ about $x_k \in \mathbb{R}^n$ and that there exists a constant $L(x_k) > 0$ and $\epsilon > 0$ such that for all $y \in B(x_k, \epsilon)$, $\|\nabla f(x_k) - \nabla f(y)\| \leq L(x_k)\|x_k - y\|$. Let d_k be a descent direction for f at x_k. Then, a step t_k satisfying the Wolfe conditions (III.14)–(III.15) is such that*

$$t_k \geq \frac{(\gamma_1 - 1)\nabla f(x)^T d}{L(x)\|d\|^2}. \tag{III.16}$$

Proof. From (III.15), and the fact that $x_{k+1} = x_k + t_k d_k$, we have

$$(\nabla f(x_{k+1}) - \nabla f(x_k))^T d_k \geq (\gamma_1 - 1)\nabla f(x_k)^T d_k.$$

Using Lipschitz continuity,

$$(\nabla f(x_{k+1}) - \nabla f(x_k))^T d_k \leq \|\nabla f(x_{k+1}) - \nabla f(x_k)\|\|d_k\| \leq L(x_k)t_k\|d_k\|^2.$$

Combining the two inequalities proves the theorem. □

3.5.1.5.3 The Strong Wolfe Conditions. Steplengths satisfying the Wolfe conditions (III.14)–(III.15) are not necessarily close to a local minimizer of ϕ. This can be desirable in certain Newton-like methods. From Fig. III.5, we see that this can be imposed by changing the curvature conditions. We thus obtain the *strong Wolfe* conditions

$$f(x_k + t_k d_k) \leq f(x_k) + \alpha t_k \nabla f(x_k)^T d_k, \tag{III.17}$$
$$|\nabla f(x_k + t_k d_k)^T d_k| \leq \gamma_2 |\nabla f(x_k)^T d_k|, \tag{III.18}$$

where $\alpha \in]0, 1[$ and $\gamma_2 \in]\alpha, 1[$.

Unlike the Wolfe conditions, the strong Wolfe conditions rule out values of t where $\phi'(t)$ has large positive values. This is illustrated in Fig. III.6. Note however that neither of the three types of conditions on the steplength rules out local maximizers of ϕ that fall under the line defined by the Armijo condition.

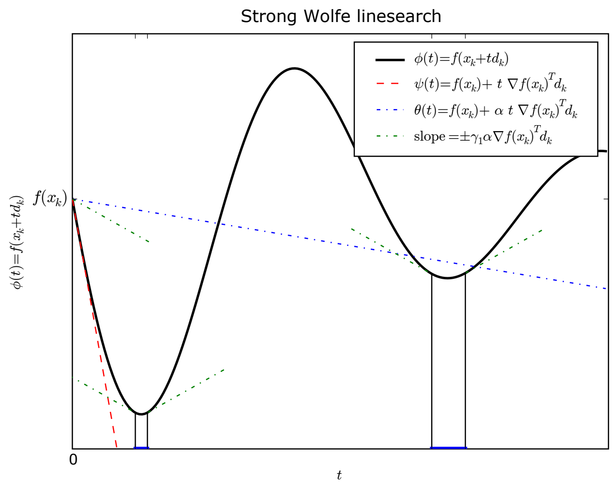


Figure III.6: Illustration of the strong Wolfe conditions. In this example, there are two intervals of acceptable steplengths, where the curve lies below the dotted blue line and where the curvature condition is met.

The strong Wolfe conditions appear to be more stringent but it turns out that it is not more difficult to identify a steplength satisfying (III.18) than one satisfying (III.15).

Theorem 3.5.4. *Let $f \in \mathcal{C}^1$, let d be a descent direction for f at $x \in \mathbb{R}^n$ and assume ϕ is bounded below on $\mathbb{R}^+$. Then there exist intervals of $\mathbb{R}$ where the conditions (III.14)–(III.15) and (III.17)–(III.18) are satisfied.*

Since the strong Wolfe conditions are stronger than the Wolfe conditions, Theorem 3.5.3 still holds.

Theorem 3.5.5. *Assume $f \in \mathcal{C}^1$ about $x_k \in \mathbb{R}^n$ and that there exists a constant $L(x_k) > 0$ and $\epsilon > 0$ such that for all $y \in B(x_k, \epsilon)$, $\|\nabla f(x_k) - \nabla f(y)\| \leq L(x_k)\|x_k - y\|$. Let d_k be a descent direction for f at x_k. Then, a step t_k satisfying the strong Wolfe conditions (III.17)–(III.18) is such that*

$$t_k \geq \frac{(\gamma_2 - 1)\nabla f(x)^T d}{L(x)\|d\|^2}. \tag{III.19}$$

Proof. Note that (III.18) implies (III.15). The proof is identical to that of Theorem 3.5.3. □

3.5.1.6 Global Convergence of Linesearch Methods

Linesearch-based descent algorithms differ from one another in two respects: the choice of descent direction and the choice of conditions imposed on the step.

From §3.5.1.5, and in particular Corollary 3.5.1, Theorem 3.5.3 and Theorem (3.5.5) it is clear that the conditions we imposed on the step ensure that the steplength is large enough. This guarantees that if the angle between the search direction and the gradient does not approach $\pi/2$ too fast or at all, Algorithm 3.5.1 generates a sequence $\{x_k\}$ such that

$$\lim_{k\to\infty} \nabla f(x_k) = 0. \tag{III.20}$$

Property (III.20) is what we call *global convergence*. It is about as strong a result as can be obtained from this framework. It is important to realize what (III.20) does *not* say about the sequence $\{x_k\}$. In particular, it does *not* say that $\{x_k\}$ is convergent or even that it has a limit point.

Example 3.5.3. Assume we are minimizing $f(x) = e^{-x}$ over $\mathbb{R}$ and consider the sequence $x_k = k$ for all k. Then, $\lim_{k\to\infty} f'(x_k) = \lim_{k\to\infty} -e^{-k} = 0$ but $\{x_k\}$ has no limit point. □

However, *if* $\{x_k\}$ has a limit point x^*, then by continuity we have from (III.20) that $\nabla f(x^*) = 0$ and this limit point is stationary. It will be important to have appropriate mechanisms or assumptions to guarantee that the sequences generated by our algorithms have limit points. It is common to see results with assumptions similar to

- "Assume $\{x_k\}$ is bounded ..."
- "Assume $\{x_k\}$ remains in a closed and bounded set ..."
- "Assume the level set $\{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\}$ is bounded ..."

or more generally,

- "Assume $\{x_k\}$ has a limit point ..."

Definition 3.5.2. The angle between a descent direction d_k and the opposite of the gradient of f at x_k is denoted θ_k and is given by

$$\cos(\theta_k) = \frac{-\nabla f(x_k)^T d_k}{\|\nabla f(x_k)\|\|d_k\|}. \tag{III.21}$$

The following result immediately implies global convergence of all linesearch methods for which there exists $\bar{\theta} > 0$ such that $|\theta_k| \leq \bar{\theta}$ for all k.

Theorem 3.5.6. *Let Algorithm 3.5.1 be a linesearch-based descent method imposing either type of condition of §3.5.1.5 on the steplength. Let* $\mathcal{L} = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\}$ *and assume* f *is* $\mathcal{C}^1$ *over an open set containing* $\mathcal{L}$ *and that* ∇f *is Lipschitz continuous with Lipschitz constant* $L > 0$ *over* $\mathcal{L}$*. Then either* $\{f(x_k)\} \to -\infty$ *or*

$$\sum_{k=0}^{\infty} \cos^2\theta_k \|\nabla f(x_k)\|^2 < +\infty. \tag{III.22}$$

Proof. The assumptions of Corollary 3.5.1, Theorem 3.5.3 and Theorem (3.5.5) are satisfied. Substituting $L(x) = L(x_k) = L$ in (III.13), (III.16) and (III.19), we see that in all cases, there is constant $\epsilon \in]0,1[$ such that

$$t_k \geq -\epsilon \frac{\nabla f(x_k)^T d_k}{L\|d_k\|^2}.$$

By substituting in the Armijo condition (III.10), we have successively

$$\begin{aligned}
f(x_{k+1}) &\leq f(x_k) - \frac{\alpha\epsilon}{L}\left(\frac{\nabla f(x_k)^T d_k}{\|d_k\|}\right)^2 \\
&= f(x_k) - \frac{\alpha\epsilon}{L}\cos^2\theta_k\|\nabla f(x_k)\|^2 \\
&\leq f(x_{k-1}) - \frac{\alpha\epsilon}{L}\cos^2\theta_{k-1}\|\nabla f(x_{k-1})\|^2 - \frac{\alpha\epsilon}{L}\cos^2\theta_k\|\nabla f(x_k)\|^2 \\
&\leq \dots \\
&\leq f(x_0) - \frac{\alpha\epsilon}{L}\sum_{j=0}^{k}\cos^2\theta_j\|\nabla f(x_j)\|^2.
\end{aligned} \tag{III.23}$$

At this point, either f is unbounded below and the theorem is proved or there exists $M \in \mathbb{R}$ such that for all $x \in \mathcal{L}$, $f(x) \geq M$, and in particular $f(x_k) \geq M$ for all k. We then have from (III.23) that

$$\sum_{j=0}^{k}\cos^2\theta_j\|\nabla f(x_j)\|^2 \leq \frac{L(f(x_0) - M)}{\alpha\epsilon}$$

for all k. Therefore, taking the limit $k \to \infty$, we obtain (III.22). □

Condition (III.22) is the *Zoutendijk* condition. It is a crucial result in the theory of linesearch methods. Since the series is convergent we must have

$$\lim_{k\to\infty} \cos^2\theta_k\|\nabla f(x_k)\|^2\} = 0.$$

Consequently, if there exists $\bar{\theta} > 0$ such that $|\theta_k| \leq \bar{\theta}$ for all k, then (III.20) must be satisfied and we have global convergence.

3.5.1.7 The Steepest Descent Method

The *steepest descent method* is Algorithm 3.5.1 where for all k, the descent direction $d_k = -\nabla f(x_k)$ is used. Alternatively, we can always choose a multiple of this direction since it does not have any particular scaling associated to it.

Theorem 3.5.6 immediately implies global convergence of the steepest descent method, since in this case $\theta_k = 0$ for all k. Therefore, in the steepest descent method, either f is unbounded below or any limit point of $\{x_k\}$ is stationary.

Due to a property of the gradient, the steepest descent, however globally convergent, often performs rather poorly. Indeed, the gradient is always orthogonal to the level curves and this causes the method to oscillate heavily, even close to a local minimizer. For this reason, a lot of iterations are often necessary to obtain reasonable accuracy. This is clearly in disfavor of the method and, if the objective function and its gradient are expensive to evaluate, can become a reason to choose against it. The steepest descent method remains nevertheless an important method to conceptualize and illustrate most principles in optimization.

Example 3.5.4. The Rosenbrock function is a classic in unconstrained optimization and is defined by $f : \mathbb{R}^2 \to \mathbb{R}$, $f(x, y) = 100(y - x^2)^2 + (1 - x)^2$. It is often called the *banana* function because of the shape of its contours. It is easy to see that this function has a single local minimizer at $(1, 1)$. We choose the starting point $(x_0, y_0) = (-0.5, 1)$ and the stopping tolerance $\epsilon_0 = 10^{-5}$. No less than 7379 iterations are necessary to reach this moderate level of accuracy due to the intense oscillations of the method. The progress of the method is illustrated in Fig. III.7 and appears in more detail in Table 3.1. □

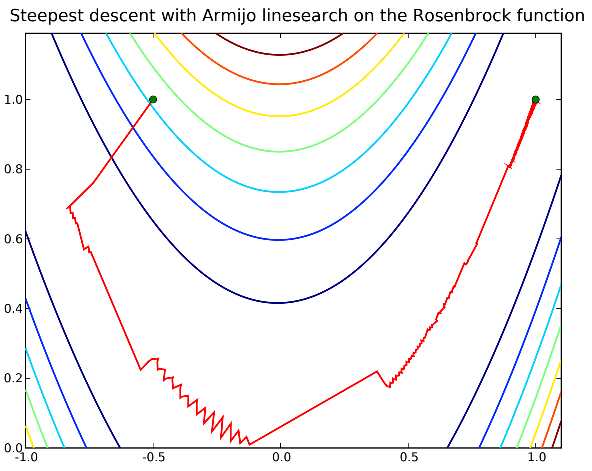


Figure III.7: Illustration of the steepest descent method applied to the Rosenbrock function.

It is informative to study how the method performs once it has reached a neighbourhood of a solution. This is called the *local convergence* properties of the method. It is customary in local convergence analysis to assume that the iterates are converging to a solution and the point of the analysis is *how* or *how fast* this convergence is occuring.

For the steepest descent method, we have the following result.

Theorem 3.5.7. *If $f : \mathbb{R}^n \to \mathbb{R}$ is $\mathcal{C}^2$ over an open ball around the stationary point x^*. Assume the sequence $\{x_k\}$ is generated by the algorithm where the linesearches are* exact *and that $\{x_k\} \to x^*$. If $\nabla^2 f(x^*)$ is positive definite, then*

$$f(x_{k+1}) - f(x^*) \le \left(\frac{\lambda^*_{\max} - \lambda^*_{\min}}{\lambda^*_{\max} + \lambda^*_{\min}} \right)^2 (f(x_k) - f(x^*)),$$

where $\lambda^*_{\min}$ and $\lambda^*_{\max}$ are the smallest and largest eigenvalue of $\nabla^2 f(x^*)$ respectively.

This result typically indicates a linear rate of convergence in the sequence $\{f(x_k)\}$. Depending on the condition number of $\nabla^2 f(x^*)$ this linear rate is close to 0 or close to 1. Assume the condition number is close to 1, then $\lambda^*_{\min} \approx \lambda^*_{\max}$ and Theorem 3.5.7 implies that the decrease achieved in f from an iteration to the next is very substantial. If however $\lambda^*_{\max} \gg \lambda^*_{\min}$, then the decrease is minimal.

k	$f(x_k, y_k)$	$\|\nabla f(x_k, y_k)\|$	x_k	y_k	t_k	$\cos\theta_k$
0	58.5	150	-0.5	1	0	-1
1	7.82811	61.0746	-0.7352	0.76	0.0016	-1
2	3.36118	4.99467	-0.832919	0.689766	0.0016	-1
3	3.34146	2.10729	-0.824928	0.691042	0.0016	-1
4	3.32694	4.97752	-0.823543	0.674184	0.008	-1
5	3.30695	2.06153	-0.815579	0.675476	0.0016	-1
6	3.28925	4.50077	-0.813431	0.658984	0.008	-1
$\vdots$						
7372	2.08363e-10	1.74672e-05	0.999986	0.999971	0.0016	-1
7373	2.08041e-10	1.2911e-05	0.999986	0.999971	0.0016	-1
7374	2.07755e-10	1.36776e-05	0.999986	0.999971	0.0016	-1
7375	2.06988e-10	3.59679e-05	0.999986	0.999971	0.008	-1
7376	2.06361e-10	2.05725e-05	0.999986	0.999971	0.0016	-1
7377	2.05966e-10	1.67022e-05	0.999986	0.999971	0.0016	-1
7378	2.05655e-10	1.47675e-05	0.999986	0.999971	0.0016	-1
7379	2.05375e-10	9.70339e-06	0.999986	0.999971	0.0016	-1

Table 3.1: Detail of the progress of the steepest method applied to the Rosenbrock function.

Example 3.5.5. At the minimizer $x^* = (1, 1)$ of the Rosenbrock function, we have

$$\nabla^2 f(x^*) = \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix}$$

whose eigenvalues are $\lambda^*_{\text{min}} \approx 0.4$ and $\lambda^*_{\text{max}} \approx 1001$ so the ratio in Theorem 3.5.7 is approximately 0.999. This explains the slow convergence that we observe numerically. □

There is a similar result regarding the convergence of $\{x_k\}$ to x^*. Taylor's theorem reveals that, under the assumptions of Theorem 3.5.7,

$$\|x_{k+1} - x^*\| \le 2\sqrt{\frac{\lambda^*_{\text{max}}}{\lambda^*_{\text{min}}}} \left(\frac{\lambda^*_{\text{max}} - \lambda^*_{\text{min}}}{\lambda^*_{\text{max}} + \lambda^*_{\text{min}}} \right) \|x_k - x^*\|, \tag{III.24}$$

in Euclidian norm. The convergence is also linear.

Exercise 3.5.1. Prove (III.24) using Taylor's theorem and Theorem 3.5.7.

3.5.1.8 Newton's Method

It is difficult to study the global convergence of Newton's method since when x_k is far from a local minimizer, there is no reason for $\nabla^2 f(x_k)$ to be positive definite. If by chance it happens to be, be it because f is strictly convex or simply because we are lucky, then the global convergence analysis of §(3.5.1.10) below applies.

Example 3.5.6. Newton's method on the Rosenbrock function, defined in Example 3.5.4, can be applied and, by chance, the Hessian does not need to be modified. The starting point is $(x_0, y_0) = (-0.5, 1)$ and the stopping tolerance is $\epsilon_0 = 10^{-8}$. Attaining this accuracy is a matter of 19 iterations. The path followed by the iterates is represented in Fig. III.8 and a closer look at the iterations appears in Table.3.2. □

The obvious question is, what should an algorithm do if second derivatives are available at reasonable cost but $\nabla^2 f(x_k)$ is not positive definite? A possible answer is the modified Newton method described in §3.5.1.9. Another answer will be provided by trust-region methods.

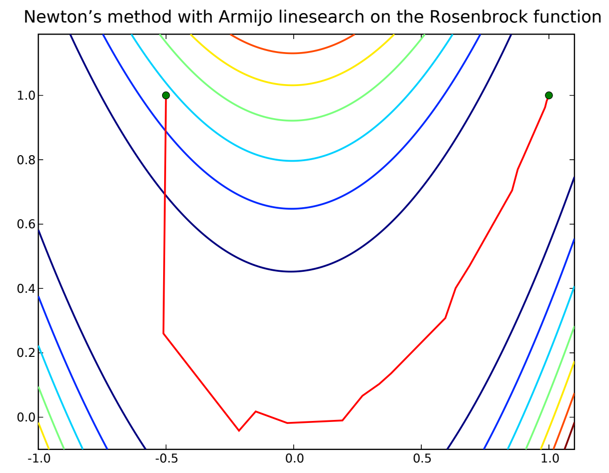


Figure III.8: Illustration of Newton's method applied to the Rosenbrock function.

k	$f(x_k, y_k)$	$\|\nabla f(x_k, y_k)\|_\infty$	x_k	y_k	t_k	$\cos\theta_k$
0	58.5	150	-0.5	1	0	-0.723669
1	2.2803	3.04081	-0.510067	0.260067	1	-0.695273
2	2.24314	17.541	-0.214054	-0.041886	0.2	-0.950388
3	1.32106	2.55196	-0.148574	0.0177868	1	-0.821578
4	1.08549	3.74477	-0.0249061	-0.0181035	0.2	-0.544007
5	0.872026	9.33185	0.191101	-0.0101395	1	-0.538353
6	0.537544	1.22592	0.269393	0.0664431	1	-0.885299
7	0.450662	1.8426	0.335039	0.103038	0.2	-0.619958
8	0.39128	1.91186	0.381824	0.13623	0.2	-0.534205
9	0.367866	9.89897	0.59412	0.307909	1	-0.322664
10	0.133749	0.32856	0.634652	0.40114	1	-0.99528
11	0.0981994	0.867829	0.689651	0.471279	0.2	-0.358041
12	0.0970093	9.16227	0.855806	0.704796	1	-0.214888
13	0.0149283	0.0977756	0.877916	0.770248	1	-0.992649
14	0.0100468	0.177162	0.900158	0.809399	0.2	-0.456197
15	0.00540067	2.80418	0.984974	0.96298	1	-0.164745
16	7.8726e-05	0.00759261	0.991135	0.982311	1	-0.991924
17	6.0357e-07	0.0308251	0.999933	0.999789	1	-0.159733
18	1.03882e-12	8.6541e-07	0.999999	0.999998	1	-0.992428
19	1.0828e-22	4.12972e-10	1	1		

Table 3.2: Detail of the progress of Newton's method applied to the Rosenbrock function.

Locally, assume for now that the iterates are in a region around some x^* in which $\nabla^2 f(x)$ is positive definite and assume the iterates are generated from

$$x_{k+1} = x_k + d_k \quad \text{where} \quad d_k = -\nabla^2 f(x_k)^{-1}\nabla f(x_k). \tag{III.25}$$

The following result gives the local rate of convergence of Newton's method

Theorem 3.5.8. *Assume $f : \mathbb{R}^n \to \mathbb{R}$ is $\mathcal{C}^2$ over a ball centered at x^* where $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite. Assume $\nabla^2 f(\cdot)$ is Lipschitz-continuous around x^*, i.e., there exist $\delta > 0$ and $L > 0$ such that for all $x, y \in B(x^*, \delta)$,*

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \le L\|x - y\|.$$

Then there exists $\epsilon > 0$ such that if $x_0 \in B(x^*, \epsilon)$,

(a) the sequence $\{x_k\}$ generated by (III.25) converges to x^* and the rate of convergence is quadratic, i.e., there exists $C > 0$ such that for all sufficiently large k,

$$\|x_{k+1} - x^*\| \le C\|x_k - x^*\|^2,$$

(b) the sequence $\{f(x_k)\}$ converges to $f(x^*)$ quadratically.

Note that in Theorem 3.5.8, the rate of convergence is independent of the condition number of $\nabla^2 f(x^*)$, unlike that of the steepest descent method.

3.5.1.9 Modified Newton Methods

When the Newton direction is not a descent direction, a possibility is to compute a slightly different direction by modifying the Hessian of the objective. If we want to retain the good properties of the quadratic model, this modification should be as small as possible. Instead of Newton's direction we compute a direction from the linear system

$$(\nabla^2 f(x_k) + M_k)d_k = -\nabla f(x_k),$$

where M_k is an $n \times n$ symmetric matrix. M_k is chosen such that

(a) $M_k = 0$ if $\nabla^2 f(x_k)$ is positive definite and has uniformly bounded condition number,

(b) $\nabla^2 f(x_k) + M_k$ is positive definite if $\nabla^2 f(x_k)$ was not positive definite. The modified Hessian should also have uniformly bounded condition number.

The problem of finding the smallest modification M so that all eigenvalues of $\nabla^2 f(x_k) + M$ are larger than a safety threshold $\epsilon > 0$ could be formulated as

$$\begin{array}{ll} \underset{M}{\text{minimize}} & \|M\|^2 \\ \text{subject to} & M = M^T, \\ & \nabla^2 f(x_k) + M - \epsilon I \succeq 0, \end{array} \tag{III.26}$$

where $\|\cdot\|$ is some matrix norm. Different choices of norm give rise to different modifications.

Since $\nabla^2 f(x_k)$ is symmetric there are orthonormal eigenvectors $q_1, \ldots, q_n$ and eigenvalues $\lambda_1, \ldots, \lambda_n$ such that $\nabla^2 f(x_k) = \sum_{i=1}^n \lambda_i q_i q_i^T$. To ensure that M_k is the smallest modification such that all the eigenvalues of the modified Hessian are at least $\epsilon > 0$, we can choose M_k such that

$$\nabla^2 f(x_k) + M_k = \sum_{i=1}^n \max(\lambda_i, \epsilon) q_i q_i^T.$$

This modification has a price, however, as it requires to compute a basis of orthonormal eigenvectors $\{q_i\}$. This can be done either by computing directly the singular-value decomposition of $\nabla^2 f(x_k)$ or iteratively using the Lanczos method and Rayleigh-Ritz approximations to eigenvalue-eigenvector pairs. In all cases, this process quickly becomes costly as the number of variables becomes large and, in the case of the Lanczos iterative method, will likely require preconditioning. Note also that in general, this M_k is not diagonal. It can be proved to solve the problem (III.26) in the Frobenius norm, $\|M\|_F^2 = \sum_{i,j} M_{ij}^2$.

A cheaper alternative is to set $M_k = \alpha I$ for some $\alpha > 0$ and to choose α such that all eigenvalues of the modified Hessian are sufficiently positive. Because of this particular form of M_k, we have $M_k = \alpha \sum_{i=1}^n q_i q_i^T$ and we pick $\alpha = \max(0, \epsilon - \lambda_{\min}(\nabla^2 f(x_k)))$. This only requires computation of the smallest eigenvalue of the Hessian at each iteration. This can also be done by means of the Lanczos method. A disadvantage of this approach is that very negative eigenvalues may be overly emphasized by the modification. If $\lambda_{\min}(\nabla^2 f(x_k)) \ll -1$ and all other eigenvalues are of order 1, then

$$\nabla^2 f(x_k) + M_k \approx M_k = (\epsilon - \lambda_{\min}(\nabla^2 f(x_k)))I$$

and this is producing a direction which is essentially the steepest descent direction. This modification M_k solves (III.26) in Euclidian norm, $\|M\|_2^2 = \lambda_{\max}(M)$.

A compromise is to attempt a Cholesky factorization of $\nabla^2 f(x_k)$ and to modify the Cholesky factors directly if there is evidence that the factorization will otherwise fail. This *modified Cholesky* factorization comes in different forms but always produces the decomposition $\nabla^2 f(x_k) + M_k = L_k L_k^T$. This can be done while at the same time ensuring that the matrices $B_k = \nabla^2 f(x_k) + M_k$ keep a uniformly bounded condition number.

In all cases, we say that Newton's quadratic model was *convexified*.

3.5.1.10 Quasi-Newton Methods

If a descent method is based on the direction $d_k^{\text{QN}} = -B_k^{-1} \nabla f(x_k)$ where B_k is a symmetric positive definite approximation to $\nabla^2 f(x_k)$, we have by substitution

$$\cos \theta_k = \frac{\nabla f(x_k) B_k^{-1} \nabla f(x_k)}{\|\nabla f(x_k)\| \|B_k^{-1} \nabla f(x_k)\|}.$$

For symmetric positive definite B_k we always have, in Euclidian norm,

$$\nabla f(x_k) B_k^{-1} \nabla f(x_k) \geq \lambda_{\min}(B_k^{-1}) \|\nabla f(x_k)\|^2 = \frac{1}{\lambda_{\max}(B_k)} \|\nabla f(x_k)\|^2$$

and

$$\|B_k^{-1} \nabla f(x_k)\| \leq \lambda_{\max}(B_k^{-1}) \|\nabla f(x_k)\| = \frac{1}{\lambda_{\min}(B_k)} \|\nabla f(x_k)\|,$$

where $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ denote the smallest and largest eigenvalue of the matrix A respectively. Therefore, we obtain the bound

$$|\cos \theta_k| \geq \frac{\lambda_{\min}(B_k)}{\lambda_{\max}(B_k)} = \frac{1}{\kappa_2(B_k)}, \tag{III.27}$$

where $\kappa_2(A)$ is the *condition number* of the matrix A.

Thus, if for all k, $\lambda_{\min}(B_k)$ remains bounded away from zero and $\lambda_{\max}(B_k)$ does not diverge to $+\infty$, $\cos \theta_k$ remains bounded away from zero and Theorem III.22 implies global convergence. This occurs if there exist $m, M > 0$ such that

$$0 < m \leq \lambda_{\min}(B_k) \leq \lambda_{\max}(B_k) \leq M < +\infty \quad \text{for all } k.$$

In this case, we say that the matrices B_k are *uniformly well conditioned*. Their condition numbers are uniformly bounded.

This is only a sufficient condition. If the matrices B_k are not uniformly well conditioned, global convergence *may* still occur. We may have to use other means to prove it.

Example 3.5.7. Observe the evolution of $\cos\theta_k$ in Table 3.2. It seems that asymptotically, its value converges to -1 and certainly stays bounded away from zero. In this example the condition number of $\nabla^2 f(x^*)$ is $\kappa_2^* \approx 2508.0096$ so that $1/\kappa_2^* \approx 0.0003987$. The bound (III.27) is thus fairly pessimistic. $\square$

Local convergence properties of quasi-Newton methods are rather general in the sense that fast asymptotic convergence is equivalent to a precise property of the matrices B_k. The quasi-Newton direction d_k^{QN} minimizes exactly the convex quadratic model in the variable d

$$m_k(d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d.$$

For this reason, it is important that any the initial steplength in any linesearch procedure based on Newton or quasi-Newton directions be equal to one.

Intuitively, it seems natural to think that fast local convergence should occur if the matrices B_k converge to $\nabla^2 f(x^*)$. It turns out that it holds under the weaker assumption that asymptotically, B_k converges to $\nabla^2 f(x^*)$ along the direction of the step. More precisely, this condition can be written

$$\lim_{k\to\infty} \frac{\|(B_k - \nabla^2 f(x^*)) d_k^{\text{QN}}\|}{\|d_k^{\text{QN}}\|} = 0. \tag{III.28}$$

This condition, known as the Dennis-Moré condition, is fundamental in quasi-Newton methods as it is both necessary and sufficient for fast local convergence.

In practice, (III.28) is difficult to verify and most algorithm build matrices B_k which are sufficiently close to $\nabla^2 f(x_k)$ for large k and whenever the latter Hessian matrix becomes positive definite.

Theorem 3.5.9. *Assume $f : \mathbb{R}^n \to \mathbb{R}$ is $\mathcal{C}^3$ and the sequence $\{x_k\}$ is generated from the iteration $x_{k+1} = x_k + t_k d_k^{QN}$ where the steplength t_k satisfies the Wolfe conditions with $0 < \alpha \leq 1/2$. Assume furthermore that $\{x_k\}$ converges to x^* where $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite. Then, if the Dennis-Moré condition (III.28) is satisfied,*

(a) there is a k_0 such that for all $k \geq k_0$, $t_k = 1$,

(b) $\{x_k\} \to x^*$ superlinearly.

Note that at variance with the steepest descent method and Newton's method, this result does not specify the precise rate of convergence. All it says is that

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0. \tag{III.29}$$

Under the assumptions of Theorem 3.5.9, if $t_k = 1$ for all sufficiently large k, it is possible to show that (III.28) is equivalent to (III.29).

A reason why this seems reasonable is that even though d_k^{QN} is not quite the Newton direction, it converges to the Newton direction fast enough. In fact, (III.28) can be rewritten

$$\lim_{k\to\infty} \frac{\|\nabla f(x_k) + \nabla^2 f(x_k) d_k^{\text{QN}}\|}{\|d_k^{\text{QN}}\|} = 0.$$

3.5.1.11 Limited-Memory Methods

[to come]

3.5.2 Trust Regions

3.5.2.1 A Basic Trust-Region Algorithm

Trust-region methods appear to date back to a 1944 paper in which they were used to solve nonlinear least-squares problems [Lev44]. For similar purposes, they were independently used by Morrison [Mor60] and Marquardt [Mar63]. Later, Goldfeldt, Quandt and Trotter [GQT66] introduced updating rules for the size of the region, a crucial step towards modern trust-region methods. In 1970, Powell [Pow70] proved global convergence of a particular trust-region algorithm. Different terminologies are used in the community, but substantial standardization appears as a result of the survey [Mor83]. Trust-region methods now form one of the most popular globalization schemes and are often praised for their robustness and flexibility. They are used throughout optimization, from regularization problems to derivative-free and interior-point methods. We trust that the reader's appetite for lists of references together with more historical notes and thorough theoretical developments across the whole optimization spectrum will be satisfied by the recent book [CGT00].

For simplicity and because our later numerical tests will only concern such problems, assume we wish to solve the unconstrained programming problem (III.1) where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice-continuously differentiable function. For problem (III.1) to be well defined, we assume throughout that f is bounded below. The philosophy of trust-region methods is to assume that f might be highly nonlinear and/or costly to evaluate. At iteration k, instead of manipulating f directly, f is replaced by a suitable local model m_k which is easier and cheaper to evaluate. However, from the very nature of a local model, it might not be wise to trust that m_k accurately represents f far from the current iterate x_k. A region $\mathcal{B}_k \subset \mathbb{R}^n$, referred to as the *trust region*, is therefore defined around x_k to represent the extent to which m_k is believed to reasonably model f. The trust region is defined as the ball

$$\mathcal{B}_k \equiv \{x_k + s \in \mathbb{R}^n \ : \ \|s\| \le \delta_k\},$$

where $\delta_k > 0$ is the current trust-region radius and $\|\cdot\|$ represents any norm on $\mathbb{R}^n$. To simplify the exposition, we choose the Euclidean norm but other choices are acceptable [CGT00].

Instead of applying a procedure to minimize f starting from x_k, the model m_k is approximately minimized within $\mathcal{B}_k$. If the decrease thus achieved is sufficient and if the agreement between f and m_k at the trial point is satisfactory, the step is accepted and the radius δ_k is possibly increased. Otherwise, the step is rejected and the radius is decreased. This last option indicates that m_k might have been trusted in too large a neighbourhood of x_k.

Global convergence of trust-region schemes is ensured by mild assumptions on m_k and on the decrease that should be achieved at each iteration. In practice, one of the most popular models is the quadratic model

$$m_k(x_k + s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T H_k s,$$

where $H_k = \nabla^2 f(x_k)$, or a symmetric approximation to it. For simplicity, we assume in the remainder of this paper that H_k is the exact Hessian of f at x_k.

Sufficient decrease in the model is established by considering the decrease obtained at two points of $\mathcal{B}_k$. The first is the *Cauchy point* x_k^{C}—the minimizer of m_k along the steepest descent direction $d = -\nabla m_k(x_k)$. The second is the minimizer of m_k along a direction of approximately minimal negative curvature, referred to as the *eigen point* and noted x_k^{E}. Sufficient decrease is achieved if the decrease in m_k is at least a fraction of that obtained at the best of these two points:

$$m_k(x_k) - m_k(x_k + s) \ge \theta \left[m_k(x_k) - \min \{ m_k(x_k^{\mathrm{C}}), m_k(x_k^{\mathrm{E}}) \} \right], \tag{III.30}$$

where $0 < \theta < 1$ is a fixed value, independent of the iteration number.

Algorithm 3.5.3: [Basic Trust-Region Algorithm]

Step 0. [Initialization] An initial point $x_0 \in \mathbb{R}^n$ and an initial trust-region radius $\delta_0 > 0$ are given, as well as parameters η_1, η_2, α_1 and α_2 satisfying

$$0 \leq \eta_1 < \eta_2 < 1 \quad \text{and} \quad 0 < \alpha_1 < 1 < \alpha_2.$$

Compute $f(x_0)$ and set $k = 0$.

Step 1. [Step calculation] Define a model $m_k(x_k + s)$ of $f(x_k + s)$ in $\mathcal{B}_k$ and compute a step $s_k \in \mathcal{B}_k$ which satisfies (III.30).

Step 2. [Acceptance of the trial point] Compute $f(x_k + s_k)$ and

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}. \tag{III.31}$$

If $\eta_1 \leq \rho_k$, then set $x_{k+1} = x_k + s_k$; otherwise, set $x_{k+1} = x_k$.

Step 3. [Trust-region radius update] Set

$$\delta_{k+1} = \begin{cases} \alpha_1 \|s_k\| & \text{if} \quad \rho_k < \eta_1 \\ \delta_k & \text{if} \quad \eta_1 \leq \rho_k < \eta_2 \\ \max[\alpha_2 \|s_k\|, \delta_k] & \text{if} \quad \eta_2 \leq \rho_k. \end{cases} \tag{III.32}$$

Increment k by one, and go to Step 1.

Combining all of the above, a typical trust-region framework for problem (III.1) may be stated as Algorithm 3.5.3.

We should alert the reader that the updating rule (III.32) at Step 3 of Algorithm 3.5.3 is not the only one used in practice, but most likely the most common one. Other rules involve polynomial interpolation of $\rho_k = \rho(s_k)$ as a function of the step s_k [DS96], while others devise more sophisticated functions to obtain the new radius [Hei00, WD04].

Requirements on the function f and each model m_k are gathered in Assumption 3.5.1.

Assumption 3.5.1. *The function f is bounded below and its Hessian matrix remains bounded over a set containing all iterates x_k. The model m_k coincides up to first order with f at x_k, i.e., $m_k(x_k) = f(x_k)$ and $\nabla m_k(x_k) = \nabla f(x_k)$.*

3.5.2.2 Convergence Properties of the Basic Algorithm

We recall in this section the important global convergence properties of Algorithm 3.5.3 without proof. The proofs may be found in [CGT00]. Quite remarkably, Assumption 3.5.1 is all that is required to prove global convergence of the basic trust-region framework. Note that no assumption is made on *how* the model should be minimized within $\mathcal{B}_k$ but rather, (III.30) imposes a condition on the *quality* of the resulting trial point.

Step 3 of Algorithm 3.5.3 is often referred to as the computation of achieved versus predicted reduction. Achieved reduction is the actual reduction in the objective f, defined by $\text{ared}_k = f(x_k) - f(x_k + s_k)$. Predicted reduction is the reduction suggested by the model, defined by $\text{pred}_k = m_k(x_k) - m_k(x_k + s_k)$. The quotient ρ_k is thus simply $\text{ared}_k/\text{pred}_k$. The step s_k will be accepted whenever $\text{ared}_k \geq \eta_1 \text{pred}_k$, an iteration we refer to as *successful*. If additionally, $\text{ared}_k \geq \eta_2 \text{pred}_k$, we shall say that the iteration is *very successful*. Otherwise, it is *unsuccessful*.

Critical to the global convergence is the fact that the difference between the objective and model values at the trial point decreases quadratically with δ_k, i.e., $|\text{ared}_k - \text{pred}_k| \leq \kappa\delta_k^2$, where $\kappa > 0$ is a constant, possibly dependent on k [CGT00, Theorem 6.4.1]. This fact ensures that after a finite number of unsuccessful steps, a successful step will be generated [CGT00, Theorem 6.4.2]. A first result considers the situation where there are only a finite number of successful iterations.

Theorem 3.5.10. *Suppose that Assumption 3.5.1 holds. If there are only finitely many successful iterations, then $x_k = x^*$ for all sufficiently large k where x^* is first-order critical for (III.1).*

The first stage in the global convergence analysis of Algorithm 3.5.3 is usually summarized by Theorem 3.5.11, which addresses the case where an infinite number of successful iterations occur.

Theorem 3.5.11. *Suppose that Assumption 3.5.1 is satisfied. Then*

$$\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0.$$

Theorem 3.5.11 was first proved by Powell [Pow70] in a framework where $\eta_1 = 0$, i.e., where all trial points are accepted as soon as they produce a decrease in the objective. This result proves that if $\{x_k\}$ has limit points, at least one of them is critical. In fact, this is as good a convergence result as we can obtain when $\eta_1 = 0$ [Yua98]. The framework of Algorithm 3.5.3 differs in that it is more demanding on the trial point—*sufficient* reduction must be achieved. This sheds some light on the importance of the value of η_1 in the framework, for as the next result shows, a much stronger conclusion holds in this case.

Theorem 3.5.12. *Suppose Assumption 3.5.1 is satisfied, and that $\eta_1 > 0$. Then*

$$\lim_{k\to\infty} \|\nabla f(x_k)\| = 0.$$

In other words, Theorem 3.5.12 shows that *all* limit points are first-order critical for (III.1). The distinction between Theorem 3.5.11 and Theorem 3.5.12 was reinforced by the careful example of [Yua98], where it is shown that an algorithm with $\eta_1 = 0$ may very well produce limit points which are not critical.

3.6 Nonlinear Least-Squares Problems

[To come]

3.6.1 Software

[To come]

Chapter 4

Solving Positive-Definite Systems

A fundamental subproblem in most algorithms for nonlinear optimization is to solve a linear system with a positive-definite coefficient matrix. In this chapter, we review the two major classes of methods for solving such systems. The first class concerns so-called direct *methods and we concentrate on the Cholesky factorization. The second class concerns* iterative *methods and we concentrate on the method of conjugate directions.*

4.1 Symmetric positive definite systems in nonlinear optimization

Each iteration of most linesearch algorithms requires that we solve a linear system of equations of the form

$$Hd = -g, \tag{IV.1}$$

where H is an $n \times n$ symmetric positive definite matrix. This type of subproblem occurs for instance when computing a search direction. In unconstrained programming, we then have $g = \nabla f(x)$ and for the direction d to be a descent direction, H must be positive definite. It may be a Hessian matrix or an appropriate approximation to a Hessian matrix. We will see later that this subproblem also occurs when we apply a *preconditioner*, i.e., modify a linear system $Ax = b$ into another which has a better eigenvalue distribution.

Two major options appear when the time comes to solve a system of linear equations of the form $Ax = b$. The first is to perform a factorization of the coefficient matrix, which consists in decomposing the matrix A into a product of two or three matrices $A = A_1A_2A_3$ in such a way that subsequently solving the system $A_1A_2A_3x = b$ is in a sense *easier* than solving the original system. Since we concentrate here on systems where the coefficient matrix is symmetric and positive definite, a specialized and very efficient factorization is available: the *Cholesky* factorization, which we review in §4.3. A factorization is often referred to as a *direct* method for solving linear systems. By opposition, the second option is to solve the system iteratively. Again, several possibilities are available among which one is particularly interesting from the point of view of optimization: the method of conjugate directions. We cover this iterative method in §4.4.

Before we examine these two classes of methods, it is important to review a few basic issues pertaining to sparse matrices and their storage.

4.2 A word on sparsity and storage

In optimization, exploiting the structure of the problem is always a good idea. For instance, if the objective function is a sum of squares, we know exactly how to take advantage of this knowledge. The same goes in linear algebra. To see this, denote by a_{ij} the element at the intersection of the i-th row and j-th column of the $n \times m$ coefficient matrix A. Storage of A requires us to store nm real values in an array `a[1 ...`

nm]. We could take the convention, for example, to store A by rows and would be able to access a_{ij} as the element in position $(i-1)n+j$ of the array a. When n or m becomes large, such storage requirements quickly becomes prohibitive.

If we know that A is symmetric, there is clearly no need to store all n^2 elements but it suffices to remember either of the upper or lower triangle, including the diagonal. This amounts to $n(n+1)/2$ elements. If we go further a suppose that A is, say, diagonal, it would be extremely wasteful to not take advantage of this and store explicitely the $n(n-1)/2$ zeros that are located below the diagonal. In this case, a vector of length n would be sufficient to store the values of A! Similarly, if A is *banded* with bandwidth $q \in \mathbb{N}$, $1 \leq q \leq n$, i.e., $a_{ij} = 0$ whenever $|i-j| \geq q$, we can save storage by remembering only the values of A in places where *nonzeros* may occur. Another special case which we will encouter often is that of *triangular* matrices. A is *lower triangular* if $a_{ij} = 0$ whenever $j > i$ and *upper triangular* if $a_{ij} = 0$ whenever $j < i$.

Example 4.2.1. Consider the objective function

$$f : \mathbb{R}^n \to \mathbb{R}, \quad f(x) = \sum_{i=1}^{n-1} (x_{i+1} - x_i^2)^2 + (1 - x_i)^2.$$

This function is such that $\partial^2 f(x)/\partial x_i \partial x_j = 0$ whenever $|i-j| \geq 3$. The matrix $\nabla^2 f(x)$ is thus symmetric tridiagonal for all x and $2n-1$ real values are sufficient to store it. $\square$

The structure of a matrix can be rich and come in different forms. A general useful notion is the *sparsity pattern* of A, i.e., the index set

$$\mathcal{S} = (\{1, \dots n\} \times \{1, \dots m\}) \setminus \{(i,j) \mid a_{ij} = 0\}.$$

We have indicated that $\mathcal{S}$ is the set of places in A where the element is possibly nonzero. It may also happen that elements in $\mathcal{S}$ take the value zero. It is just not *always* zero. The sparsity pattern is informative on the structure of a problem and may be used advantageously for instance during a factorization. We will call *dense* a matrix whose structure is unknown and/or such that $|S| \approx nm$. We will call *sparse* a matrix whose structure we decide to exploit and such that $|S| \ll nm$.

In practice, sparsity does not only affect storage. If the matrices arising in a problem are sufficiently sparse, it is also well worth using an algorithm that takes advantage of this and that avoids operating on the zero entries.

4.3 The Cholesky factorization

Recall that the $n \times n$ symmetric matrix A is positive definite if and only if $x^T A x > 0$ for all $x \in \mathbb{R}^n$, $x \neq 0$. Our first result gives a few properties of positive definite matrices and it is used to construct the Cholesky factorization.

4.3.1 Construction of the Cholesky factorization

Theorem 4.3.1. *The following properties are all equivalent to the symmetric matrix A being positive definite.*

(a) $M^T A M$ is positive definite for any nonsingular matrix M,

(b) any principal submatrix of A is positive definite,

(c) all eigenvalues of A are positive,

(d) $a_{ii} > 0$ *for all* $i = 1, \dots, n$ *and* $\max_{i,j} |a_{ij}| = \max_i a_{ii} > 0$*,*

The main result of this section leads to the definition of the Cholesky factorization. Its proof is interesting because it is constructive and gives an algorithm for computing the factorization.

Theorem 4.3.2. *The symmetric matrix A is positive definite if and only if there exists a unique lower triangular matrix L with positive diagonal entries such that $A = LL^T$.*

Proof. If $A = LL^T = LIL^T$ with L nonsingular, A must be positive definite by Theorem 4.3.1, part (a).

If A is positive definite, we construct L by induction on n. If $n = 1$, the result is true with $L = \sqrt{a_{11}}$ and $a_{11} > 0$ from Theorem 4.3.1. Assume the result is true if A has order $n-1$. If A has order n, we can write the block decomposition

$$A = \begin{bmatrix} a_{11} & v_1^T \\ v_1 & A_{22} \end{bmatrix},$$

where $a_{11} > 0$, $v_1 \in \mathbb{R}^{n-1}$ and A_{22} is symmetric and has order $n-1$. Define

$$M = \begin{bmatrix} a_{11}^{1/2} & 0 \\ a_{11}^{-1/2} v_1 & I \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & A_{22} - a_{11}^{-1} v_1 v_1^T \end{bmatrix}.$$

It is easy to check that $A = MBM^T$. Since M is nonsingular and A is positive definite, B must be positive definite, by part (a) of Theorem 4.3.1. From part (b) of Theorem 4.3.1, all principal submatrices of B are also positive definite and by induction, there thus exists a triangular matrix L_{22} with positive diagonal entries such that

$$A_{22} - a_{11}^{-1} v_1 v_1^T = L_{22} L_{22}^T \quad \text{and} \quad B = \begin{bmatrix} 1 & 0 \\ 0 & L_{22} L_{22}^T \end{bmatrix}.$$

Finally,

$$A = MBM^T = \begin{bmatrix} a_{11}^{1/2} & 0 \\ a_{11}^{-1/2} v_1 & L_{22} \end{bmatrix} \begin{bmatrix} a_{11}^{1/2} & a_{11}^{-1/2} v_1^T \\ 0 & L_{22}^T \end{bmatrix} = LL^T.$$

□

Definition 4.3.1. Given a symmetric positive definite matrix A, its *Cholesky factorization* is the unique decomposition $A = LL^T$ given by Theorem 4.3.2. In this factorization, the matrix L—called the *Cholesky factor of* A—is lower triangular with positive diagonal entries.

4.3.2 Computing the Cholesky factorization

To clarify the recursion in the proof of Theorem 4.3.2, it is enlightening to look at a generic 3×3 example.

Example 4.3.1. Let

$$A = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \qquad L = \begin{bmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ l_{31} & l_{32} & l_{33} \end{bmatrix}.$$

By direct multiplication, we have

$$LL^T = \begin{bmatrix} l_{11}^2 & l_{11} l_{21} & l_{11} l_{31} \\ l_{11} l_{21} & l_{21}^2 + l_{22}^2 & l_{21} l_{31} + l_{22} l_{32} \\ l_{11} l_{31} & l_{21} l_{31} + l_{32} l_{22} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{bmatrix}.$$

Equating A and LL^T, we first discover the first column of L

$$l_{11} = a_{11}^{1/2}, \quad l_{21} = a_{12}/l_{11}, \quad l_{31} = a_{13}/l_{11},$$

then the second column

$$l_{22} = (a_{22} - l_{21}^2)^{1/2}, \quad l_{32} = (a_{32} - l_{21}l_{31})/l_{22},$$

and finally

$$l_{33} = (a_{33} - l_{31}^2 - l_{32}^2)^{1/2}.$$

□

The proof of Theorem 4.3.2 and Example 4.3.1 suggest Algorithm 4.3.1 for computing L.

Algorithm 4.3.1: [LL^T Cholesky factorization]

```
For $j = 1, \ldots, n$,
    $l_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2\right)^{1/2}$
    For $i = j+1, \ldots, n$,
        $l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}\right) / l_{jj}$
    End
End
```

If the matrix A is not positive definite, Algorithm 4.3.1 will break down for some j trying to compute the square root of a negative number or because of a division by zero.

In practice, storage is saved by overwriting entries of the lower triangle of A with the entries of L. Only $n(n+1)/2$ entries are necessary for A and no additional storage is required for L. The total number of floating-point operations necessary to compute L is obtained by replacing loops by sums whose general term is the number of operations requires by each line of the loop. The first loop of Algorithm 4.3.1 will become a sum from 1 through n and the computation of each l_{jj} involves one addition, $(j-1)$ additions and squarings and one square root, for a total of $2j$ operations.

$$N_{\text{flops}} = \sum_{j=1}^{n} \left(2j + \sum_{i=j+i}^{n} 2j \right) = 2 \sum_{j=1}^{n} (j + j(n-j)).$$

Using the fact that $\sum_{i=1}^{n} i^2 = (2n^3 - 3n^2 + n)/6$, we get

$$N_{\text{flops}} = \frac{1}{3}(n^3 + 9n^2 + 2n) \approx \frac{n^3}{3}$$

by retaining only the dominant term.

Once the factor L is computed, solving the original system $Ax = b$ can be replaced by the problem of solving $LL^T x = b$. First, let $w = L^T x$ and solve the lower triangular system $Lw = b$ to obtain w. Then, solve $L^T x = w$ to obtain x. We see that we need to solve two triangular systems, which is an easy task. For instance, since L is lower triangular and $l_{11} > 0$, the first equation readily determines $w_1 = b_1/l_{11}$, the first component of w. Then, the second equation gives $w_2 = (b_2 - l_{12}w_1)/l_{22}$. Following this process, we determing each component of the solution w in turn, from the first to the last. Solving the second system is similar. This is often referred to as a *forward solve*. The last equation gives x_n and in turn, we determine all components of x from the last to the first. This is referred to as a *backsolve*.

The floating-point cost of solving a triangular system of order n is $\mathcal{O}(n^2)$. Therefore, the dominant cost in solving a symmetric positive-definite linear system is the factorization itself. Once it has been computed, it may be used to solve as many systems as necessary.

A variant of the so-called "LL^T" factorization is the "LDL^T" factorization of a symmetric positive-definite matrix, also referred to as the Cholesky factorization. The difference with Algorithm 4.3.1 is that the factor

L is required to be *unit* lower triangular, i.e., its diagonal entries are all equal to 1, and the matrix D is diagonal positive definite and its diagonal entries are the l_{ii}^2 of Algorithm 4.3.1. This variant does not require more storage but is slightly less costly in that it does not require computation of square roots. The diagonal entries of L, all equal to 1, are not stored and a vector of length n suffices to store D. Proceeding as before, the corresponding variant of Algorithm 4.3.1 may be stated as Algorithm 4.3.2.

Algorithm 4.3.2: [LDL^T **Cholesky factorization**]

```
For $j = 1, \ldots, n$,
    $d_{jj} = a_{jj} - \sum_{k=1}^{j-1} d_{kk} l_{jk}^2$
    For $i = j + 1, \ldots, n$,
        $l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} d_{kk} l_{ik} l_{jk}\right) / d_{jj}$
    End
End
```

Evidently, Algorithm 4.3.2 introduces no further complication when solving the factorized system $LDL^T x = b$ since solving a system with a nonsingular diagonal coefficient matrix is trivial.

4.3.3 A word on ordering

Manipulating sparse matrices and in particular factorizing them poses a few problems that do not arise when dealing with dense matrices. One such problem is referred to as *fill-in* and is illustrated in Example 4.3.2.

Example 4.3.2. Let the $n \times n$ symmetric positive-definite matrix A be defined as $a_{ii} = 1$ for $i = 1, \ldots, n$ and $a_{1j} = 0.1$ for $j = 2, \ldots n$. The sparsity pattern of A and of its Cholesky factor L is illustrated in Fig. IV.1. This matrix A is often referred to as the *arrow* matrix. $\square$

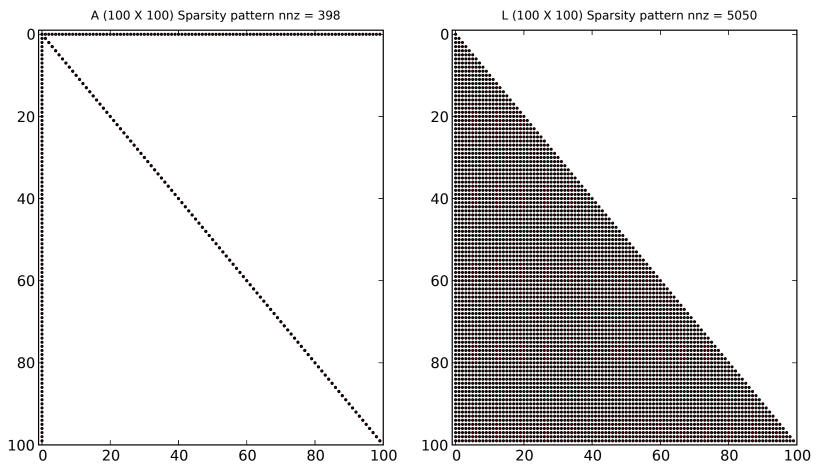


Figure IV.1: Fill-in occurs in the Cholesky factor L of the arrow matrix A.

It turns out that while A was extremely sparse, L is dense! The process consisting of having entries of L that are not part of the sparsity pattern of A taking nonzero values is referred to as *fill-in*. Consider now the very similar matrix of Example 4.3.3.

Example 4.3.3. Let the $n \times n$ symmetric positive-definite matrix A be defined as $a_{ii} = 1$ for $i = 1, \ldots, n$ and $a_{jn} = 0.1$ for $j = 1, \ldots n-1$. The sparsity pattern of A and of its Cholesky factor L is illustrated in Fig. IV.2. This matrix A is simply a permutation of the matrix of Example 4.3.2. □

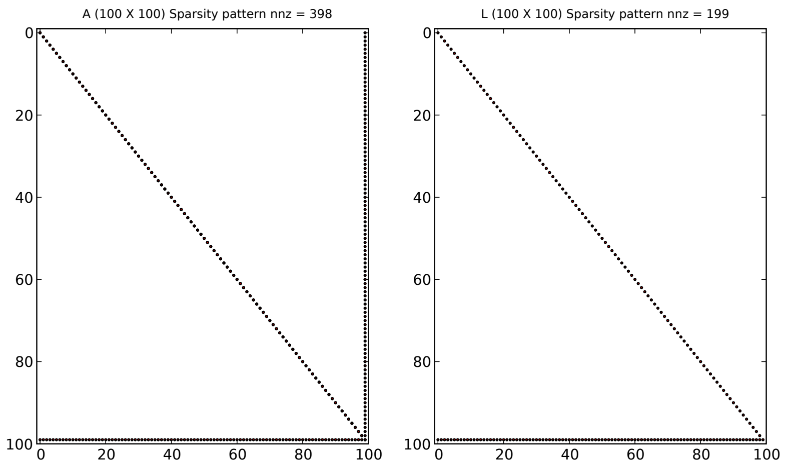


Figure IV.2: No fill-in occurs in the Cholesky factor L of the permuted arrow matrix A.

A system involving the coefficient matrix of Example 4.3.3 is nothing else that the system of Example 4.3.2 where the equations and unknowns were renumbered. Such a renumbering is called a *symmetric permutation* and we see that an inappropriate ordering can induce extreme fill-in, with consequences in storage and solution time.

Definition 4.3.2. A *permutation matrix* of order n is an $n \times n$ nonsingular matrix whose rows are a permutation of the rows of the identity matrix. A *symmetric permutation* of the $n \times n$ matrix A is a matrix of the form PAP^T where P is a permutation matrix of order n.

Finding an ordering is finding a permutation matrix P. Finding an ordering so as to minimize the fill-in in the Cholesky factor L of A is an NP-hard problem, which means that it is more difficult than solving the linear system $Ax = b$. Fortunately, we know of a few heuristic orderings which perform well in practice. We briefly review a few of them in the next section.

4.3.4 A few popular orderings

(a) minimum degree/approximated minimum degree (AMD) ordering

(b) Cuthill-McKee ordering

(c) reverse Cuthill-McKee ordering

(d) Gibbs-Poole-Stockmeyer ordering

4.3.5 Software

4.4 The method of conjugate directions

While a factorization is an intuivite, convenient and numerically accurate means to solve a linear system of equations, it poses a serious problem when the order of the system becomes large. This problem is *storage*. Reordering algorithms are often little more than heuristics and cannot be expected to work well in all cases. In Table 4.1 a few matrices are examined and for each matrix, we report the number of nonzero elements in its lower triangle and the number of nonzero elements in its Cholesky factor. The matrices come from the MatrixMarket matrix collection.

Name	n	nnz(A)	nnz(L)	density(A)	density(L)
1138bus	1138	2596	38312	0.400561	5.91152
bcsstk08	1074	7017	234160	1.21554	40.563
bcsstk09	1083	9760	63702	1.66273	10.8524
bcsstk10	1086	11578	20771	1.96157	3.51907
bcsstk11	1473	17857	77270	1.6449	7.11772
bcsstk18	11948	80519	2684634	0.112798	3.76087
bcsstk19	817	3835	39433	1.14768	11.8009

Table 4.1:

In this section, we look at an alternative method for the solution of a symmetric positive definite linear system such as (IV.1). At variance with factorizations, the method of this section is iterative. First note that if the matrix H is positive definite, by the first-order necessary and second-order sufficient optimality conditions, the system (IV.1) is *equivalent* to solving the convex quadratic problem

$$\text{minimize } \phi(x), \qquad \phi(x) = g^T x + \frac{1}{2} x^T H x. \tag{IV.2}$$

The method of conjugate directions is an iterative procedure to solve (IV.1) or, equivalently, (IV.2). Because of its properties in relation to $\phi(\cdot)$ and its excellent numerical behaviour, it is a popular method in nonlinear optimization.

Consider the convex quadratic in two variables $\phi(x)$ where

$$g = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \qquad H = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}.$$

The contours of this function are show in the left part of Fig. IV.3 and its unique global minimizer, the unique solution to (IV.1), is $x^* = (2, 1)$. We can reach this global minimizer by following each coordinate direction in turn, stopping as soon as we have reached the minimizer along that direction. Starting from $x_0 = (-1, -3)$, we first move to $x_1 = (2, -3)$ and then to $x_2 = x^* = (2, 1)$. Two steps are sufficient in this case. A single step would have been sufficient had x_0 been aligned with x^* along one of the axes. Consider now a second example where

$$g = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \qquad H = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}.$$

The contours of this second function appear in the right part of Fig. IV.3 and its global minimizer is $x^* = (1, 1)$. Following the same strategy as previously and moving along the coordinate directions no longer leads to x^* in two steps. However, if instead of the coordinate directions we move along the axes of the ellipses, we recover the two steps convergence.

In the two examples of Fig. IV.3, we found a procedure to reach the global minimizer of $\phi(x)$ in at most n steps, provided the search directions are chosen carefully. In the case of the axes of the ellipses, a set

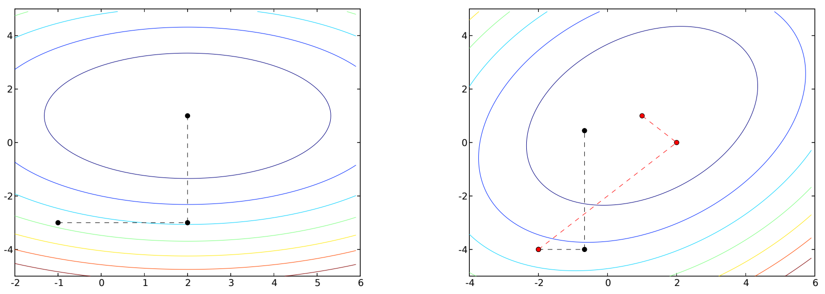


Figure IV.3: Minimizing a convex quadratic in at most n steps. On the left, we reach the global minimizer.

of such directions are the eigenvectors of H. In the second example, these eigenvectors are $v_1 = (1, 1)$ and $v_2 = (-1, 1)$. A property of eigenvectors v_i and v_j of H that are associated to distinct eigenvalues is that they are orthogonal. This implies that $v_i^T H v_j = 0$ so that they are also orthogonal in the inner product defined by H. This property is called *conjugacy* and is not only satisfied by eigenvectors.

Definition 4.4.1. Let $H \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix. The k nonzero vectors $p_1, \ldots, p_k$ in $\mathbb{R}^n$ are said to be *conjugate* (with respect to H) if and only if $p_i^T H p_j = 0$ whenever $i \neq j$.

A simple consequence of conjugacy is linear independence. Therefore if we have a set of n conjugate vectors in $\mathbb{R}^n$, they form a basis.

Exercise 4.4.1. [HJ85, p. 400] Let $H \in \mathbb{R}^{n\times n}$ be positive semidefinite and $x \in \mathbb{R}^n$. Show that $x^T Hx = 0$ if and only if $Hx = 0$. Conclude that a positive semidefinite matrix $H \in \mathbb{R}^{n\times n}$ has rank n if and only if it is positive definite. *Hint: Consider the quadratic $p(t) = (x + td)^T H(x + td)$ for $t \in \mathbb{R}$. If $x^T Hx = 0$, show that $p(t) \geq 0$ for all t, that $p(0) = 0$ and $p'(0) = 0$. Conclude that $d^T Hx = 0$ for all $d \in \mathbb{R}^n$ and hence that $Hx = 0$.*

Exercise 4.4.2. [HJ85, p. 400] Show that if a positive semidefinite matrix has a zero entry on the main diagonal, then the entire row and column to which it belongs must be zero.

Exercise 4.4.3. [HJ85, p. 400] Show that if the main diagonal entries of a positive definite matrix are all $+1$, then all entries of the matrix are bounded by 1 in absolute value. Can equality occur?

Exercise 4.4.4. [HJ85, p. 400] Show that a positive semidefinite matrix matrix $H \in \mathbb{R}^{n\times n}$ is of rank 1 if an only if $H = xx^T$ for some $x \in \mathbb{R}^n$.

Chapter 5

Constrained Optimization

We turn our attention in this chapter to problems for which the decision variables are constrained to satisfy prescribed conditions. Mathematically, this can be represented in two ways, each having its advantages and disadvantages. In the first, the decision variables $x \in \mathbb{R}^n$ are constrained to remain in an abstract set, called the *feasible set*, $\Omega \subseteq \mathbb{R}^n$. The problem is then written as

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & x \in \Omega, \end{array} \tag{V.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function. In the present text, we only consider problems with continuous data and in particular, a continuous objective function. Because we know that continuous functions attain their extrema over closed bounded sets, we only consider problems for which the feasible set Ω is closed. Whether the objective attains a minimum over Ω when Ω is unbounded is a difficulty algorithms will need to sort out.

In the second representation, the set Ω is assumed to be described by a set of equalities and/or inequalities

$$\Omega = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} h_i(x) = 0, & i = 1, \ldots, m, \\ c_j(x) \geq 0, & j = 1, \ldots, p \end{array} \right\}, \tag{V.2}$$

where the functions $h_i, c_j : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, $j = 1, \ldots, p$, are the *constraint functions* or simply the *constraints*. We distinguish between equality and inequality constraints as they are convenient for modeling real problems and as algorithms usually treat them differently. With this form for the feasible set, the problem is written as

$$\begin{array}{lll} \underset{x}{\text{minimize}} & f(x) & \\ \text{subject to} & h_i(x) = 0, & i = 1, \ldots, m, \\ & c_j(x) \geq 0, & j = 1, \ldots, p. \end{array} \tag{V.3}$$

We will similarly assume that the constraint functions are continuous. In this second formulation, they describe a closed feasible set.

Working with the abstract formulation (V.1) has the advantage of generality while designing algorithms usually requires working with the explicit formulation (V.3).

Solutions for the constrained problems come, as for the unconstrained problem, in different sorts.

Definition 5.0.2. Consider the constrained problem (V.1) and let $x^* \in \Omega$. We say that x^* is

(a) a *global constrained minimizer* of f if and only if for all $x \in \Omega$, $f(x) \geq f(x^*)$,

(b) a *local constrained minimizer* of f if and only if there exists $\epsilon > 0$ such that for all $x \in B(x^*, \epsilon) \cap \Omega$, $f(x) \geq f(x^*)$,

(c) a *strict local constrained minimizer* of f if and only if there exists $\epsilon > 0$ such that for all $x \in B(x^*, \epsilon) \cap \Omega$, $x \neq x^*$, $f(x) > f(x^*)$,

(d) an *isolated local constrained minimizer* of f if and only if there exists $\epsilon > 0$ such that x^* is the only local unconstrained minimizer of f in $B(x^*, \epsilon) \cap \Omega$.

Example 5.0.1. Consider the equality-constrained problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & x^2 \\ \text{subject to} & x^6 \sin(\pi/x) = 0. \end{array}$$

The feasible set is given by $\Omega = \{0\} \cup \{\frac{1}{k} \mid k \in \mathbb{Z}_0\}$. Around every feasible point except 0, it is possible to define an open ball that does not contain any other feasible point. Therefore, there exist no feasible directions from those points, let alone feasible descent directions. Every such point is thus a local minimizer. Moreover, $x^* = 0$ is the only global minimizer. It is not isolated because for any $\epsilon > 0$, the feasible points $\pm\frac{1}{k}$ with $k > \lceil \frac{1}{\epsilon} \rceil$ fall in the open ball $]-\epsilon, \epsilon[$ centered at $x^* = 0$. □

In the case where $\Omega = \emptyset$, the problem is termed *infeasible* and by convention, the optimal value is $+\infty$.

Depending on the form of the objective function and of the feasible set, the constrained problem is cast into a named category. A few of those categories follow.

(a) If f is linear and all h_i and c_j are affine functions, the problem is a *linear program*,

(b) if f is convex, the functions h_i are affine and the functions c_j are concave, the problem is a *convex program*,

(c) if f is quadratic and all h_i and c_j are affine functions, the problem is a *quadratic program*,

(d) if f, the h_i and c_j are all polynomial, the problem is a *polynomial program*,

(e) if either the objective or the constraint functions are more general, the problem is a *general nonlinear program.*

A major element of this chapter is a set of first-order necessary conditions for optimality. Three main ingredients will be used to obtain them: the *tangent cone*, the *normal cone* and *constraint qualifications.* Each of these is introduced in turn below and by combining them, we obtain the Karush-Kuhn-Tucker optimality conditions.

5.1 Geometry of the Feasible Set

5.1.1 Tangent and normal cones

The major difficulty with the constrained problem is that a local solution x^* must satisfy $x^* \in \Omega$. If an algorithm is to follow a number of directions to identify x^*, some of these directions will have to be such that either the iterates never leave the feasible set, or progress is made towards the feasible set. We first introduce the concept of feasible direction. Before we start, we note that we assumed Ω to be closed. The following definitions are easily generalized to non-closed sets simply by considering $\text{cl}(\Omega)$ instead of Ω.

Definition 5.1.1. Let $x \in \Omega$ and $d \in \mathbb{R}^n$, $d \neq 0$. The direction d is said to be a *feasible direction* from x if there is $\epsilon > 0$ such that for all $t \in]0, \epsilon[$, $x + td \in \Omega$.

In other words, a feasible direction is one which leads towards the feasible set. If $x \in \Omega$, the set of all feasible directions from x forms a *cone* in $\mathbb{R}^n$, i.e., a set $K \subseteq \mathbb{R}^n$ such that for all $d \in K$ and all $\lambda \geq 0$, $\lambda d \in K$. This cone is not necessarily closed but its closure plays an important role in the quest for optimality conditions.

Definition 5.1.2. Let $x \in \Omega$. A vector $v \in \mathbb{R}^n$ is said to be *tangent* to Ω at x if there exists a sequence $\{v_k\}$ in $\mathbb{R}^n$ such that $v_k \to v$, a sequence $\{t_k\}$ of positive scalars such that $t_k \to 0$, and if these sequences are such that for all k, $x + t_k v_k \in \Omega$.

Note that for any $x \in \Omega$, the vector $v = 0$ is a tangent vector to Ω at x. Indeed, it suffices to choose $v_k = 0$ for all k and any sequence $\{t_k\}$ converging to zero—for instance $t_k = 1/k$.

If v is a tangent vector to Ω at x, so is λv for any $\lambda > 0$. To verify this claim, we simply replace v_k with λv_k and t_k with t_k/λ. This fact leads to the following definition.

Definition 5.1.3. Let $x \in \Omega$. The set $\mathcal{T}_\Omega(x)$ of all tangent vectors to Ω at x forms a cone called the *tangent cone* to Ω at x or, for short, the *tangent cone.*

A direct property of the tangent cone is that $0 \in \mathcal{T}_\Omega(x)$ for any $x \in \Omega$. The tangent cone of Definition 5.1.3 is sometimes called the *contingent cone* or the *sequential Bouligand tangent cone.* An equivalent definition of $\mathcal{T}_\Omega(x)$ is given in the next result.

Theorem 5.1.1. *Let $x \in \Omega$ and $v \in \mathbb{R}^n$. Then $v \in \mathcal{T}_\Omega(x)$ if and only if there exists a sequence $\{x_k\}$ of elements of Ω converging to x and a sequence $\{\lambda_k\}$ of positive scalars such that $\lim_{k\to\infty} \lambda_k(x_k - x) = v$.*

Proof. If $v \in \mathcal{T}_\Omega(x)$, choose $x_k = x + t_k v_k$. This sequence converges to x and for all k, $x_k \in \Omega$. Next, choose $\lambda_k = 1/t_k$. We have $\lambda_k > 0$ for all k and $\lambda_k(x_k - x) = v_k$, which converges to v.

Conversely, we can assume $v \neq 0$. The sequences $v_k = \lambda_k(x_k - x)$ and $t_k = 1/\lambda_k$ verify Definition 5.1.2. □

The cone $\mathcal{T}_\Omega(x)$ is always a closed cone but need not be convex.

The second important concept that will lead to first-order necessary conditions is the following.

Definition 5.1.4. Let $x \in \Omega$. A vector $w \in \mathbb{R}^n$ is *normal* to Ω at x if $w^T v \leq 0$ for all $v \in \mathcal{T}_\Omega(x)$. The set of all normal vectors to Ω at x forms a cone called the *normal cone* to Ω at x or simply the *normal cone* and denoted $\mathcal{N}_\Omega(x)$.

The normal cone is always nonempty since $0 \in \mathcal{N}_\Omega(x)$. At variance with the tangent cone, $\mathcal{N}_\Omega(x)$ is always closed and convex, as is easily seen from the definition.

Below are some examples and Fig. V.1 illustrates different geometrical situations.

Example 5.1.1. If $x \in \text{int}(\Omega)$, any vector is tangent. Therefore $\mathcal{T}_\Omega(x) = \mathbb{R}^n$. The normal cone is $\mathcal{N}_\Omega(x) = \{0\}$. □

Example 5.1.2. If $\Omega = \text{cl}(B(x_0; r))$, the closed ball centered at some $x_0 \in \mathbb{R}^n$ and with radius $r > 0$, and if x lies on the boundary of Ω, $\mathcal{T}_\Omega(x)$ is a closed half space. The normal cone is the half straight line orthogonal to this half space. □

Example 5.1.3. If Ω is the positive orthant, $\mathcal{T}_\Omega(0)$ coincides with Ω and $\mathcal{N}_\Omega(0)$ is the negative orthant. □

Example 5.1.4. If $\Omega = \{x \in \mathbb{R}^n \mid Ax = b\}$ is an affine subspace of $\mathbb{R}^m$, where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, and if $x \in \Omega$, $\mathcal{T}_\Omega(x) = \{x \in \mathbb{R}^n \mid Ax = 0\}$ is the subspace parallel to Ω and passing through the origin. In other words, it is the nullspace of A. The normal cone is the subspace orthogonal to $\mathcal{T}_\Omega(x)$, i.e., the range space of A^T. □

Example 5.1.5. If $\Omega = \{x_0\}$, $\mathcal{T}_\Omega(x_0) = \{0\}$ and $\mathcal{N}_\Omega(x_0) = \mathbb{R}^n$. □

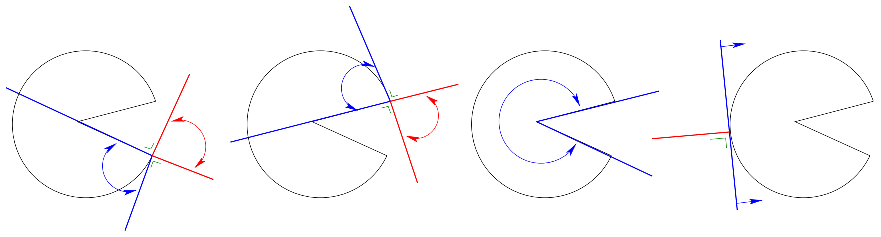


Figure V.1: The tangent cone (in blue) and the normal cone (in red) at four points of the Pacman set. Green wedges indicate right angles. Note that in the third case, the tangent cone is nonconvex and the normal cone is reduced to $\{0\}$. In the fourth case, the normal cone is a half straight line. The point of each cone should be at the origin; the cones were translated for clarity of the picture.

5.1.2 Tangent cone to the feasible set

When the feasible set Ω is described in the explicit notation (V.2) it is possible to obtain an analytical description of the tangent and normal cones to Ω.

Consider the case where all the function h_i and c_j are affine, i.e.,

$$h_i(x) = a_i^T x - b_i \quad (i = 1, \dots, m), \tag{V.4a}$$
$$c_j(x) = r_j^T x - q_j \quad (j = 1, \dots, p), \tag{V.4b}$$

where for all i, j, a_i, $r_j \in \mathbb{R}^n$ and b_i, $q_j \in \mathbb{R}$. The feasible set is in this case a polyhedron, such as the polyhedron of Fig. V.2. As we can see, the tangent cone to this polyhedron at some point $x \in \Omega$ is also a polyhedron, entirely specified by the equality constraints and those inequality constraints that are verified as equations at x. Such constraints are termed *active*, as in Definition 5.1.5.

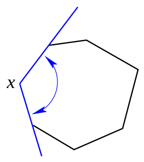


Figure V.2: With constraints of the form (V.4), the feasible set is a polyhedron. The tangent cone to Ω at x is then defined by the active constraints. For clarity of the picture, $\mathcal{T}_\Omega(x)$ was translated so its point coincides with x.

Definition 5.1.5. Let $x \in \Omega$ where Ω is represented by (V.2). The index set $\mathcal{A}(x) = \{1, \dots, m\} \cup \{j = 1, \dots, p \mid c_j(x) = 0\}$ is called the *set of active constraints* at x.

The next result specifies the exact form of $\mathcal{T}_\Omega(x)$ in this case.

Theorem 5.1.2. *Let Ω be given in explicit form (V.2) and let the constraints be affine and be given by (V.4). Then for any $x \in \Omega$ the tangent cone to Ω at x is given by*

$$\mathcal{T}_\Omega(x) = \left\{ v \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} a_i^T v = 0 & i = 1, \dots m, \\ r_j^T v \geq 0 & j \in \mathcal{A}(x) \end{array} \right\}.$$

As is apparent from Theorem 5.1.2, inactive constraints at x are irrelevant to the tangent cone.

The question naturally arises as to whether Theorem 5.1.2 still holds in the case of nonlinear constraints. Unfortunately, in the more general case, the inequality fails to hold and we only have an inclusion, as specified by Theorem 5.1.3.

Theorem 5.1.3. *Let Ω be given in explicit form (V.2). Then for any $x \in \Omega$,*

$$\mathcal{T}_\Omega(x) \subseteq \left\{ v \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} \nabla h_i(x)^T v = 0 & i = 1, \dots m, \\ \nabla c_i(x)^T v \geq 0 & j \in \mathcal{A}(x) \end{array} \right\}. \tag{V.5}$$

Example 5.1.6. In $\mathbb{R}^2$, consider the feasible set

$$\Omega = \left\{ (x_1, x_2) \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} x_1 \geq 0, \\ x_2 \geq 0, \\ (1 - x_1)^3 - x_2 \geq 0 \end{array} \right\},$$

and $x = (1, 0)$. Then $\mathcal{A}(x) = \{2, 3\}$ and it is easy to see that $\mathcal{T}_\Omega(x) = \mathbb{R}_- \times \{0\}$ while the set in the right-hand side of (V.5) is $\mathbb{R} \times \{0\}$. This example is illustrated in Fig. V.3. □

Example 5.1.7. In $\mathbb{R}^2$, consider the feasible set

$$\Omega = \left\{ (x_1, x_2) \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} x_2 \geq -x_1^2, \\ (x_1^2 - x_2)^3 \geq 0 \end{array} \right\},$$

and $x = (0, 0)$. Then both constraints are active at x and $\mathcal{T}_\Omega(x) = \mathbb{R} \times \{0\}$ while the set in the right-hand side of (V.5) is $\mathbb{R} \times \mathbb{R}_+$. To make this example even more dramatic, consider representing the *same* feasible set as

$$\Omega = \left\{ (x_1, x_2) \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} (x_1^2 + x_2)^3 \geq 0, \\ (x_1^2 - x_2)^3 \geq 0 \end{array} \right\}.$$

At $x = (0, 0)$ the set in the right-hand side of (V.5) is now $\mathbb{R}^2$! Note however that if Ω were represented as

$$\Omega = \left\{ (x_1, x_2) \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} x_1^2 + x_2 \geq 0, \\ x_1^2 - x_2 \geq 0 \end{array} \right\}, \tag{V.6}$$

the equality would hold in (V.5) at $x = (0, 0)$. This example is illustrated in Fig. V.3. □

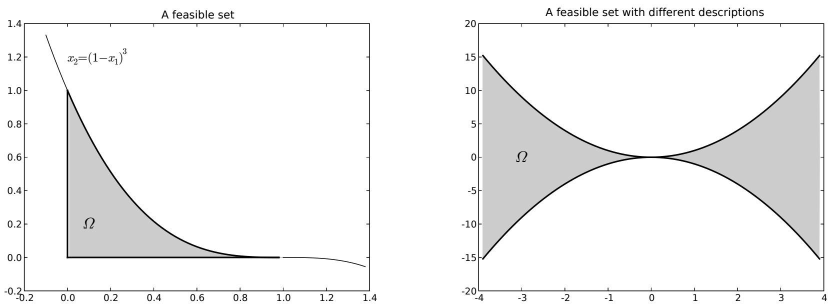


Figure V.3: On the left, the feasible set of Example 5.1.6. Note that the curve $x_2 = (1 - x_1)^3$ reaches the horizontal axis at $(1, 0)$ tangentially. On the right, the feasible set corresponding to the three descriptions of Example 5.1.7. Note that the two curves $x_2 = x_1^2$ and $x_2 = -x_1^2$ are tangent at $(0, 0)$.

5.1.3 Normal cone to the feasible set

In the case of affine constraints (V.4), there is similarly an analytical expression for the normal cone.

Theorem 5.1.4. *Let Ω be given in explicit form (V.2) and let the constraints be affine and be given by (V.4). Then for any $x \in \Omega$ the normal cone to Ω at x is given by*

$$\mathcal{N}_\Omega(x) = \left\{ \sum_{i=1}^{m} \mu_i a_i - \sum_{j \in \mathcal{A}(x)} \lambda_j r_j \;\middle|\; \begin{array}{ll} \mu_i \in \mathbb{R} & i = 1, \dots m, \\ \lambda_j \geq 0 & j \in \mathcal{A}(x) \end{array} \right\}.$$

As before, this expression generalizes to the convex case but not to the general nonlinear case, where only one inclusion holds.

Theorem 5.1.5. *Let Ω be given in explicit form (V.2). Then for any $x \in \Omega$,*

$$\mathcal{N}_\Omega(x) \supseteq \left\{ \sum_{i=1}^m \mu_i \nabla h_i(x) - \sum_{j \in \mathcal{A}(x)} \lambda_j \nabla c_j(x) \;\middle|\; \begin{array}{ll} \mu_i \in \mathbb{R} & i = 1, \dots m, \\ \lambda_j \geq 0 & j \in \mathcal{A}(x) \end{array} \right\}. \tag{V.7}$$

Example 5.1.8. With the feasible set of Example 5.1.6, the normal cone to Ω at $(1, 0)$ is $\mathbb{R} \times \mathbb{R}_+$ while the set on the right-hand side of (V.7) is $\{0\} \times \mathbb{R}$. □

Clearly, an element is missing to be able to completely describe tangent and normal cones analytically. Exploration of this missing element is the subject of the next section.

5.1.4 Constraint qualification conditions

When non-specialists apply nonlinear optimization algorithms, they are often confronted with unnatural and counter-intuitive assumptions that seem unrelated to the problem, only to its formulation. This section attempts to explain the reason for such assumptions and how they help solve the problem.

Definition 5.1.6. Any assumption on the constrained problem (V.3) allowing the equality to hold in (V.5) is called a *constraint qualification condition* or, for short, a *constraint qualification*.

Why do we need constraint qualification conditions? The reason was already hinted at in Example 5.1.7. Remember the tentative analytical description (V.5) and suppose that the constraint $c_i(x) \geq 0$ is active at $x^* \in \Omega$ and explicitly appears in the right-hand side. Now consider an alternative description of Ω where, instead of imposing the condition $c_i(x) \geq 0$, we impose $c_i(x)^3 \geq 0$. Clearly, this leaves Ω unchanged and this constraint is still active at x^* only this time its gradient at x^* vanishes and therefore, does not appear in the analytical description any longer. Consequently, the set on the right-hand side of (V.5) is enlarged.

The reason why inequality does not hold in (V.5) is that the right-hand side depends explicitly on the algebraic form of the constraints. A constraint qualification condition is generally added to suppress such forms of "degeneracy". As Theorem 5.1.2 suggests, affine constraints do not suffer from multiple algebraic representations and no constraint qualification condition is required in their case.

In order to begin describing such conditions, we give a base case in the following definition.

Definition 5.1.7. Let $x \in \Omega$ and define the set

$$\Lambda(x) = \{(\mu, \lambda) \in \mathbb{R}^m \times \mathbb{R}^p \mid \lambda_j \geq 0 \; \forall j \in \mathcal{A}(x) \quad \text{and} \quad \lambda_j = 0 \; \forall j \notin \mathcal{A}(x)\} \tag{V.8}$$

We say that the *basic constraint qualification condition* is satisfied at x if and only if the only $(\mu, \lambda) \in \Lambda(x)$ such that

$$\sum_{i=1}^m \mu_i \nabla h_i(x) - \sum_{j \in \mathcal{A}(x)} \lambda_j \nabla c_j(x) = 0$$

is $(\mu, \lambda) = (0, 0)$.

Example 5.1.9. In the case of Example 5.1.7 and the description (V.6), both constraints are active at $(0, 0)$ and their gradients are opposites of each other. Therefore, $(0, 0)$ is expressed by any couple $(\lambda, -\lambda) \in \Lambda(0, 0)$ with $\lambda \geq 0$. The basic constraint qualification condition is thus not satisfied. □

The relevance of the basic constraint qualification condition is specified in the next result.

Theorem 5.1.6. *Let $x \in \Omega$ where Ω is given explicitly by (V.2). If the basic constraint qualification condition is satisfied at x, equality holds in (V.5) and in (V.7).*

The next result summarizes the very reason why we introduced the notion of tangent cone.

Theorem 5.1.7. *Let $x \in \Omega$ where Ω is given explicitly by (V.2). If the basic constraint qualification condition is satisfied at x, then the tangent cone to Ω at x coincides with the closure of the set of all feasible directions from x, i.e.,*

$$\mathcal{T}_\Omega(x) = \{0\} \cup cl\,\{d \mid d \text{ is a feasible direction from } x\}. \tag{V.9}$$

Example 5.1.10. To see why the basic constraint qualification condition is required in Theorem 5.1.7, consider the feasible set $\Omega \subset \mathbb{R}$ described by the two constraints $h(x) = 0$ and $c(x) \geq 0$ where $h(x) = x^6 \sin(\pi/x)$ and $c(x) = x^3$. We have $\Omega = \{0\} \cup \{1/k \mid k \in \mathbb{N}_0\}$ so that there exists no feasible direction from 0. The right-hand side of (V.9) at 0 is thus $\{0\}$. However, $v = 1$ is a tangent vector to Ω at 0 since in Definition 5.1.2 we can choose $v_k = 1$ for all k and $t_k = 1/k$. □

Therefore, under the basic constraint qualification condition, we have an analytical description of both the tangent and the normal cone to the feasible set at a given feasible point. There remains one difficulty. It is not clear how to verify that the basic constraint qualification condition is satisfied in practice. To remedy this difficulty, different conditions are used which are stronger but easier to verify.

Definition 5.1.8 (SCQ)**.** Assume the constraint functions h_i are all affine and the constraint functions c_j are all concave. We say that the *Slater constraint qualification condition* is satisfied if there exists $\hat{x} \in \mathbb{R}^n$ such that $h_i(\hat{x}) = 0$ for all $i = 1, \ldots, m$ and $c_j(\hat{x}) > 0$ for all $j = 1, \ldots p$. Such an $\hat{x}$ is called *strictly feasible*.

Definition 5.1.9 (MFCQ)**.** The *Mangasarian and Fromowitz constraint qualification condition* is satisfied at $x \in \Omega$ if the gradients $\nabla h_i(x)$ $(i = 1, \ldots, m)$ are linearly independent in $\mathbb{R}^n$ and if there exists a direction $d \in \mathbb{R}^n$ such that $\nabla h_i(x)^T d = 0$ for all $i = 1, \ldots, m$ and $\nabla c_j(x)^T d > 0$ for all $j \in \mathcal{A}(x)$.

Definition 5.1.10 (LICQ)**.** The *Linear Independence Constraint Qualification condition* is satisfied at $x \in \Omega$ if

$$\{\nabla h_i(x) \mid i = 1, \ldots m\} \cup \{\nabla c_j(x) \mid j \in \mathcal{A}(x)\}$$

is a set of linearly independent vectors in $\mathbb{R}^n$.

Note that the Slater condition is global in the sense that it concerns the feasible set as a whole while the other two are to be checked at a given feasible point.

It is possible to show that satisfaction of either of these conditions implies satisfation of the basic constraint qualification condition. In fact, the MFCQ can be shown to be *equivalent* to the basic constraint qualification. The LICQ is stronger than the MFCQ and when the constraint functions h_i are all affine and the constraint functions c_j are all concave, the Slater condition is equivalent to the MFCQ.

5.2 First-Order Optimality Conditions for Constrained Optimization

Intuitively, a feasible point x is a local minimizer if it is impossible to find a feasible descent direction. Given our exploration of tangent and normal cones in §5.1, the following result confirms geometrical intuition.

Theorem 5.2.1. *Let $f \in \mathcal{C}^1(\Omega)$ in problem (V.1). If $x^* \in \Omega$ is a local constrained minimizer of f, then $-\nabla f(x^*) \in \mathcal{N}_\Omega(x^*)$.*

Proof. If x^* is a local constrained minimizer, there is $\epsilon > 0$ such that for all $x \in B(x^*; \epsilon) \cap \Omega$, $f(x) \geq f(x^*)$. To prove $-\nabla f(x^*) \in \mathcal{N}_\Omega(x^*)$, we choose an arbitrary $v \in \mathcal{T}_\Omega(x^*)$ and show that $-\nabla f(x^*)^T v \leq 0$.

Let $\{x_k\} \subseteq \Omega$ and $\{\lambda_k\}$ be the two sequences given by Theorem 5.1.1. Since $\{x_k\} \to x^*$, there is $k_0 \in \mathbb{N}$ such that for all $k \geq k_0$, $x_k \in B(x^*; \epsilon) \cap \Omega$. Therefore, by Taylor's theorem, $0 \leq f(x_k) - f(x^*) = \nabla f(x^*)^T(x_k - x^*) + o(\|x_k - x^*\|)$. Multiplying by $\lambda_k > 0$, we have

$$\nabla f(x^*)^T[\lambda_k(x_k - x^*)] + o(\|x_k - x^*\|)\lambda_k \geq 0 \qquad (k \geq k_0),$$

which can be rewritten

$$\nabla f(x^*)^T[\lambda_k(x_k - x^*)] + \frac{o(\|x_k - x^*\|)}{\|x_k - x^*\|}\lambda_k\|x_k - x^*\| \geq 0 \qquad (k \geq k_0).$$

Taking the limit, the second term vanishes and there remains $\nabla f(x^*)^T v \geq 0$, which completes the proof. □

To make this result usable in practice, we rely on the explicit problem (V.3), Theorem 5.1.6 and a constraint qualification condition.

Theorem 5.2.2. *Consider Problem (V.3) where the objective function f and the constraint functions h and c are all $\mathcal{C}^1(\Omega)$. Assume the basic constraint qualification condition is satisfied at $x^* \in \Omega$. If x^* is a local constrained minimizer, there exist $\mu \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^p$ such that*

$$\begin{aligned} \nabla f(x^*) - \sum_{i=1}^m \mu_i \nabla h_i(x^*) - \sum_{j=1}^p \lambda_j \nabla c_j(x^*) &= 0, && \text{(V.10a)}\\ \lambda_j c_j(x^*) &= 0 \quad j = 1, \ldots, p, && \text{(V.10b)}\\ h(x) &= 0, && \text{(V.10c)}\\ c(x) &\geq 0, && \text{(V.10d)}\\ \lambda &\geq 0. && \text{(V.10e)} \end{aligned}$$

Proof. From Theorem 5.2.1 and (V.7), there are $(\mu, \lambda) \in \Lambda(x^*)$ such that

$$\nabla f(x^*) - \sum_{i=1}^m \mu_i \nabla h_i(x^*) - \sum_{j \in \mathcal{A}(x^*)} \lambda_j \nabla c_j(x^*) = 0,$$

which is equivalent to (V.10a)–(V.10b). The remaining conditions follow from the feasibility of x^* and the definition of $\Lambda(x^*)$. □

The conditions (V.10) are known as the Karush-Kuhn-Tucker conditions, or KKT for short.

Upon defining the following *Lagrangian* function

$$L(x, \mu, \lambda) = f(x) - \sum_{i=1}^m \mu_i h_i(x) - \sum_{j=1}^p \lambda_j c_j(x) = f(x) - \mu^T h(x) - \lambda^T c(x), \qquad \text{(V.11)}$$

condition (V.10a) may also be written

$$\nabla_x L(x^*, \mu, \lambda) = 0.$$

As in the unconstrained case, it is important to note that the conditions (V.10) are only necessary for optimality. In other words, the KKT conditions also identify saddle points and local maximizers.

Note also the importance of the constraint qualification condition. Without it, there may exist local minimizers that do not satisfy the KKT conditions, as the following example shows.

Example 5.2.1. Consider the problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & x_2 \\ \text{subject to} & x_2^3 - x_1 \geq 0, \\ & x_2^3 + x_1 \geq 0. \end{array}$$

All feasible points (x_1, x_2) have $x_2 \geq 0$ and the feasible set has a *cusp* at $(0,0)$. It is easy to see geometrically that $(0,0)$ is the only optimal solution of this problem. However, the first KKT condition cannot hold since it requires that $\nabla f(0,0)$ be a linear combination of the two active constraint gradients at $(0,0)$. In this case,

$$\nabla f(0,0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \nabla c_1(0,0) = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \text{and} \quad \nabla c_2(0,0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

so there can exist no Lagrange multipliers. Note that in this example, no constraint qualification holds. □

5.3 Second-Order Optimality Conditions for Constrained Optimization

As in the unconstrained case, the first-order conditions are only necessary. Their only role is to identify potential candidates for optimality. Each one of those candidates may be a local minimizer, a local maximizer, or a saddle point. In order to identify in which category each candidate falls, we must resort to second-order information. The necessary and sufficient second-order conditions in the constrained case are more complex than in the unconstrained case because not only the curvature of the objective function is important, but the curvature of the constraints plays a role. This is illustrated geometrically in the following example.

Example 5.3.1. Consider the following problem in $\mathbb{R}^2$

$$\begin{array}{ll} \underset{x_1,x_2}{\text{minimize}} & (x_1-1)^2 + x_2^2 \\ \text{subject to} & x_1 = \beta x_2^2, \end{array}$$

where $\beta \geq 1/2$ is a parameter. Geometrically, the problem consists in finding the points of the parabola given by the constraint that are closest to $(1,0)$. For $\beta = 1/2$, $(0,0)$ is the closest point to $(1,0)$. However, for larger values of β such as $\beta = 2$, the parabola becomes narrow and $(0,0)$ turns into a local maximizer. Two other local minimizers appear on the parabola. Fig. V.4 illustrates the situation. □

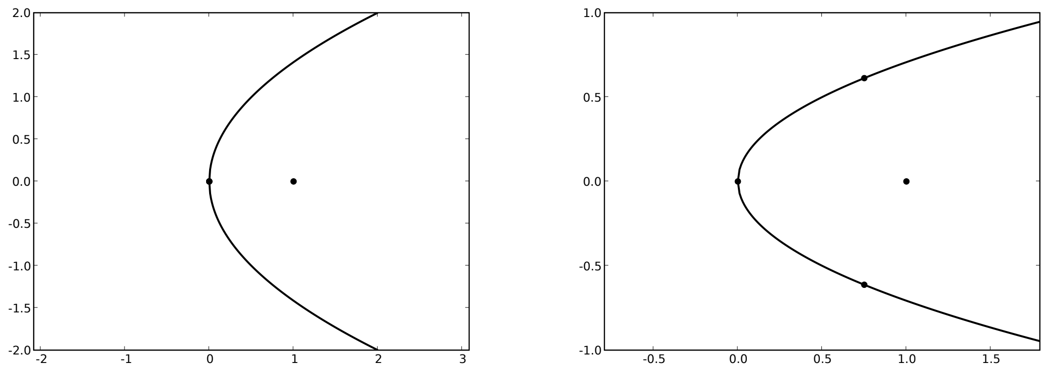


Figure V.4: The curvature of the constraints influences the nature of stationary points. On the left, $\beta = 1/2$ and $(0,0)$ is the only KKT point. For $\beta > 1/2$, two other KKT points appear. On the right, $\beta = 2$ and $(0,0)$ becomes a local maximizer while the other two candidates are local minimizers.

From the results of the previous section, we know that if the objective function f is $\mathcal{C}^1(\Omega)$, and if $x^* \in \Omega$ is a stationary point, there may exist no feasible descent direction for f from x^*. As a consequence, for all feasible direction d from x^*, we have $\nabla f(x^*)^T d \geq 0$. To determine the nature of x^* there remains to examine feasible directions d such that $\nabla f(x^*)^T d = 0$, i.e., feasible directions along which the objective function remains unchanged up to first order.

Let μ^* and λ^* be Lagrange multipliers associated to x^*. Substituting into (V.10a), those directions are such that

$$\sum_{i=1}^{m} \mu_i^* \nabla h_i(x^*)^T d + \sum_{j=1}^{p} \lambda_j^* \nabla c_j(x^*)^T d = 0. \tag{V.12}$$

We must be a bit more stringent on the set of directions that are of concern and for that, we define the *critical cone*

Definition 5.3.1. Let (x^*, μ^*, λ^*) satisfy (V.10) and let $\mathcal{A}(x^*)$ be the set of active constraints at x^*. The set

$$\mathcal{K}(x^*, \mu^*, \lambda^*) = \left\{ d \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} \nabla h_i(x^*)^T d = 0 & i = 1, \dots, m, \\ \nabla c_j(x^*)^T d = 0 & j \in \mathcal{A}(x^*) \text{ such that } \lambda_j^* > 0, \\ \nabla c_j(x^*)^T d \geq 0 & j \in \mathcal{A}(x^*) \text{ such that } \lambda_j^* = 0 \end{array} \right\}$$

is called the *critical cone* at (x^*, μ^*, λ^*).

Note that there may be multiple multipliers (μ^*, λ^*) associated to x^* and that in general, the critical cone differs if different multipliers are used.

It is easy to verify that $\mathcal{K}(x^*, \mu^*, \lambda^*)$ is indeed a cone and if there exists no index $j \in \mathcal{A}(x^*)$ such that $\lambda_j^* = 0$, $\mathcal{K}(x^*, \mu^*, \lambda^*)$ is a subspace that is tangent to the feasible set. It is often referred to as the tangent subspace to the active constraints.

For all $d \in \mathcal{K}(x^*, \mu^*, \lambda^*)$, (V.12) is satisfied and therefore, $\nabla f(x^*)^T d = 0$.

We are now ready to state the necessary second-order conditions.

Theorem 5.3.1. *Let x^* be a local minimizer of (V.3) satisfying the first-order necessary conditions (V.10) and let the MFCQ be satisfied at x^*. Then, for all values of the Lagrange multipliers (μ^*, λ^*) associated to x^*,*

$$d^T \nabla_{xx} L(x^*, \mu^*, \lambda^*) d \geq 0 \quad \textit{for all} \quad d \in \mathcal{K}(x^*, \mu^*, \lambda^*). \tag{V.13}$$

Remark 5.3.1. Note that if x^* is a local minimizer satisfying the KKT conditions, this results must hold for all values of the Lagrange multipliers (μ^*, λ^*) associated to x^*, if there are several from which to choose. Note also that a constraint qualification condition must hold. In the special case where the LICQ holds at x^*, there is only one choice of optimal Lagrange multipliers. When the LICQ does not hold at x^* but the MFCQ does, there is a bounded set to choose from. Conditions (V.13) may hold for some of them and not for others. Theorem 5.3.1 says that (V.13) must hold for all possible choices.

The second-order sufficient conditions are slightly different in nature.

Theorem 5.3.2. *Let (x^*, μ^*, λ^*) satisfy (V.10). If*

$$d^T \nabla_{xx} L(x^*, \mu^*, \lambda^*) d > 0 \quad \textit{for all} \quad d \in \mathcal{K}(x^*, \mu^*, \lambda^*),\ d \neq 0,$$

then x^* is a strict minimizer of (V.3).

Remark 5.3.2. In the sufficient conditions, there is no requirement for a constraint qualification condition to be satisfied at x^*. Of course, x^* must be a KKT point and satisfy (V.10). Note also that the strict inequality must be verified for *one* value of the Lagrange multipliers associated to x^* if there are several from which to choose. Finally, while Theorem 5.3.2 only concludes to a strict minimizer, it is possible to strenghten the assumptions to obtain an isolated minimizer. The required assumptions are that a constraint qualification hold at x^* and that the Lagrange multipliers for which the strict inequality holds be *strictly complementary*, i.e., $c_i(x^*) + \lambda_i^* > 0$ for all $i = 1, \dots, p$.

5.4 Linear Programming

5.4.1 Introduction to Interior-Point Methods

Even though this section is not primarily concerned with the history of interior-point methods, the authors feel that a class of algorithms which has over the years been qualified as a *revolution* [FGW03, Wri98b] deserves a proper, even if brief, historical introduction.

In the 40 years after its introduction in 1947 in [Dan63], the simplex method, an iterative procedure exploiting the property that the set of optimal solutions to linear programs must include vertices, has strongly dominated the field of linear programming. It was and remains a method well suited to competitive implementation, the most prominent modern example being without doubt [CPL98]. Its proven exponential complexity bound, demonstrated in a famous example in [KM72], did not impede on its success, as its empirical performance is almost always polynomial. Simultaneously, during the 1960s, nonlinear constrained problems were thought of as intrinsically different from linear problems and were solved by means of *logarithmic barrier* methods, the classic book [FM68] being a prime theoretical work forming the basis of such methods. Unfortunately, strong concerns about the inherent ill-conditioning of barrier subproblems caused these methods to be considered superannuated and attention turned instead to augmented Lagrangian and SQP frameworks. Khachian's ellipsoid method, based on the notion of analytic center, was introduced in 1979 as the first polynomial-time algorithm for linear programming. It was soon to be abandoned however, as its behaviour in practice was constantly outperformed by the simplex method. The advent of what is nowadays referred to as the *interior-point revolution* happened with the announcement in [Kar84] of a polynomial-time method capable of outperforming the simplex. Its equivalence to logarithmic barrier methods was later formally shown and ill-conditioning was shown to be benign [Wri98a, Wri01]. The simultaneous introduction of a primal-dual framework in [Meg89] and [KMY89] was a departure from Karmarkar's primal setting and proved to be a new dawn on practical and effective methods for large problems. Interestingly, the two appeared in the same volume.

Karmarkar's revolutionary idea of treating linear programs using a sequence of nonlinear problems, although much controverted, played the major role, over the next 20 years, of unifying linear and nonlinear programming. It was successfully applied to such diverse problem classes as quadratic programming, convex programming, semidefinite programming, linear and bilinear matrix inequalities, linear and nonlinear complementarity problems, among others. Across all these classes, its unifying power cannot be ignored. The interested reader will find a wealth of information on the history and evolution of interior-point methods and further references in the excellent review papers [FGW03, Wri92].

It is some of the most recent ideas sparked by Karmarkar's algorithm and the primal-dual setting and developed, for the most part, in the past decade, and some of the most effective numerical implementations for modern large-scale problems, that are discussed next. The general framework for interior-point methods is given in §5.4.2 in the context of linear programming. Attention then turns to practical methods for large-scale quadratic and nonlinear programming problems in §5.5.1 and §5.7.1. For each class of problems, attention will be given to the adjective *large*, which takes different meanings in different contexts.

5.4.2 Interior-Point Methods for Linear Programming

5.4.3 Software

5.5 Quadratic Programming

5.5.1 Interior-Point Methods

5.5.2 Software

5.6 Bound-Constrained Programming

5.6.1 Software

5.7 General Linearly-Constrained Programming

5.7.1 Interior-Point Methods

5.7.2 Software

5.8 Useful Resources on the Internet

The NEOS Server. Online optimization server.
neos.mcs.anl.gov

NEOS **Optimization Software Guide** Categorized optimization software packages.
www.mcs.anl.gov/otc/Guide/SoftwareGuide

Decision tree. A decision tree, online books and tutorials, a dictionary of optimization, tools for automatic differentiation, and more.
plato.la.asu.edu/guide.html

Linear algebra survey. Freely available software for the solution of linear algebra problems.
www.netlib.org/utk/people/JackDongarra/la-sw.html

Optimization Online. A repository of eprints about optimization and related topics.
www.optimization-online.org

Interior-Point Methods Online. This page is no longer maintained but contains a large database of technical reports related to interior-point methods until approximately 2001.
www.mcs.anl.gov/otc/InteriorPoint

Seminal Papers. Nick Gould's list of seminal papers in nonlinear programming.
www.numerical.rl.ac.uk/nimg/msc/lectures/seminal.pdf

Chapter 6

Introduction to Variational Calculus

This chapter is an introduction to the vast subject of calculus of variations. It will prove to be a valuable tool to transition to the continuous optimal control problem in the next chapter. We review here the basic concepts, the first variation of a functional and the Euler-Lagrange equations. We will pay some attention to problems with isoperimetric constraints.

6.1 Introductory examples

The goal of this chapter is to introduce the tools necessary for a preliminary analysis of infinite-dimensional problems. The framework developed here will be useful to transition to continuous optimal control problems. Problems that can be classified as problems of calculus of variations are perhaps best introduced by examples. We start with a few classical problems.

Because it is convenient to denote our unknowns by x and to adhere to some historical notation, unknown functions in this chapter will be denoted $x(t)$. Since the name t for the independent variable often arises from applications involving time, the derivative of x at t, when it exists, will be denoted $\dot{x}(t)$. Similarly, the second derivative of x at t, when it exists, will be denoted $\ddot{x}(t)$.

Consider the following problem: given two different points in the plane, what is the shortest path between them? Of course, we all know that the answer to this question is the straight line between the two points. However, formulating the problem and solving it formally with help us gain insight into related problems. Let us call (t_0, x_0) and (t_1, x_1) the two points in the plane. Without loss of generality, we may assume that $t_0 < t_1$. We could model a path between the two points as a continuous curve originating from the first point and ending at the second. For simplicity, we will also require that this curve be continuously differentiable and will specify it in parametric form $(t, x(t))$ for $t \in [t_0, t_1]$. We are now interested in finding a *function* $x(t)$ such that the parametric curve has minimal length. Recall that the length of such a curve is obtained by integrating the norm of the tangent vector to the curve over $[t_0, t_1]$. Our problem may thus be formulated as

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \displaystyle\int_{t_0}^{t_1} \sqrt{1 + \dot{x}(t)^2}\, \mathrm{d}t \\ \text{subject to} & x(t_0) = t_0, \\ & x(t_1) = t_1. \end{array} \tag{VI.1}$$

Of course, we should be more specific about the domain in which we search for an appropriate function x. We will specify various possibilities in the next section and leave this concern aside for now.

A natural generalization of the above problem is to find the shortest path between two given points on a given surface $\psi(t, u, x) = 0$. If the surface is a sphere or an ellipsoid, this problem may be interesting to identify the shortest flight path between two cities on Earth. The curve $((t, u), x(t, u))$ is now no longer planar but constrained to lie on the given surface, i.e., $\psi(t, u, x(t, u)) = 0$.

Our next example is the famous *brachistochrone* problem, a problem first solved by Johann Bernoulli in 1696.

The name of the problem comes from the Greek *brachistos*, meaning *shortest*, and *chronos*, meaning *time*. Given two distinct points in the plane, $A = (t_0, x_0)$ and $B = (t_1, x_1)$, with $t_0 < t_1$ and $x_0 > x_1 \geq 0$, the problem consists in finding a slide joining the two points such that a unit mass released from A and subject to gravity travels to B in minimum time. The mass is initially at rest and friction is neglected in this first formulation. Again, the slide is modeled as a parametric curve $(t, x(t))$ for $t \in [t_0, t_1]$ satisfying $x(t_0) = x_0$ and $x(t_1) = x_1$. If ds is an element of curve and dt an element of time the speed of the mass at a given time is $v = \mathrm{d}s/\mathrm{d}t$. The total travel time can thus be expressed as

$$T = \int_A^B dt = \int_A^B \frac{1}{v} ds = \int_{t_0}^{t_1} \frac{1}{v(t)} \sqrt{1 + \dot{x}(t)^2}\, \mathrm{d}t. \tag{VI.2}$$

We must now express the speed v in terms of the function x. For that, we use the principle of conservation of total energy. At any given position, the total energy of the mass is constant, and therefore equals its total energy at the starting position. Since the mass is initially at rest, the total energy at the starting position is the potential energy and for all $t \in [t_0, t_1]$ we thus have

$$gx(t) + \tfrac{1}{2}v(t)^2 = gx(t_0) = gx_0.$$

From this last equation we extract

$$v(t) = \sqrt{2g}\sqrt{x_0 - x(t)},$$

taking the positive square root. Introducing this expression for $v(x)$ into the total travel time (VI.2), our problem can be stated as

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \dfrac{1}{\sqrt{2g}} \displaystyle\int_{t_0}^{t_1} \frac{\sqrt{1 + \dot{x}(t)^2}}{\sqrt{x_0 - x(t)}}\, \mathrm{d}t \\ \text{subject to} & x(t_0) = x_0, \\ & x(t_1) = x_1. \end{array} \tag{VI.3}$$

A final introductory example is a minimal surface problem. Consider again two given points in the plane (t_0, x_0) and (t_1, x_1) such that $t_0 < t_1$ and $x_0, x_1 > 0$. Consider a curve joining those two points. Such a curve, when rotated around the t-axis, generates a surface of revolution. The problem is to find a curve such that this surface is minimal. This problem has applications in the study of soap films which naturally form minimum surfaces. As stated, this problem does not necessarily possess a solution, not even a local solution. For consistency, we restrict ourselves to functions $x(t)$ taking nonnegative values. Given $x(t)$, and letting ds denote an element of curve, it is easy to see that an element of surface is given by

$$\mathrm{d}S = 2\pi x(t)\, \mathrm{d}s = 2\pi x(t)\sqrt{1 + \dot{x}(t)^2}\, \mathrm{d}t$$

so that our problem can be stated as

$$\begin{array}{ll} \underset{x}{\text{minimize}} & 2\pi \displaystyle\int_{t_0}^{t_1} x(t)\sqrt{1 + \dot{x}(t)^2}\, \mathrm{d}t \\ \text{subject to} & x(t_0) = x_0, \\ & x(t_1) = x_1, \\ & x(t) \geq 0, \ \forall t \in [t_0, t_1]. \end{array} \tag{VI.4}$$

6.2 Basic definitions

In this section, we introduce some of the basic working tools used later to develop first-order necessary optimality conditions for variational calculus problems.

We will mostly work with $\mathcal{C}^1$ functions but it will often be useful to allow more generality, especially in the framework of control problems. We therefore define the following function sets that will serve as domains for most problems that we encounter in the rest of this chapter.

Definition 6.2.1. Consider an interval $[t_0, t_1] \subset \mathbb{R}$. We define

(a) $\mathcal{C}([t_0, t_1])$ as the set of continuous functions $x : [t_0, t_1] \to \mathbb{R}$.

(b) $\mathcal{C}_{\text{PW}}([t_0, t_1])$ as the set of piecewise continuous functions $x : [t_0, t_1] \to \mathbb{R}$, i.e., the functions which are either continuous or possess a finite number of jump discontinuities in $[t_0, t_1]$. For each such point of discontinuity t, the left and right limits $x(t^-)$ and $x(t^+)$ must exist.

(c) $\mathcal{C}^1([t_0, t_1])$ as the set of continously differentiable functions $x : [t_0, t_1] \to \mathbb{R}$.

(d) $\mathcal{C}^1_{\text{PW}}([t_0, t_1])$ as the set of piecewise continously differentiable functions $x : [t_0, t_1] \to \mathbb{R}$, i.e., those functions x that are differentiable at every $t \in [t_0, t_1]$ and whose derivative $\dot{x} \in \mathcal{C}_{\text{PW}}([t_0, t_1])$.

Note that Definition 6.2.1 concerns closed intervals $[t_0, t_1]$ but we will potentially also consider unbounded intervals such as $]-\infty, t_1]$, $[t_0, +\infty[$, or $]-\infty, +\infty[$. The definitions for such extentions are easily adapted.

From Definition 6.2.1 we see that

$$\mathcal{C}^1([t_0, t_1]) \subset \mathcal{C}^1_{\text{PW}}([t_0, t_1]) \subset \mathcal{C}([t_0, t_1]) \subset \mathcal{C}_{\text{PW}}([t_0, t_1]),$$

and therefore, $\mathcal{C}^1([t_0, t_1])$ is the most restrictive domain of those four, while $\mathcal{C}_{\text{PW}}([t_0, t_1])$ is the most general. For most of our purposes, the four domains defined in Definition 6.2.1 will be sufficient. It is easy to verify that the function spaces defined in Definition 6.2.1 are vector spaces on the field of real numbers when endowed with the usual addition of functions and product with a scalar. Neither of them, however, possesses a finite basis. Those vector spaces are thus all infinite dimensional.

By convention, if $x : [t_0, t_1] \to \mathbb{R}$ is differentiable over $[t_0, t_1]$ except at a finite number of points and if t is such a point, we redefine $\dot{x}(t)$ as the right limit $\dot{x}(t^+)$ whenever $t \in [t_0, t_1[$ and as $\dot{x}(t_1^-)$ if $t = t_1$. This will ensure that $\dot{x}$ is defined at every point of $[t_0, t_1]$. Of course, it may still be discontinuous.

Example 6.2.1. The function defined by

$$x(t) = \begin{cases} t^2 \sin(1/t) & t \neq 0, \\ 0 & t = 0 \end{cases}$$

is in $\mathcal{C}^1_{\text{PW}}([-1, 1])$, but the function defined by

$$x(t) = \begin{cases} 1 & t \in \mathbb{Q} \\ 0 & t \in \mathbb{R} \setminus \mathbb{Q} \end{cases}$$

is not in $\mathcal{C}_{\text{PW}}([t_0, t_1])$ for any interval $[t_0, t_1]$. □

In variational calculus, the problem is to identify a function satisfying a set of prescribed conditions, or *constraints*, and which minimizes locally a given objective. Such an objective must be a function whose domain is an open subset of $\mathcal{C}^1([t_0, t_1])$, $\mathcal{C}^1_{\text{PW}}([t_0, t_1])$, $\mathcal{C}([t_0, t_1])$, or $\mathcal{C}_{\text{PW}}([t_0, t_1])$ for some interval $[t_0, t_1]$, and whose image lies in $\mathbb{R}$. Such a function is traditionally named a *functional*.

Example 6.2.2. The integral operator

$$\begin{array}{ccc} \mathcal{C}_{\text{PW}}([t_0, t_1]) & \to & \mathbb{R} \\ x & \rightsquigarrow & \displaystyle\int_{t_0}^{t_1} x(t)\, \mathrm{d}t \end{array}$$

is a functional. □

Typically, we will consider objectives that can be written in the form

$$\begin{aligned} J: \quad \mathcal{C}_{\text{PW}}([t_0,t_1]) &\rightarrow \mathbb{R} \\ x \quad &\rightsquigarrow J(x) = \int_{t_0}^{t_1} f(x(t),\dot{x}(t),t)\, \mathrm{d}t \end{aligned} \tag{VI.5}$$

where $f : \Omega \subseteq \mathbb{R}^3 \to \mathbb{R}$ is a known function of three variables and Ω is an open subset of $\mathbb{R}^3$ such that for all functions of interest x and all $t \in [t_0,t_1]$, the triples $(x(t),\dot{x}(t),t) \in \Omega$ and $(x(t),\dot{x}(t)^{\pm},t) \in \Omega$. Of course, we will need a set of assumptions on f to be able to formulate optimality conditions.

In order to define the notion of a local minimizer, we need to be able to measure distances and norms in the domains of Definition 6.2.1. This will help formalize the notion of a neighbourhood.

Definition 6.2.2. Given an interval $[t_0,t_1] \subset \mathbb{R}$, we endow the function spaces of Definition 6.2.1 with the following norms.

(a) For each $x \in \mathcal{C}([t_0,t_1])$, $\|x\| = \max_{t\in[t_0,t_1]} |x(t)|$,

(b) For each $x \in \mathcal{C}_{\text{PW}}([t_0,t_1])$, $\|x\| = \sup_{t\in[t_0,t_1]} |x(t)|$,

(c) For each $x \in \mathcal{C}^1([t_0,t_1])$, $\|x\| = \max_{t\in[t_0,t_1]} |x(t)| + \max_{t\in[t_0,t_1]} |\dot{x}(t)|$,

(d) For each $x \in \mathcal{C}^1_{\text{PW}}([t_0,t_1])$, $\|x\| = \max_{t\in[t_0,t_1]} |x(t)| + \sup_{t\in[t_0,t_1]} |\dot{x}(t)|$.

It is easy to verify that each of the above measures satisfies the properties of a norm.

Note that wherever functions may be discontinuous, a maximum must be replaced with a supremum. This is illustrated in the next example.

Example 6.2.3. Consider $[t_0,t_1] = [0,1]$ and the function

$$x(t) = \begin{cases} \frac{1}{2}t^2 & 0 \le t \le \frac{1}{2}, \\ \frac{1}{8} & \frac{1}{2} \le t \le 1. \end{cases}$$

The derivative of x does not exist at $t = \frac{1}{2}$ and we have

$$\dot{x}(t) = \begin{cases} t & 0 \le t < \frac{1}{2}, \\ 0 & \frac{1}{2} < t \le 1. \end{cases}$$

Following our convention, we let

$$\dot{x}(\tfrac{1}{2}) = \dot{x}(\tfrac{1}{2}^+) = \lim_{t\downarrow 0} \dot{x}(\tfrac{1}{2}+t) = 0.$$

With this new definition, we have $x \in \mathcal{C}^1_{\text{PW}}([0,1])$. Moreover, $\sup_{t\in[0,1]} |\dot{x}(t)| = \frac{1}{2}$ but there exists no $t \in [0,1]$ such that $|\dot{x}(t)| = \frac{1}{2}$. □

Definition 6.2.2 allows us to define *neighbour* functions. By way of example, we work with $\mathcal{C}^1_{\text{PW}}([t_0,t_1])$ in what follows, but the same reasoning can be applied to the other function sets of Definition 6.2.1.

Definition 6.2.3. Let $\epsilon > 0$ be fixed. We say that $x, y \in \mathcal{C}^1_{\text{PW}}([t_0,t_1])$ are ϵ-neighbours (or simply *neighbours*) if

$$\|x - y\| = \max_{t\in[t_0,t_1]} |x(t) - y(t)| + \sup_{t\in[t_0,t_1]} |\dot{x}(t) - \dot{y}(t)| < \epsilon.$$

In this case, we also say that y is an ϵ-variation (or simply a *variation*) of x. The set of all ϵ-variations of x is denoted by $B(x;\epsilon)$.

Given Definition 6.2.3, it is straightforward to define the notions of continuity and linearity of functionals as well as the notion of limit in $\mathcal{C}^1_{\text{PW}}([t_0, t_1])$.

In finite-dimensional optimization, an essential tool in our search for local optima was the gradient of the objective function. In the same manner, the concept of differentiability of a functional will be crucial in the present, infinite-dimensional, context.

Definition 6.2.4. Let J be a functional defined on an open domain $\mathcal{D} \subseteq \mathcal{C}^1_{\text{PW}}([t_0, t_1])$. Let $x \in \mathcal{D}$. Since $\mathcal{D}$ is open, there is $\delta > 0$ such that $B(x;\delta) \subset \mathcal{D}$. We say that J is Fréchet-differentiable (or simply *differentiable*) at x if there exist $0 < \epsilon \leq \delta$ and a *linear* functional $\mathrm{d}J_x : B(x, \epsilon) \to \mathbb{R}$ such that for all $y \in B(x;\epsilon)$,

$$J(x + y) = J(x) + \mathrm{d}J_x(y) + o(y - x),$$

where the function $o(y - x)$ is such that

$$\lim_{y \to x} \frac{o(y - x)}{\|y - x\|} = 0.$$

The linear functional $\mathrm{d}J_x$ is termed the *Fréchet differential*, or the *first variation* of J at x.

The equivalent of the first variation for differentiable functions defined on $\mathbb{R}^n$ is the differential. If $\phi : \mathbb{R}^n \to \mathbb{R}$ is differentiable at $z \in \mathbb{R}^n$, the differential of ϕ at z is the linear function $\mathrm{d}\phi_z(w) = \nabla\phi(z)^T(w - z)$. This is an expression that was used extensively in the earlier chapters. In particular, we can think of $w - z$ as a *direction* emanating from z and of $\mathrm{d}\phi_z(w)$ as the slope of ϕ at z in the direction $w - z$, i.e., the directional derivative of ϕ at z in the direction $w - z$. We then know that if z is a local minimizer, there can be no feasible descent direction for ϕ from z. In the constrained case, this means that $\mathrm{d}\phi_z(w) \geq 0$ for all w such that $z - w$ is a feasible direction. In the unconstrained case, it means that $\mathrm{d}\phi_z(w) = 0$ for all w.

We give two immediate properties of the first variation in the next result.

Proposition 6.2.1. *Suppose the Fréchet differential of J at x exists. Then*

(a) it is unique,

(b) J is continuous at x.

6.3 Minimization

Given the concept of proximity in Definition 6.2.3, it is now easy to define global and local minimizers.

Definition 6.3.1. Let the functional J be defined on $\mathcal{C}^1_{\text{PW}}([t_0, t_1])$ and let $x \in \mathcal{C}_{\text{PW}}([t_0, t_1])$.

(a) x is a global minimizer of J if and only if $J(x) \leq J(y)$ for all $y \in \mathcal{C}^1_{\text{PW}}([t_0, t_1])$,

(b) x is a local minimizer of J if and only if there exists $\epsilon > 0$ such that $J(x) \leq J(y)$ for all $y \in B(x;\epsilon)$.

Necessary optimality conditions exist when J is Fréchet-differentiable.

Proposition 6.3.1. *Suppose $x \in \mathcal{C}^1_{PW}([t_0, t_1])$ is a local minimizer of J and suppose J is Fréchet-differentiable at x. Then, $\mathrm{d}J_x(h) = 0$ for all $h \in \mathcal{C}^1_{PW}([t_0, t_1])$ having the same points of discontinuity as $\dot{x}$.*

Proposition 6.3.1 is fairly general and parallels the result in finite dimension that there may exist no descent direction from a local minimizer. Note that Propostion 6.3.1 only gives necessary conditions.

It is difficult to derive practically useful results without making assumptions on the form of the functional J. For this reason, we assume from now on that J has the form (VI.5) and that the following assumption is satisfied.

Assumption 6.3.1. *The function f is continuous in x, $\dot{x}$ and t and has continuous partial derivatives in x and $\dot{x}$.*

In this case, it is possible to obtain an explicit expression for the Fréchet differential of J. Under Assumptions 6.3.1, it is possible to show that J is Fréchet-differentiable. Suppose $x \in \mathcal{C}^1_{\text{PW}}([t_0, t_1])$ is a local minimizer of J. It is easy to show that for all $h \in \mathcal{C}^1_{\text{PW}}([t_0, t_1])$ having the same points of discontinuity as $\dot{x}$,

$$\mathrm{d}J_x(h) = \int_{t_0}^{t_1} \frac{\partial f}{\partial x}(x(t), \dot{x}(t), t)\; h(t)\; \mathrm{d}t + \int_{t_0}^{t_1} \frac{\partial f}{\partial \dot{x}}(x(t), \dot{x}(t), t)\; \dot{h}(t)\; \mathrm{d}t. \tag{VI.6}$$

From Proposition 6.3.1, this last expression must vanish for all functions h of interest. This condition is not yet very practical. However, after integrating by parts and using the Lemma of Dubois-Reymond, we obtain a result of practical interest. This result applies to the problem of the minimization of (VI.5) where the function x is required to satisfy the *fixed end-points conditions*

$$x(t_0) = x_0, \qquad x(t_1) = x_1$$

where $x_0, x_1 \in \mathbb{R}$ are given.

Theorem 6.3.1 (Euler-Lagrange Conditions)**.** *Under our assumptions on the function f, suppose $x \in \mathcal{C}^1_{PW}([t_0, t_1])$ is a local minimizer of the functional (VI.5) over all functions in $\mathcal{C}^1_{PW}([t_0, t_1])$ satisfying the fixed end-points conditions $x(t_0) = x_0$ and $x(t_1) = x_1$. Then every smooth section of x must satisfy the second-order partial-differential equation*

$$\frac{\partial f}{\partial x}(x(t), \dot{x}(t), t) - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial f}{\partial \dot{x}}(x(t), \dot{x}(t), t) = 0. \tag{VI.7}$$

The conditions (VI.7), referred to as the Euler-Lagrange equations, are a central result in variational calculus. Note that among other things, the result states that $\partial f / \partial \dot{x}$ is differentiable, which was not assumed. Note also that (VI.7) really is a second-order equation, for if we develop the derivatives, we obtain

$$\frac{\partial f}{\partial x}(x, \dot{x}, t) - \frac{\partial^2 f}{\partial x \partial \dot{x}}(x, \dot{x}, t)\dot{x} - \frac{\partial^2 f}{\partial \dot{x}^2}(x, \dot{x}, t)\ddot{x} - \frac{\partial^2 f}{\partial t \partial \dot{x}}(x, \dot{x}, t) = 0, \tag{VI.8}$$

provided we assume that the second partial derivatives exist and are continuous, and where we dropped the argument t for readability.

In general, it can be tedious and cumbersome to write out the Euler-Lagrange equations and seek a solution. In some cases, the form of f is such that (VI.7) largely simplifies. This fact is used extensively in physics and in particular in mechanics where well-known results such as the conservation of energy follow from those simplifications.

6.3.1 First integrals of the Euler-Lagrange equation

Assume that in Theorem 6.3.1, the function f does not depend explicitly on t, i.e., $\partial f / \partial t = 0$. Assuming the second partial derivatives of x exist and are continuous, we then have from (VI.8) that

$$\frac{\partial f}{\partial x}(x, \dot{x}, t) - \frac{\partial^2 f}{\partial x \partial \dot{x}}(x, \dot{x}, t)\dot{x} - \frac{\partial^2 f}{\partial \dot{x}^2}(x, \dot{x}, t)\ddot{x} = 0.$$

Multiplying this last equation by $\dot{x}$, it is easy to see that we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[f(x(t), \dot{x}(t), t) - \frac{\partial f}{\partial \dot{x}}(x(t), \dot{x}(t), t)\dot{x}(t) \right] = 0, \qquad t \in [t_0, t_1].$$

The function between square brackets must therefore remain constant over the interval $[t_0, t_1]$, i.e., there must exist a constant $\alpha \in \mathbb{R}$ such that

$$f(x(t), \dot{x}(t), t) - \frac{\partial f}{\partial \dot{x}}(x(t), \dot{x}(t), t)\dot{x}(t) = \alpha, \qquad t \in [t_0, t_1]. \tag{VI.9}$$

In this case, the second-order Euler-Lagrange equations simplify into a first-order partial-differential equation. Because the function on the right-hand side of (VI.9) must remain constant, it is termed a *first integral* of the Euler-Lagrange equations. This terminology is often used in mechanics where, for instance, the Hamiltonian must remain constant in conservative systems. Physicists say that the Hamiltonian is a first integral of the system.

Consider now the case where the function f does not depend explicitly on x. The we have immediately from (VI.7) that there must be a constant $\beta \in \mathbb{R}$ such that

$$\frac{\partial f}{\partial \dot{x}}(x(t), \dot{x}(t), t) = \beta, \qquad t \in [t_0, t_1], \tag{VI.10}$$

and $\partial f / \partial \dot{x}$ is, in this case, another first integral of the Euler-Lagrange equations. The constants in the two above first integrals are usually determined from the end-point conditions.

6.3.2 Problems with several dependent variables

The results of §6.3 generalize to the situation where the objective function depends on several unknown functions $x_1(t)$, …, $x_n(t)$:

$$J(x_1, \dots, x_n) = \int_{t_0}^{t_1} f\left(x_1(t), \dots, x_n(t), \dot{x}_1(t), \dots, \dot{x}_n(t), t\right)\, \mathrm{d}t, \tag{VI.11}$$

where this time $f : R^{2n+1} \to \mathbb{R}$ satisfies assumptions similar to those of §6.3.

A prime example of this situation is the principle of least action according to which the evolution of a system is always such that the *action* is minimized. In $\mathbb{R}^3$ and in cartesian coordinates, the action between instants t_0 and t_1 is defined as

$$\int_{t_0}^{t_1} \tfrac{1}{2}\left(\dot{x}(t)^2 + \dot{y}(t)^2 + \dot{z}(t)^2\right) - U(x(t), y(t), z(t))\, \mathrm{d}t$$

where $U(x, y, z)$ is a potential. In this minimization, the three functions $x(t)$, $y(t)$ and $z(t)$ must be determined. A second example concerns the determination of a *geodesic*—a shortest path between two points lying on a given surface, as mentioned in the introductory examples of the present chapter. Problems with several dependent variables also often occur when curves are represented in parametric form. Consider for instance the first introductory example of the present chapter, that of finding the shortest curve between two points of the plane. Note that this is a special case of a geodesic. A possible approach to this problem is to look for a parametric curve of the form $(x(t), y(t))$ in the plane, defined for $t \in [0, 1]$ and such that $x(0) = x_0$, $y(0) = y_0$, $x(1) = x_1$ and $y(1) = y_1$ where (x_0, y_0) and (x_1, y_1) are the points to be joined. The length of the curve in this case is expressed as

$$J(x, y) = \int_0^1 \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}\, \mathrm{d}t \tag{VI.12}$$

and we have two functions to determine, $x(t)$ and $y(t)$, subject to the fixed end-points conditions.

It can be show that in this situation, and under assumptions similar to those of §6.3 on f, we have the following result.

Theorem 6.3.2 (Euler-Lagrange Conditions for Multiple Dependent Variables)**.** *Under our assumptions on the function f, suppose $x = (x_1, x_2, \ldots, x_n) \in \mathcal{C}^1_{PW}([t_0, t_1])^n$ is a local minimizer of the functional (VI.5) over all functions in $\mathcal{C}^1_{PW}([t_0, t_1])^n$ whose first derivatives have the same points of jump discontinuity as $\dot{x}(t)$ and satisfying the fixed end-points conditions $x(t_0) = x_0 \in \mathbb{R}^n$ and $x(t_1) = x_1 \in \mathbb{R}^n$. Then every smooth section of each x_i must satisfy the second-order partial-differential equation*

$$\frac{\partial f}{\partial x_i}(x(t), \dot{x}(t), t) - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial f}{\partial \dot{x}_i}(x(t), \dot{x}(t), t) = 0, \qquad \textit{for } i = 1, \ldots, n, \tag{VI.13}$$

where we used the notation $x(t) = (x_1(t), \ldots, x_n(t))$ and $\dot{x}(t) = (\dot{x}_1(t), \ldots, \dot{x}_n(t))$.

Regarding the equations (VI.13), the same first integrals as in §6.3.1 apply, for each dependent variable x_i. Moreover, when f does not depend explicitly on t, we have the important additional first integral

$$f(x(t), \dot{x}(t), t) - \sum_{i=1}^{n} \dot{x}_i(t)\frac{\partial f}{\partial \dot{x}_i}(x(t), \dot{x}(t), t) = \gamma, \qquad t \in [t_0, t_1], \tag{VI.14}$$

where $\gamma \in \mathbb{R}$ is a constant.

Example 6.3.1. We consider the example (VI.12). We have the two Euler-Lagrange equations

$$\frac{\partial f}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial f}{\partial \dot{x}} = 0 \qquad \text{and} \qquad \frac{\partial f}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial f}{\partial \dot{y}} = 0.$$

Since f depends explicitly on neither x nor y, the first integral (VI.10) applies to both equations. Since $f(x, y, \dot{x}, \dot{y}, t) = \sqrt{\dot{x}^2 + \dot{y}^2}$, this yields

$$\frac{\dot{x}}{\sqrt{\dot{x}^2 + \dot{y}^2}} = \alpha_1 \qquad \text{and} \qquad \frac{\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}} = \beta_1,$$

for some constants α_1 and β_1. Next, since f does not depend explicitly on t, the first integral (VI.9) also applies to both Euler-Lagrange equations, i.e.,

$$\frac{\dot{x}^2}{\sqrt{\dot{x}^2 + \dot{y}^2}} = \alpha_2 = \dot{x}\alpha_1, \qquad \text{and} \qquad \frac{\dot{y}^2}{\sqrt{\dot{x}^2 + \dot{y}^2}} = \beta_2 = \dot{y}\beta 1,$$

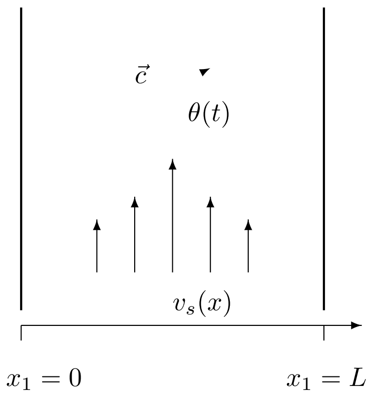
for some constants α_2 and β_2. Combining these sets of equations, we see that both $\dot{x}(t)$ and $\dot{y}(t)$ must remain constant. Again, we find the result that the shortest path, described by a parametric curve, joining two points of the plane is a segment of straight line. Its precise expression is found by imposing the end-point conditions. Note that in this example, the first integral (VI.14) applies but does not bring any additional information. □

6.4 Problems with variable end points

6.4.1 Introductory Example

Problems with variable end points arise naturally as in the following example.

Consider a river with parallel straight banks. Without loss of generality, we assume that the banks are vertical straight lines at $x_1 = 0$ and $x_1 = L > 0$. The velocity of the stream is given by $v_s(x) = (0, v(x_1))$ where v is a given function. Starting from $(0,0)$, a boat must cross the river with constant speed $c > 0$ and reach the opposite bank in minimum time. In this problem, the end point, i.e., the exact place where the boat meets the opposite bank, is free. If we let $\theta(t)$ denote the course of the boat (i.e, the angle between the horizontal and the velocity vector of the boat), we have



$$\dot{x}_1(t) = c\cos\theta(t), \qquad \dot{x}_2(t) = v(x_1(t)) + c\sin\theta(t).$$

In Chapter 7, we will formulate this problem in an intuitive way where the course $\theta(t)$ is a control upon which to act in order to steer the ship. For the time being, we may describe the trajectory of the ship as a function $x_2 = x_2(x_1)$ and try and eliminate $\theta(t)$ from the equations. The total travel time may be written

$$T = \int_0^T \mathrm{d}t = \int_0^L \frac{\mathrm{d}t}{\mathrm{d}x_1}\,\mathrm{d}x_1 = \int_0^L \frac{1}{c\cos\theta(t)}\,\mathrm{d}x_1.$$

The preceding equations give

$$x_2'(x_1) = \frac{\mathrm{d}x_2}{\mathrm{d}x_1} = \frac{v(x_1(t)) + c\sin\theta(t)}{c\cos\theta(t)} = \frac{v(x_1(t)) \pm c\sqrt{1-\cos^2\theta(t)}}{c\cos\theta(t)}.$$

Multiplying through by $c\cos\theta(t)$ and squaring, we obtain a quadratic equation in $c\cos\theta(t)$. For this quadratic equation to have real roots, we assume that $v(x_1)^2 < c^2$. After elimination, we are left with

$$T = \int_0^L \frac{1 + x_2'(x_1)^2}{v(x_1)x_2'(x_1) + \sqrt{c^2(1 + x_2'(x_1)^2) - v(x_1)^2}}\,\mathrm{d}x_1.$$

This last integral must be minimized over all functions $x_2(x_1)$ such that

$$x_2(0) = 0, \qquad x_2(L) \text{ is free.}$$

It is easy to generalize the problem by leaving the starting point free, instead of fixing it at $(0,0)$. In fact, because of the form of the velocity field above, the initial point will be irrelevant in the solution. However, for more general velocity fields $(v_1(x_1,x_2), v_2(x_1,x_2))$, the initial point will influence the solution. The problem can be further generalized by assuming that the banks of the river are described by more general functions.

6.4.2 Transversality conditions

In a more general setting, the variable end point may have to lie on a given curve $(t, \psi(t))$, i.e., we may impose $x(t_1) = \psi(t_1)$ where ψ is a known function. In the preceding example, the curve, i.e., the graph of ψ, was a vertical straight line. Some problems may leave both end points unspecified and may even leave the final time t_1 unspecified, as in the preceding example. In such cases, the necessary optimality conditions consist in the Euler-Lagrange equations and a number of additional conditions called the *transversality conditions*. We give the following result without proof.

Theorem 6.4.1. *Under our assumptions on the function f, suppose $x \in \mathcal{C}^1_{PW}([t_0,t_1])$ is a local minimizer of the functional (VI.5) over all functions in $\mathcal{C}^1_{PW}([t_0,t_1])$ satisfying the end-points conditions*

$$x(t_0) \textit{ is free or must satisfy } x(t_0) = \psi_0(t_0),$$

and

$$x(t_1) \text{ is free or must satisfy } x(t_1) = \psi_1(t_1),$$

where $\psi_0, \psi_1 : \mathbb{R} \to \mathbb{R}$ are known $\mathcal{C}^1$ functions and where t_0 and/or t_1 are possibly unspecified. Then every smooth section of x must satisfy the Euler-Lagrange equation (VI.7) over $[t_0, t_1]$. Moreover, the following transversality conditions *must hold at the end points:*

(a) if t_i is given and $x(t_i)$ is fixed $(i = 0, 1)$, then $x(t_i) = x_i$,

(b) if t_i is given but $x(t_i)$ is free $(i = 0, 1)$, then

$$\frac{\partial f}{\partial \dot{x}}(x(t_i), \dot{x}(t_i), t_i) = 0, \tag{VI.15}$$

(c) if t_i is free $(i = 0, 1)$, then

$$f(x(t_i), \dot{x}(t_i), t_i) + \left(\dot{\psi}_i(t_i) - \dot{x}(t_i)\right) \frac{\partial f}{\partial \dot{x}}(x(t_i), \dot{x}(t_i), t_i) = 0. \tag{VI.16}$$

In the case where an end point is left free, the condition (VI.15) is called the *natural bound* condition. The natural bound condition is a transversality condition and is in fact a special case of (VI.16) when $\dot{\psi}_i(t_i) = +\infty$. Such is the case in the introductory example of the previous section.

Example 6.4.1. In the (t, x) plane, consider the problem of finding the shortest curve emanating from the origin and intercepting a given curve $(t, \psi(t))$ for some unknown value of t. Here, $t_0 = 0$ is given and $x(0) = 0$ is fixed. Regarding t_1, we are in case (c) of Theorem 6.4.1. We already know from the Euler-Lagrange equations that $\dot{x}(t)$ must remain constant. We now impose (VI.16) to $f(x, \dot{x}, t) = \sqrt{1 + \dot{x}^2}$ at t_1 to obtain

$$\sqrt{1 + \dot{x}(t_1)^2} + \left(\dot{\psi}(t_1) - \dot{x}(t_1)\right) \frac{\dot{x}(t_1)}{\sqrt{1 + \dot{x}(t_1)^2}} = 0.$$

This equation gives immediately that $\dot{x}(t_1) = -1/\dot{\psi}(t_1)$, which means that the two curves must be orthogonal where they meet. The value of t_1 may be determined upon solving the set of algebraic equations $x(t_1) = \psi(t_1)$ and $\dot{\psi}(t_1)\dot{x}(t_1) = -1$. □

Example 6.4.1 may be at the origin of the term *transversality* conditions, to mean *orthogonality*.

Example 6.4.2. We are looking to determine a signal $x(t)$ over the interval $[0, 1]$ that is as close as possible to e^{-t} and whose derivative is as close as possible to -1 over the whole interval. The signal must be such that $x(0) = 1$. This problem may be formulated as

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2} \int_0^1 (x(t) - e^{-t})^2 + (\dot{x}(t) + 1)^2 \, \mathrm{d}t \\ \text{subject to} & x(0) = 1, \\ & x(1) \text{ is free.} \end{array}$$

The Euler-Lagrange equations for this problem read

$$\ddot{x}(t) - x(t) = -e^{-t}.$$

The general solution to this second-order differential equation has the form

$$x(t) = \alpha_1 e^t + \alpha_2 e^{-t} + \tfrac{1}{2} t e^{-t}$$

where $\alpha_1, \alpha_2 \in \mathbb{R}$ are integration constants. We impose the initial condition $x(0) = 1$ to have $\alpha_1 + \alpha_2 = 1$ and the natural bound (b) of Theorem 6.4.1, to obtain $\dot{x}(1) = -1$, i.e, $\alpha_1 e - \alpha_2 e^{-1} = -1$. We conclude that $\alpha_1 = (1 - e)/(e^2 + 1)$ and $\alpha_2 = 1 - \alpha_1$ so that

$$x(t) = \frac{1 - e}{e^2 + 1}\, e^t + e\frac{1 + e}{e^2 + 1}\, e^{-t} + \tfrac{1}{2} t e^{-t}, \qquad t \in [0, 1].$$

□

6.5 Problems with isoperimetric constraints

Essentially, the problems of the previous sections were unconstrained, save for some end-point conditions. More interesting problems however will feature more complex constraints that a solution will have to satisfy. Often, such constraints will themselves be given by a functional. Let us look at an example to make things more tangible. Consider the famous problem of the hanging cable. A flexible cable of given length with negligible section is hooked at its ends to two opposite walls. It hangs subject to gravity and we are looking for a function to describe its shape when it stabilizes. Let the walls be at abscissae $t = 0$ and $t = 1$ and let $x \in \mathcal{C}([0,1])$ describe the shape of the cable. Physicits know that the cable stabilizes when its potential energy is minimal—that is therefore what x will need to minimize. If $\mathrm{d}s$ is an element of cable, its length is $\sqrt{1+\dot{x}(t)^2}\,\mathrm{d}t$ and its mass is given by its length multiplied by the linear density $\rho > 0$ of the cable. We assume without loss of generality that $\rho = 1$. If the horizontal axis is taken as the ground level, the potential energy of the element of cable is given by $x(t)$ multiplied by its mass. The problem may thus be stated as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \int_0^1 x(t)\sqrt{1+\dot{x}(t)^2}\,\mathrm{d}t \\ \text{subject to} \quad & \int_0^1 \sqrt{1+\dot{x}(t)^2}\,\mathrm{d}t = L, \\ & x(0) = x_0, \\ & x(1) = x_1, \end{aligned} \tag{VI.17}$$

where the objective function is the total potential energy of the cable, x_0 and x_1 are the heights at which the cable ends are hooked to each wall, $L > 0$ is the length of cable and the constraint specifies the length constraint. Note that this constraint is given by a functional. A constraint of this form is called an *isoperimetric* constraint.

A problem with isoperimetric constraints has the general form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \int_{t_0}^{t_1} f(x(t),\dot{x}(t),t)\,\mathrm{d}t \\ \text{subject to} \quad & \int_{t_0}^{t_1} h_i(x(t),\dot{x}(t),t)\,\mathrm{d}t = 0, \qquad i = 1,\ldots,m, \\ & x(t_0) = x_0, \\ & x(t_1) = x_1, \end{aligned} \tag{VI.18}$$

where $x(t) = (x_1(t),\ldots,x_n(t))$ and, possibly, one or several end-points may be free or be required to meet a given curve. Finally, one or both of t_0 and t_1 may be free.

Note that the isoperimetric constraint of (VI.17) has the form required in (VI.18) by setting

$$h(x,\dot{x},t) = \sqrt{1+\dot{x}^2} - \frac{L}{t_1 - t_0}.$$

To solve isoperimetric problems, we first introduce an auxilliary function.

Definition 6.5.1. Let the isoperimetric problem (VI.18) be given. The function defined by

$$H(x,\dot{x},t,\lambda) = f(x,\dot{x},t) - \sum_{i=1}^{m} \lambda_i h_i(x,\dot{x},t), \tag{VI.19}$$

where $\lambda_1,\ldots,\lambda_m \in \mathbb{R}$ is called the *Hamiltonian* of the problem. The scalars $\lambda_1,\ldots,\lambda_m$ are called *Lagrange multipliers*.

Note that the Hamiltonian is not quite defined as the Lagrangian in finite-dimensional optimization but embodies the corresponding concept. In the main result of this section, we will see that the usage of the Hamiltonian is similar to the usage of a Lagrangian. Note also that when there are no isoperimetric constraints, the Hamiltonian is simply $f(x, \dot{x}, t)$.

Assumption 6.5.1. *The functions f and h_i $(i = 1, \dots, m)$ are continuous in x, $\dot{x}$ and t, and have continuous partial derivatives in x and $\dot{x}$.*

Before we can state our main result, we need a condition on the algebraic representation of the feasible set—a concept similar to a constraint qualification. We set the following definition.

Definition 6.5.2. Let $\bar{x} \in \mathcal{C}^1([t_0, t_1])^n$ be given. We say that $\bar{x}$ is *regular* for the problem (VI.18) if the functions

$$h_1(\bar{x}(t), \dot{\bar{x}}(t), t),\ h_2(\bar{x}(t), \dot{\bar{x}}(t), t),\ \dots,\ h_m(\bar{x}(t), \dot{\bar{x}}(t), t)$$

are linearly independent in $\mathcal{C}^1([t_0, t_1])^m$.

We can now state the main result of this section.

Theorem 6.5.1 (Euler-Lagrange Conditions for Isoperimetric Constraints)**.** *Under Assumption 6.5.1, suppose $x = (x_1, x_2, \dots, x_n) \in \mathcal{C}^1([t_0, t_1])^n$ is a local minimizer of (VI.18) which is* regular*. Then there must exist $\lambda_1, \dots, \lambda_m \in \mathbb{R}$ such that each x_i satisfies the second-order partial-differential equation*

$$\frac{\partial H}{\partial x_i}(x(t), \dot{x}(t), t, \lambda) - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial H}{\partial \dot{x}_i}(x(t), \dot{x}(t), t, \lambda) = 0, \qquad \textit{for } i = 1, \dots, n, \tag{VI.20}$$

where we used the notation $x(t) = (x_1(t), \dots, x_n(t))$, $\dot{x}(t) = (\dot{x}_1(t), \dots, \dot{x}_n(t))$ and where $H(x, \dot{x}, t, \lambda)$ is the Hamiltonian defined in (VI.19). The values of the Lagrange multipliers $\lambda_1, \dots, \lambda_m$ are found by imposing satisfaction of the isoperimetric constraints of (VI.18).

There is a theorem similar to Theorem 6.4.1 stating natural bounds and transversality conditions when the end-points are either not fixed or must follow a curve, or when one or both of t_0 and t_1 are free.

Finally, the first integrals of the previous sections apply to (VI.20) under the same conditions.

Example 6.5.1. This example is the quintessential Dido problem. Consider all the closed curves in parametric form $(x(t), y(t))$ where $t \in [0, 1]$ and $x, y \in \mathcal{C}^1([0, 1])$ and such that $x(0) = x(1) = \bar{x}$ and $y(0) = y(1) = \bar{y}$. We require that these curves not be self intersecting. Among all such curves whose length is fixed, say to $L > 0$, we seek one such that the *enclosed area* is maximal. We see immediately that the length requirement imposes an isoperimetric constraint. For such problems it is necessary to specify that, when computing the enclosed area, the curve must be followed counter-clockwise as t goes from 0 to 1. It can be shown that for such curves, the enclosed area is given by

$$\oint_C \mathrm{d}S = \tfrac{1}{2}\int_0^1 x(t)\dot{y}(t) - y(t)\dot{x}(t)\ \mathrm{d}t.$$

The problem can thus be stated as

$$\begin{array}{ll} \underset{x,y}{\text{minimize}} & -\displaystyle\int_0^1 (x(t)\dot{y}(t) - y(t)\dot{x}(t))\ \mathrm{d}t \\ \text{subject to} & \displaystyle\int_0^1 \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}\ \mathrm{d}t = L, \\ & x(0) = x(1) = \bar{x}, \\ & y(0) = y(1) = \bar{y}. \end{array}$$

We start by forming the Hamiltonian

$$H(x, y, \dot{x}, \dot{y}, t, \lambda) = -x\dot{y} + y\dot{x} - \lambda(\sqrt{\dot{x}^2 + \dot{y}^2} - L)$$

and the conditions (VI.20)

$$\frac{\partial H}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial H}{\partial \dot{x}} = 0, \qquad \text{and} \qquad \frac{\partial H}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial H}{\partial \dot{y}} = 0,$$

which read

$$-\dot{y}(t) - \frac{\mathrm{d}}{\mathrm{d}t}\left(y(t) - \frac{\lambda \dot{x}(t)}{\sqrt{\dot{x}^2+\dot{y}^2}}\right) = -\frac{\mathrm{d}}{\mathrm{d}t}\left(2y(t) - \frac{\lambda \dot{x}(t)}{\sqrt{\dot{x}^2+\dot{y}^2}}\right) = 0,$$

and

$$\dot{x}(t) - \frac{\mathrm{d}}{\mathrm{d}t}\left(-x(t) - \frac{\lambda \dot{y}(t)}{\sqrt{\dot{x}^2+\dot{y}^2}}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(2x(t) + \frac{\lambda \dot{y}(t)}{\sqrt{\dot{x}^2+\dot{y}^2}}\right) = 0.$$

Therefore there are constants $\alpha, \beta \in \mathbb{R}$ such that

$$x(t) - \alpha = \frac{-\lambda \dot{y}(t)}{2\sqrt{\dot{x}^2+\dot{y}^2}} \qquad \text{and} \qquad y(t) - \beta = \frac{\lambda \dot{x}(t)}{2\sqrt{\dot{x}^2+\dot{y}^2}},$$

for all $t \in [0,1]$. Squaring these two equations and adding them together, we obtain

$$(x(t)-\alpha)^2 + (y(t)-\beta)^2 = \frac{\lambda^2}{4}.$$

In other words, the optimal curve is a perfect circle. The value of λ is given by imposing the isoperimetric constraint. The circumference of the circle above is $\pi|\lambda|$ and therefore we must have $|\lambda| = L/\pi$. For instance, the curve of length 2π of maximal enclosing area is the unit circle. $\square$

Exercise 6.5.1. Consider the problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \displaystyle\int_0^1 \sqrt{\dot{\phi}(t)^2 \sin(\phi(t))^2 + \dot{\theta}(t)^2}\ \mathrm{d}t \\ \text{subject to} & \theta(0) = \theta_0,\ \phi(0) = \phi_0, \\ & \theta(1) = \theta_1,\ \phi(1) = \phi_1. \end{array}$$

Using first integrals of the Euler-Lagrange equations, show that

$$\theta(t) = \theta_0 + t(\theta_1 - \theta_0), \tag{VI.21a}$$
$$\phi(t) = \operatorname{acos}\left[\cos(\phi_0) + t(\cos(\phi_1) - \cos(\phi_0))\right]. \tag{VI.21b}$$

Exercise 6.5.2. Use the optimality conditions for isoperimetric problems to determine the shape of the hanging cable (VI.17).

Exercise 6.5.3. Solve the isoperimetric problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \displaystyle\int_0^1 \sqrt{1+\dot{x}(t)^2}\ \mathrm{d}t \\ \text{subject to} & \displaystyle\int_0^1 x(t)\ \mathrm{d}t = \frac{\pi}{8}, \\ & x(0) = 0,\ x(1) = 0. \end{array}$$

Exercise 6.5.4. Consider the Dido problem of finding the function $x \in \mathcal{C}^1([0,1])$ such that $x(0) = x(1) = 0$ and with curve length $\pi/2$ and such that the area between the curve $(t, x(t))$ and the horizontal axis is maximal. Solve this problem in cartesian and in polar coordinates $(\theta, r(\theta))$.

Exercise 6.5.5. Solve the same problem as in the previous exercise on the interval $[0,T]$, where $x(0) = a$, $x(T) = b$ and where the curve length is $L > 0$. On what conditions on T, a, b and L is the solution a semicircle?

Chapter 7

Introduction to Optimal Control

This chapter is a natural sequel to variational calculus and covers the basic optimality conditions of continuous-time optimal control problems. The minimum principle constitutes the main theoretical result of this chapter. We examine direct and indirect numerical methods.

Optimal control problems are infinite-dimensional optimization problems in the same was as variational calculus problems—the unknowns are functions defined over certain intervals. It happens however that in those problems, as set of variables has an interpretation in terms of the command of a given system. For example, an aircraft pilot commands the flaps, elevators, ailerons, rudder and thrust to steer the aircraft. The position, speed and acceleration of the aircraft abide with the laws of motion and, through proper action upon the commands, the pilot has control on the trajectory of the aircraft. Such commands have an intrinsic interpretation in terms of the physics of the system and will be called *controls*. On the other hand, position, speed and acceleration can be viewed as a joint consequence of the action upon the controls and the laws of motion. For this reason, it is natural to consider those two sets of variables as separate. In an optimal control problem, one seeks the precise *control program* that will cause the system to behave in such a way as to minimize or maximize some efficiency measure of the system.

7.1 Problem Formulation

Consider a dynamical system in continuous time described at time t by the set $x_i(t)$, $i = 1, \ldots, n$, of *state variables*. The evolution of this system is governed by an initial-value problem

$$\dot{x}_i(t) = f_i(x(t), u(t), t), \quad x_i(t_0) = x_i^0, \quad i = 1, \ldots, n, \tag{VII.1}$$

with given initial conditions. The differential equations (VII.1) depend on some *control variables* $u_j(t)$, $j = 1, \ldots, m$. Typically, we will seek $x_i \in \mathcal{C}^1_{\text{PW}}([t_0, t_1])$ and $u_j \in \mathcal{C}_{\text{PW}}([t_0, t_1])$ where t_0 is a given initial time and t_1 is a final time, which may or may not be given.

For a given choice of the control variables, say $u = \bar{u}$, we assume that the initial-value problem (VII.1) has a unique solution $\bar{x}$. This solution $\bar{x}$ is called the *trajectory* associated to the choice $\bar{u}$. Through the control variables, we can force the sytem to follow a certain trajectory.

Given this flexibility, we seek a value of the control variables u that minimize a given measure of the efficiency of the above dynamical system. As a start, we write this measure of efficiency as

$$J(u) = h(x(t_1)) + \int_{t_0}^{t_1} f_0(x(t), u(t), t) \, \mathrm{d}t, \tag{VII.2}$$

where the term $h(x(t_1))$ is a cost on the final state. Such a cost arises naturally in many applications. For example, the pilot of a hang glider in flight may wish to maximize its range, i.e., maximize the abscissa of its final position.

The optimal control problem may then be stated as

$$\begin{array}{ll} \underset{x,u}{\text{minimize}} & h(x(t_1)) + \int_{t_0}^{t_1} f_0(x(t),u(t),t)\,\mathrm{d}t \\ \text{subject to} & \dot{x}_i(t) = f_i(x(t),u(t),t), \quad i = 1,\dots,n, \\ & x_i(t_0) = x_i^0, \quad i = 1,\dots,n. \end{array} \tag{VII.3}$$

Example 7.1.1. Any variational calculus problem in which we minimize (VI.5) subject to the end-point condition $x(t_0) = x_0$ can be cast as an optimal control problem of the form (VII.3) by setting $u(t) = \dot{x}(t)$. □

Example 7.1.2. A unit mass is free to move along a straigt line under the action of a force $u(t)$ which we are free to choose at each instant t and which satisfies $-1 \le u(t) \le 1$. Let $x(t)$ and $\dot{x}(t)$ denote the position and velocity of the mass at time t. At time t_0, the mass is at location x_0 and has velocity $\dot{x}_0$. At time t_1, we wish to bring the mass as close as possible to location 0 with a velocity as close to 0 as possible. By letting $x_1(t) = x(t)$ and $x_2(t) = \dot{x}(t)$, the equations of motion become a system of first-order differential equations. The problem may thus be stated as

$$\begin{array}{ll} \underset{x_1,x_2,u}{\text{minimize}} & \frac{1}{2}\left(x_1(t_1)^2 + x_2(t_1)^2\right) \\ \text{subject to} & \dot{x}_1(t) = x_2(t), \quad x_1(t_0) = x_0, \\ & \dot{x}_2(t) = u(t), \quad x_2(t_0) = \dot{x}_0, \\ & -1 \le u(t) \le 1. \end{array}$$

In this example, $n = 2$, $f_0(x,u,t) = 0$, $h(x) = \frac{1}{2}(x_1^2 + x_2^2)$, $f_1(x,u,t) = x_2$, and $f_2(x,u,t) = u$. □

Example 7.1.3. An investor has a capital of $x_0 > 0$ at time 0 and decides to invest until time $T > 0$, e.g, one week. At time t, he is free to decide to re-invest a fraction $u(t) \in [0,1]$ of his capital. In an optimistic market, we assume that his return is proportional to his investment so that

$$\dot{x}(t) = \gamma u(t) x(t), \quad \text{where } \gamma > 0.$$

He keeps in store the amount that was not re-invested, i.e., $(1-u(t))x(t)$ and wishes to maximize the amount in store at all times. This problem can be written as

$$\begin{array}{ll} \underset{x,u}{\text{minimize}} & -\int_0^T (1-u(t))x(t)\,\mathrm{d}t \\ \text{subject to} & \dot{x}(t) = \gamma u(t) x(t), \\ & x(0) = x_0, \\ & 0 \le u(t) \le 1. \end{array}$$

Here, $n = 1$, $f_0(x,u,t) = (u-1)x$, $h(x) = 0$, and $f_1(x,u,t) = \gamma u x$. □

In the above examples, conditions are imposed on the control variables. These should not be seen as constraints on the optimal control problem but merely as a definition of the feasible set for the variables u. For instance, we may look for controls that are piecewise continuous and take values in the interval $[0,1]$. This restriction on the control will become important in the next section. In the sequel, we will say that u is a *feasible control* if each $u_i \in \mathcal{C}_{\text{PW}}([t_0,t_1])$ and u satisfies any additional condition that may have been imposed in the problem statement.

7.2 The minimum principle

We make the following assumptions.

Assumption 7.2.1. *The controls $u_1,\dots,u_m \in \mathcal{C}_{PW}([t_0,t_1])$. Moreover, the functions $f_0, f_1, \dots, f_n : \mathbb{R}^{n+m+1} \to \mathbb{R}$ are in $\mathcal{C}([t_0,t_1])$ and their partial derivatives with respect to each x_i are in $\mathcal{C}([t_0,t_1])$.*

We then have the following result.

Lemma 7.2.1. *Under Assumption 7.2.1, the solutions to the state equations with initial condition (VII.1), if any, are in $\mathcal{C}^1_{PW}([t_0,t_1])$.*

The resolution of (VII.3) happens through an auxiliary function called the *Hamiltonian*.

Definition 7.2.1. Given the optimal control problem (VII.3), the function $H : \mathbb{R}^{2n+m+1} \to \mathbb{R}$ defined by

$$H(x,u,p,t) = f_0(x,u,t) + \sum_{i=1}^{n} p_i f_i(x,u,t) \tag{VII.4}$$

is called the *Hamiltonian* of the problem.

A major difference between the Hamiltonian associated to variational calculus problems with isoperimetric constraints and (VII.4) is that the variables $p \in \mathbb{R}^n$ will take the values of *functions* $p(\cdot) \in \mathcal{C}^1_{\mathrm{PW}}([t_0,t_1])$. They no longer remain scalars but still play a role similar to that of Lagrange multipliers. These functions will be defined as the solutions to a set of differential equations with final-value called the *adjoint equations*.

Remark 7.2.1. Although (VII.4) has the general form of a Lagrangian in the optimization sense of the term, it is here called a Hamiltonian for the following reason. In Lagrangian mechanics, the minimization of the objective function of (VII.3) is related to the minimization of *action*, following Hamilton's principle, also called the *least action principle*. In that context, the function $f_0(x,u,t)$ is the Lagrangian of the system in the physical sense of the term, i.e, the difference between the kinetic energy and the potential energy.

The first-order necessary conditions for optimality of a set of control variables are summarized in Theorem 7.2.1, which is the main result of the present chapter.

Theorem 7.2.1 (Pontryagin's Minimum Principle)**.** *Under Assumption 7.2.1, assume $u^* \in \mathcal{C}_{PW}([t_0,t_1])^m$ is the optimal control and let $x^* \in \mathcal{C}^1_{PW}([t_0,t_1])^n$ be the corresponding optimal trajectory. Then there must exist functions $p_i^* \in \mathcal{C}^1_{PW}([t_0,t_1])$, $i = 1,\ldots,n$, solving the* adjoint equations

$$\dot{p}_i(t) = -\frac{\partial H}{\partial x_i}(x^*(t), u^*(t), p(t), t), \quad p_i(t_1) = \frac{\partial h}{\partial x_i}(x^*(t_1)), \quad i = 1,\ldots,n, \tag{VII.5}$$

such that

$$u^*(t) = \underset{u\in U}{\mathit{argmin}}\, H(x^*(t), u, p^*(t), t), \quad \textit{for all } t \in [t_0,t_1], \tag{VII.6}$$

where U is the set of feasible controls.

Moreover, if the system does not depend explicitly on t, i.e., if $\partial H/\partial t = 0$, then

$$H(x^*(t), u^*(t), p^*(t), t) = H(x^*(t_0), u^*(t_0), p^*(t_0), t_0), \quad t \in [t_0,t_1]. \tag{VII.7}$$

Theorem 7.2.1 may appear a bit awkward because it demands that (VII.5) and (VII.6) be solved, but their right-hand sides themselves depend upon the optimal control and trajectories. We will see in the examples to follow that these systems are rather simple and may be solved nevertheless. However, for more realistic problems, their analytical solution quickly becomes intractable. We now apply the minimum principle to a few examples.

Example 7.2.1. Let us consider once again our eternal example of the shortest path between two points on the plane. Let $(0, x_0)$ be the first point and the second point be located at abscissa $T > 0$. Following Example 7.1.1, we may state the problem as

$$\begin{array}{ll} \underset{x,u}{\text{minimize}} & \displaystyle\int_0^T \sqrt{1+u(t)^2}\,\mathrm{d}t \\ \text{subject to} & \dot{x}(t) = u(t), \\ & x(0) = x_0. \end{array}$$

We start by forming the Hamiltonian (VII.4):

$$H(x,u,p,t) = \sqrt{1+u^2} + pu.$$

The adjoint equations (VII.5) read

$$\dot{p}(t) = 0, \quad p(T) = 0,$$

from which we immediately conclude that $p^*(t) = 0$ for all $t \in [0,T]$. Thus (VII.6) becomes

$$u = \operatorname*{argmin}_u \sqrt{1+u^2} = 0.$$

In other words, the optimal control is in this case the function $u^*(t) = 0$ for all $t \in [0,T]$. From the state equations, this implies that $\dot{x}(t) = 0$ and, combined with the initial condition, $x(t) = x_0$ for all $t \in [0,T]$.

Note that since $\partial H/\partial t = 0$, Theorem 7.2.1 states that the Hamiltonian remains constant along the optimal trajectories. Such is the case here since $H(x^*(t), u^*(t), p^*(t), t) = 1$ for all $t \in [0,T]$. □

Example 7.2.2. Consider Example 7.1.3. The Hamiltonian is given by

$$H(x,u,p,t) = (u-1)x + \gamma pux = -x + (1+\gamma p)ux,$$

and the adjoint equations

$$\dot{p}(t) = 1 - (1+\gamma p(t))u^*(t), \quad p(T) = 0.$$

The unknown contol u^* now appears in the right-hand side, making this differential equation more difficult. However, from (VII.6), we know that u^* minimizes the Hamiltonian in the variable u, with all other variables fixed at their optimal value. Since $H(x,u,p,t)$ is linear in u, and $x(t) \geq 0$, we can say that

(a) $u^*(t) = 0$ for each t such that $1+\gamma p^*(t) > 0$, i.e., such that $p^*(t) > -1/\gamma$,

(b) $u^*(t) = 1$ for each t such that $1+\gamma p^*(t) \leq 0$, i.e., such that $p^*(t) \leq -1/\gamma$.

Therefore, we may solve the optimality conditions in two stages. We know from Theorem 7.2.1 that $p^* \in \mathcal{C}^1_{\mathrm{PW}}([0,T])$ and hence, continuous. From the final condition $p^*(T) = 0$, we deduce that there is $\epsilon > 0$ such that $p^*(t) > -1/\gamma$ for all $t \in]T-\epsilon, T]$. On the latter interval, we must therefore have $u^*(t) = 0$ and the adjoint equations become

$$\dot{p}(t) = 1, \quad t \in]T-\epsilon, T], \quad p(T) = 0.$$

The solution to this equation is $p^*(t) = t - T$. Since p^* is linear in that region and must remain continuous, we have $p^*(t) \geq -1/\gamma$ for all $t \in]T-1/\gamma, T]$. By continuity, $p^*(T-1/\gamma) = -1/\gamma$. Once t reaches $T - 1/\gamma$, a switch occurs and the optimal contol becomes $u^*(t) = 1$ and the adjoint equation turns into

$$\dot{p}(t) = -\gamma p^*(t), \quad t \in [0, T-1/\gamma].$$

The solution to the above equation is

$$p^*(t) = \alpha e^{-\gamma t},$$

for some constant $\alpha \in \mathbb{R}$. By imposing $p^*(T-1/\gamma) = -1/\gamma$, we determine that $\alpha = -e^{\gamma T - 1}/\gamma < 0$. We observe that $p^*(t) \leq -1\gamma$ for all $t \in [0, T-1/\gamma]$. In other words,

$$u^*(t) = \begin{cases} 1 & \text{if } t \in [0, T-1/\gamma], \\ 0 & \text{if } t \in]T-1/\gamma, T], \end{cases} \qquad p^*(t) = \begin{cases} -\frac{1}{\gamma} e^{\gamma(T-t)-1} & \text{if } t \in [0, T-1/\gamma], \\ t - T & \text{if } t \in]T-1/\gamma, T]. \end{cases}$$

The strategy of the investor is thus simple: re-invest everything up until $t = T - 1/\gamma$ and, after that point, do not re-invest anything. Note also that the higher the return γ, the longer the investor re-invests all his stock.

We are now in position to identify the optimal trajectory. On $[0, T - 1/\gamma]$, the state equation becomes $\dot{x}(t) = \gamma x(t)$, with the initial condition $x(0) = x_0$. On $]T - 1/\gamma, T]$, it becomes $\dot{x}(t) = 0$. By imposing continuity of x^*, we finally obtain

$$x^*(t) = \begin{cases} x_0 e^{\gamma t} & \text{if } t \in [0, T - 1/\gamma], \\ x_0 e^{\gamma T - 1} & \text{if } t \in]T - 1/\gamma, T]. \end{cases}$$

The solutions to this problem are represented in Fig. VII.1. □

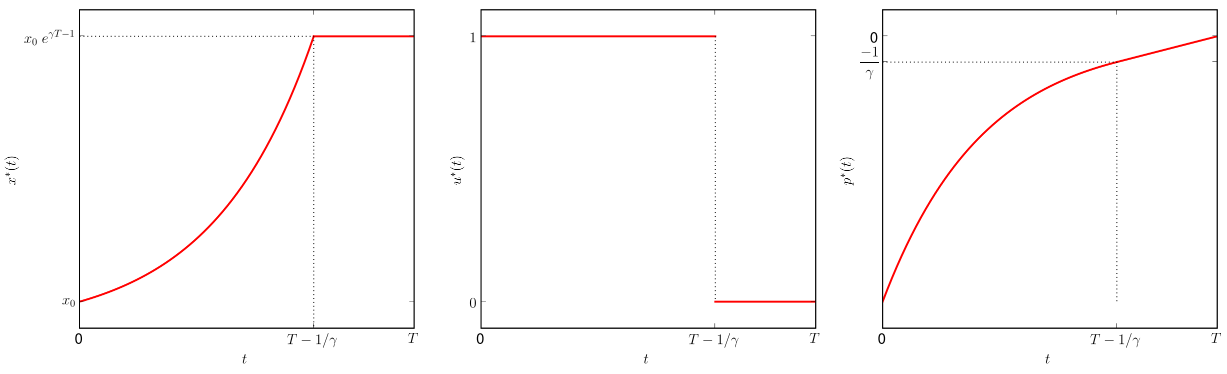


Figure VII.1: Optimal trajectory, control and adjoint states for Example 7.1.3.

7.3 Extensions to the Minimum Principle

An important special case of Problem (VII.3) occurs when $h(x) = 0$ and $f_0(x, u, t) = 1$, i.e., the objective of the problem becomes $t_1 - t_0$. For obvious reasons, when t_1 is free, this special case is called the *time-optimal control problem.* It aims to operate a dynamical system, e.g., from an initial condition to a final condition, in minimal time. Other variations are possible. The corresponding optimality conditions constitute an extention to Theorem 7.2.1 and are summarized in the next result.

Theorem 7.3.1 (Transversality Conditions in the Minimum Principle)**.** *Under the assumptions of Theorem 7.2.1, assume $u^* \in \mathcal{C}_{PW}([t_0, t_1])^m$ is the optimal control and let $x^* \in \mathcal{C}^1_{PW}([t_0, t_1])^n$ be the corresponding optimal trajectory. Then there must exist functions $p_i^* \in \mathcal{C}^1_{PW}([t_0, t_1])$, $i = 1, \dots, n$, solving the* adjoint equations

$$\dot{p}_i(t) = -\frac{\partial H}{\partial x_i}(x^*(t), u^*(t), p(t), t), \quad i = 1, \dots, n. \tag{VII.8}$$

In the system of adjoint equations, we have

(a) $p(t_1)$ is free if $x(t_1)$ is fixed,

(b) $p(t_0) = 0$ if $x(t_0)$ is free,

(c) if t_1 is free, then $H(x^*(t_0), u^*(t_0), p^*(t_0), t_0) = 0$.

The optimal control u^* is such that (VII.6) is satisfied. Moreover, if the system does not depend explicitly on t, i.e., if $\partial H/\partial t = 0$, then (VII.7) is satisfied.

A few comments on Theorem 7.3.1 are in order. Note first in Item (b) that we may extend our framework to incorporate a cost on the initial state, $g(x_0)$, in the objective function. In the same way that we had $p^*(t_1) = \nabla h(x(t_1))$ in (VII.5), we would have here $p^*(t_0) = -\nabla g(x(t_0))$. Finally, in Item (c), note that if $\partial H/\partial t = 0$, not only is the Hamiltonian constant along the optimal trajectories, but its value remains 0, i.e., $H(x^*(t), u^*(t), p^*(t), t) = 0$ for all $t \in [t_0, t_1]$.

Example 7.3.1. To illustrate Item (a), consider again Example 7.2.1 and assume that $x(T) = \beta$ is fixed. The adjoint equations then simply state that $p^*(t)$ remains constant, but do not specify its value. This is not a complication since it implies that the value of u that minimizes the Hamiltonian is always the same, i.e., $u^*(t)$ also remains constant. By the state equations, $x^*(t)$ is linear and we find its precise expression by imposing $x^*(0) = \alpha$ and $x^*(T) = \beta$. □

Example 7.3.2. A unit mass is free to move without friction along a straight line under the influence of a force $u(t)$. Given the initial position and speed $(x_0, \dot{x}_0)$ of the mass, we wish to bring it to position 0 at rest in minimal time, respecting $|u(t)| \leq 1$ for all t. The problem thus reads

$$\begin{array}{ll} \underset{x_1,x_2,u}{\text{minimize}} & T \\ \text{subject to} & \dot{x}_1(t) = x_2(t), \quad x_1(0) = x_0,\ x_1(T) = 0, \\ & \dot{x}_2(t) = u(t), \quad x_2(0) = \dot{x}_0,\ x_2(T) = 0. \end{array}$$

This is a typical example of a time-optimal control problem. The Hamiltonian is

$$H(x, u, p, t) = 1 + p_1 x_2 + p_2 u,$$

and the adjoint equations are

$$\dot{p}_1(t) = 0, \quad \dot{p}_2(t) = -p_1(t), \quad p_1(T), p_2(T) \text{ free}.$$

Therefore, there are constants $\alpha, \beta \in \mathbb{R}$ such that $p_1^*(t) = \alpha$ and $p_2^*(t) = -\alpha t + \beta$ for all $t \in [0, T]$. As in Example 7.1.3, we note that the Hamiltonian is linear in u and therefore,

(a) $u^*(t) = -1$ if $p_2(t) \geq 0$,

(b) $u^*(t) = +1$ if $p_2(t) < 0$.

Since p_2^* is itself linear, it will change sign at most once. Note that from Item (c) of Theorem 7.3.1, p_2^* cannot be identically zero. Therefore, the strategy is simple: we will initially select $u^*(0) = \pm 1$ depending on the initial conditions, and we will change the sign of u^* at most once during the whole process. From the equations of motion, we have

$$x_2(t) = x_2(0) + ut, \quad x_1(t) = x_1(0) + x_2(0)t + \tfrac{1}{2}ut^2,$$

and, by eliminating t,

$$x_1(t) - \frac{1}{2u} x_2(t)^2 = x_1(0) - \frac{1}{2u} x_2(0)^2, \quad \text{for all } t \in [0, T].$$

From this equation, we see that, in the phase plane, the system evolves along parabolas in a direction that is given by the sign of u, see Fig VII.2. Two of these parabolas go through $(0, 0)$, one along which x_2 increases and one along which x_2 decreases. The branches of the two latter parabolas that lead *towards* $(0, 0)$ constitute the *branching curve*.

From Fig. VII.2, we see that it is possible to bring the mass to $(0, 0)$ by changing u at most once. The strategy is to reach the branching curve and fork onto it in order to reach $(0, 0)$. If initially the mass is "located" above the branching curve, we use $u(t) = -1$ until we cross the blue section of the branching curve—at this point we switch to $u(t) = +1$ and follow the blue curve to $(0, 0)$. If initially, the mass is "located" under the branching curve, we first set $u(t) = +1$ and when we cross the red section of the branching curve, switch to $u(t) = -1$ to reach $(0, 0)$. □

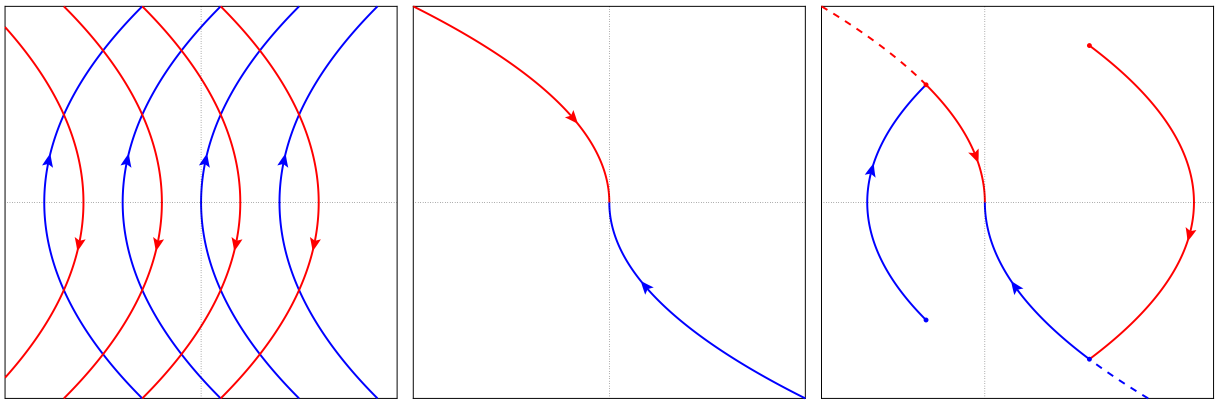


Figure VII.2: Phase plane for time-optimal control problem. Left: The system evolves along a red trajectory whenever $u(t) = -1$ and along a blue trajectory whenever $u(t) = +1$. Along red trajectories, x_2 decreases and along blue trajectories, x_2 increases. Center: Two particular branches lead to $(0, 0)$. They constitute the *branching curve.* Right: Path in phase plane followed by the system from an initial point to $(0, 0)$ by forking onto the branching curve, thereby switching u at most once.

Exercise 7.3.1. Solve the time-optimal control problem of Example 7.3.2 when $x(0) = -4$ and $\dot{x}(0) = 4$. Identify and plot the position and velocity as a function of time. Determine at what precise moment a switch in u should occur, if any. What is the travel time?

Exercise 7.3.2. By introducing a new state variable, show that the optimal control problem in the form (VII.3) can be rewritten in the form

$$\begin{array}{ll} \underset{x,u}{\text{minimize}} & \bar{h}(x(t_1)) \\ \text{subject to} & \dot{x}_i(t) = f_i(x(t), u(t), t), \quad i = 1, \dots, n+1, \\ & x_i(t_0) = x_i^0, \quad i = 1, \dots, n+1. \end{array} \tag{VII.9}$$

Problem (VII.3) is said to be a *Bolza problem* while Problem (VII.9) is said to be a *Mayer problem.*

Exercise 7.3.3. Explore again Example 7.3.2 but where this time we wish to minimize the energy spent for the mass to reach position 0 at rest, i.e.,

$$\begin{array}{ll} \underset{x_1,x_2,u}{\text{minimize}} & \displaystyle\int_0^T u(t)^2 \, \mathrm{d}t \\ \text{subject to} & \dot{x}_1(t) = x_2(t), \quad x_1(0) = x_0, \; x_1(T) = 0, \\ & \dot{x}_2(t) = u(t), \quad x_2(0) = \dot{x}_0, \; x_2(T) = 0. \end{array}$$

Choose initial conditions and determine the value of T.

Exercise 7.3.4. Explore Example 7.1.2.

7.4 Numerical Methods for the Optimal Control Problem

Chapter 8

Modeling

In this chapter we examine several problems, both academic and arising from real applications, formulate a mathematical model, implement it in the AMPL modeling language, solve it and study the solution.

8.1 Hanging Chain

This is a classical problem in which a chain of fixed length $L > 0$ is attached at both ends. The shape of the chain is represented by a function $x(t)$ defined and continuous on $[0, 1]$ and such that $x(0) = a$ and $x(1) = b$ with $a, b, \in \mathbb{R}$ given. Among all the functions satisfying these conditions, find the one describing the shape of the chain whose potential energy is minimal.

The problem is infinite-dimensional and can be written

$$\begin{array}{ll} \text{minimize} & \displaystyle\int_0^1 x(t)\sqrt{1+u(t)^2}dt \\ \text{subject to} & \displaystyle\int_0^1 \sqrt{1+u(t)^2}dt = L \\ & x(0) = a \\ & x(1) = b \\ & x'(t) = u(t). \end{array}$$

We discretize this problem to recover a finite-dimensional setting. The integrals and the differential equation are discretized using the trapeze rule. Using $N + 1$ discretization points and a step $h = 1/N$, we obtain

$$\begin{array}{ll} \text{minimize} & \displaystyle\frac{1}{2}h\sum_{i=0}^{N-1}\left(x_i\sqrt{1+u_i^2} + x_{i+1}\sqrt{1+u_{i+1}^2}\right) \\ \text{subject to} & \displaystyle\frac{1}{2}h\sum_{i=0}^{N-1}\left(\sqrt{1+u_i^2} + \sqrt{1+u_{i+1}^2}\right) \\ & x_0 = a \\ & x_N = b \\ & x_{i+1} - x_i = \frac{1}{2}h(u_i + u_{i+1}) \end{array}$$

Listing 8.1: AMPL model for the chain problem

```
# Hanging chain problem

model;

param nh;                   # number of subintervals
param L > 0;                # length of the suspended chain
param a;                    # height of the chain at t=0 (left)
param b;                    # height of the chain at t=1 (right)

param tf;                 # ODEs defined in [0,tf]
param h := tf/nh;         # uniform interval length

var x{0..nh};             # height of the chain
var u{0..nh};             # derivative of x

minimize potential_energy:
  0.5*h*sum {i in 0..nh-1} (x[i]*sqrt(1+u[i]^2)+x[i+1]*sqrt(1+u[i+1]^2));

subject to x_eqn {j in 0..nh-1}:
  x[j+1] = x[j] + 0.5*h*(u[j] + u[j+1]);

subject to length_eqn:
  0.5*h*sum {i in 0..nh-1} (sqrt(1+u[i]^2) + sqrt(1+u[i+1]^2)) = L;

# Boundary conditions

subject to x_bc1: x[0] = a;
subject to x_bc2: x[nh] = b;
```

Listing 8.2: AMPL data for the chain problem

```
# Hanging chain problem

model;

param tmin := if b > a then 0.25 else 0.75;

data;

param nh := 200;
param L := 4;
param a := 1;
param b := 3;
param tf := 1.0;

# Initial values

let {k in 0..nh} x[k] := 4*abs(b-a)*(k/nh)*(0.5*(k/nh) - tmin) + a;
let {k in 0..nh} u[k] := 4*abs(b-a)*((k/nh) - tmin);
```

8.1.1 Structure

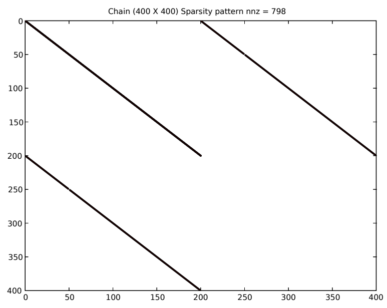
The objective and constraint functions have a special structure which can be represented by the sparsity patterns of their derivatives. The picture on the left in Fig. VIII.1 shows the sparsity pattern of the Hessian of the objective. In this pattern, a back dot represents a nonzero in the matrix for at least one feasible point. The empty space represents elements which are always equal to zero. Similarly, the picture on the right represents the sparsity pattern of the constraints Jacobian. These patterns show the inter-relationship between variables in the objective and constraints.

8.1.2 Solution

This is a problem with a nonlinear objective and nonlinear constraints. For instance, the KNITRO solver, available on the NEOS server, will be able to process it. Assuming that AMPL and KNITRO are installed locally and that the model and data files are `chain.mod` and `chain.dat` respectively, the following interactive session solves the problem and outputs the final values of the variables `x`.

We first read in the problem and pass it to KNITRO for solving.

```
ampl: model chain.mod;
ampl: data chain.dat;
ampl: option solver knitro;
ampl: solve;
```



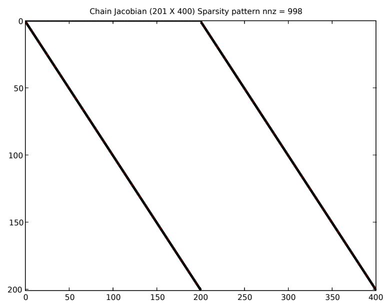


Figure VIII.1: Sparsity pattern of the Hessian matrix of the objective function (left) and of the Jacobian of the constraints (right) for the hanging chain problem.

```
KNITRO 3.0: 04/01/03

:
******************
*   KNITRO 3.0   *
******************

Nondefault Options
------------------

Problem Characteristics
-----------------------
Number of variables:                 400
    bounded below:                     0
    bounded above:                     0
    bounded below and above:           0
    fixed:                             0
    free:                            400
Number of constraints:               201
    linear equalities:               200
    nonlinear equalities:              1
    linear inequalities:               0
    nonlinear inequalities:            0
    range:                             0
Number of nonzeros in Jacobian:      999
Number of nonzeros in Hessian:       400

  nBar =  400, mBar =  201
 Iter   Res   Objective    Feas err  Opt err   ||Step||  CG its     mu
------  ---  ------------  --------  --------  --------  ------  --------
     0       4.742300E+00  1.19E+00
     1  Acc  6.037022E+00  2.27E-01  2.06E-03  2.00E+01       1  0.00E+00
     2  Acc  5.071128E+00  7.98E-02  3.11E-04  3.07E+01      22
     3  Acc  5.068763E+00  2.59E-02  6.00E-05  1.44E+01      23
     4  Acc  5.068908E+00  1.33E-03  9.15E-06  3.92E+00      29
     5  Acc  5.068917E+00  1.99E-05  3.04E-07  4.89E-01      29
     6  Acc  5.068917E+00  8.91E-09  1.58E-09  2.07E-02      37

EXIT: OPTIMAL SOLUTION FOUND.

Final Statistics
----------------
Final objective value               =     5.06891734171604E+00
Final feasibility error (abs / rel) =     8.91E-09 / 7.47E-09
Final optimality error  (abs / rel) =     1.58E-09 / 1.58E-09
# of iterations                     =          6
# of function evaluations           =          7
# of gradient evaluations           =          7
# of Hessian evaluations            =          6
Total program time (sec)            =       0.08

=========================================================================

KNITRO 3.0: OPTIMAL SOLUTION FOUND.
```

Knitro exits and claims that it found an “optimal” solution, i.e., one that satisfies the first-order optimality conditions. At this point, we recover the AMPL prompt. We next ask it to output the final values of the variables.

```
ampl: display x;

x [*] :=
  0 1            41 0.301043     82 0.154476    123 0.301145    164 1.00458
  1 0.974033     42 0.29312      83 0.154553    124 0.309362    165 1.0357
  2 0.944798     43 0.285481     84 0.154784    125 0.317878    166 1.0678
  3 0.916457     44 0.278118     85 0.155168    126 0.326701    167 1.10091
  4 0.888985     45 0.271024     86 0.155707    127 0.335838    168 1.13507
  ...            ...             ...            ...             ...
 34 0.365098     75 0.158237    116 0.251371    157 0.811837    198 2.84664
 35 0.354967     76 0.157235    117 0.257699    158 0.836865    199 2.93534
 36 0.345178     77 0.15639     118 0.264277    159 0.862686    200 3
 37 0.335721     78 0.155699    119 0.271112    160 0.889326
 38 0.326588     79 0.155163    120 0.278209    161 0.916809
 39 0.317769     80 0.15478     121 0.285576    162 0.94516
 40 0.309257     81 0.154551    122 0.293219    163 0.974408
;
```

8.1.3 Illustration

The function $x(t)$ is represented by its discretization, i.e., the vector $x \in \mathbb{R}^{N+1}$. The components of this vector are shown in Fig. VIII.2.

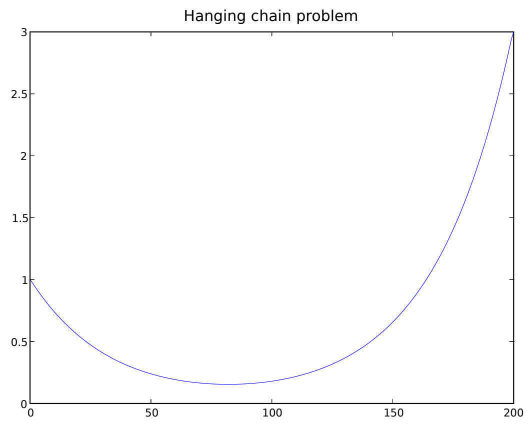


Figure VIII.2: Representation of the chain in its final state, as identified by Knitro

8.2 Largest Small Polygon

Among all polygons with N vertices and with diameter d less than 1, find the one with largest area. The diameter is the largest distance between any two vertices.

We model this problem in polar coordinates (r, θ). We fix the last vertex at $(0, \pi)$, i.e., at the origin. The vertices are ordered counter-clockwise and we measure angles counter-clockwise from the horizontal.

It is possible to show that the solution to this problem is a regular polygon for odd values of N and irregular for even values of N, except $N = 4$.

The problem with variables r_i and θ_i can be written

$$\begin{array}{lll}
\text{maximize} & \displaystyle\frac{1}{2}\sum_{i=1}^{N-1} r_i r_{i+1} \sin(\theta_{i+1} - \theta_i) & \\
\text{subject to} & \theta_{i+1} \geq \theta_i & i = 1, \ldots, N-1 \\
 & 0 \leq \theta_i \leq \pi & i = 1, \ldots, N \\
 & 0 \leq r_i \leq 1 & i = 1, \ldots, N \\
 & r_i^2 + r_j^2 - 2 r_i r_j \cos(\theta_j - \theta_i) \leq 1 & i = 1, \ldots, N \\
 & & j = i+1, \ldots, N \\
 & r_N = 0 & \\
 & \theta_N = \pi &
\end{array}$$

Listing 8.3: AMPL model for the largest small polygon problem

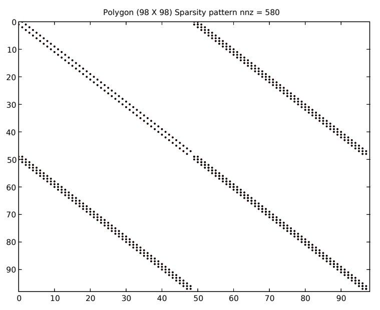
```
# Largest-small polygon problem

model;

param nv integer > 0;            # number of vertices in the polygon
param pi := 3.14159265358979;    # approximation of pi

var r {i in 1..nv};             # polar radius (distance to fixed vertex)
var theta {i in 1..nv};     # polar angle (measured from fixed direction)

maximize polygon_area:
  0.5*sum{i in 1..nv-1} r[i+1]*r[i]*sin(theta[i+1] - theta[i]);

subject to r_bounds {i in 1..nv}: 0.0 <= r[i] <= 1.0;

subject to theta_bounds {i in 1..nv}: 0.0 <= theta[i] <= pi;

subject to fix_theta_nv: theta[nv] = pi;
subject to fix_r_nv:     r[nv] = 0.0;

subject to ordered_theta {i in 1..nv-1}: theta[i] <= theta[i+1];

subject to distance {i in 1..nv-1,j in i+1..nv}:
  r[i]^2 + r[j]^2 - 2*r[i]*r[j]*cos(theta[j] - theta[i]) <= 1;

subject to convexity {i in 2..nv-1}:
  r[i]*r[i-1]*sin(theta[i]-theta[i-1]) + r[i+1]*r[i]*sin(theta[i+1]-theta[i]) >=
    r[i+1]*r[i-1]*sin(theta[i+1]-theta[i-1]);
```

Listing 8.4: AMPL data for the largest small polygon problem

```
# Largest-small polygon problem
data;

# Number of vertices
param nv := 8;

# Initial values

let {i in 1..nv-1} r[i] := 4*i*(nv + 1 - i)/(nv+1)^2;
let {i in 1..nv-1} theta[i] := pi*i/nv;
```

8.2.1 Structure

Again, we represent structure by means of sparsity patterns. These appear in Fig. VIII.3.



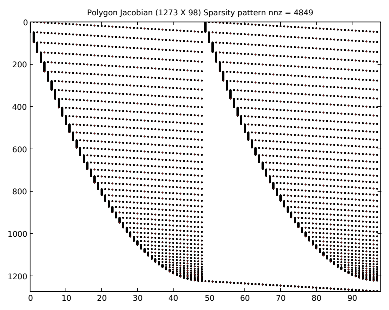


Figure VIII.3: Sparsity pattern of the Hessian of the objective (left) and of the Jacobian of the constraints (right) for the largest small polygon problem

8.2.2 Solution

We could solve this problem by means of the www form of the NEOS server. In this form, we enter `polygon.mod` and `polygon.dat` in the appropriate fields. In order to have AMPL perform additional tasks such as output the final values of the variables, we can insert a *command* file. For instace, the following file, named `polygon.ampl`, will display a formatted output with the final values of the variables `r` and `theta`.

```
printf {i in 1..nv} "%-g; ", r[i];
printf "\n";
printf {i in 1..nv} "%-g; ", theta[i];
#display {i in 1..nv-1,j in i+1..nv} distance[i,j];
```

To illustrate a different method, we use KESTREL to solve this problem. KESTREL is a “messenger” which carries the model from the user to the solver. All operations except for the solution happen locally. When handing the problem off to the solver, KESTREL establishes a network connection with the NEOS server and sends the model. It then waits until the solver replies with a message indicating that it has finished processing the model, and prints this message. In the command file `polygon.ampl`, we type

```
# Solve polygon problem using kestrel/Knitro

# Read the model
model polygon.mod;
data polygon.dat;

# Solve remotely using kestrel
option solver kestrel;

# Remote solver will be, e.g., Knitro
option kestrel_options 'solver=knitro';

# Solve !
solve;

# Look at the results
display r;
display theta;
```

Note that the “solver” is KESTREL and not the actual solver that will process the problem. The latter, KNITRO, is passed as an option to KESTREL. The above command file is executed from a command prompt with

```
ampl polygon.ampl
```

8.2.3 Illustration

For $N = 5$, the initial (irregular) and final polygons are shown in Fig. VIII.4. Note that the final polygon is regular.

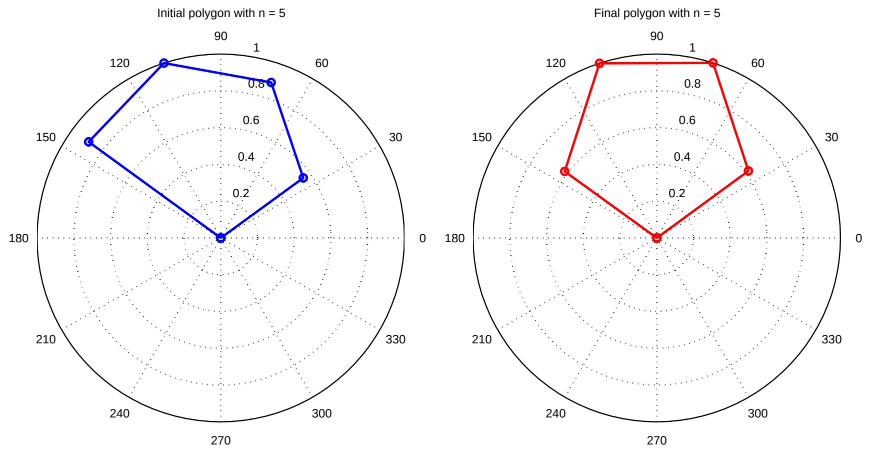


Figure VIII.4: On the left, the initial polygon and on the right, the final polygon

Appendix A

Background material

This chapter briefly reviews background material from analysis and linear algebra. It is not intended to be read linearly but should be used as a reference.

A.1 Topology

A.1.1 Inner products, norms and distances

In this text, we are only concerned with the vector spaces $\mathbb{R}^n$. An inner product on $\mathbb{R}^n$ is an application $\phi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ satisfying

(a) $\phi(x_1 + tx_2, y) = \phi(x_1, y) + t\phi(x_2, y)$,

(b) $\phi(x, y_1 + ty_2) = \phi(x, y_1) + t\phi(x, y_2)$,

(c) $\phi(x, y) = \phi(y, x)$.

We traditionally denote inner products by the symbol $\langle \cdot, \cdot \rangle$, i.e., $\langle x, y \rangle = \phi(x, y)$ for all $x, y \in \mathbb{R}^n$. The one we will use is defined by

$$\langle x, y, = \rangle \sum_{i=1}^{n} x_i y_i \quad \text{for all} \quad x, y \in \mathbb{R}^n.$$

A *norm* on $\mathbb{R}^n$ is an application $n : \mathbb{R}^n \to \mathbb{R}$ verifying the properties

(a) $n(x) \geq 0$ for all $x \in \mathbb{R}^n$,

(b) $n(x) = 0$ if and only if $x = 0$,

(c) $n(x + y) \leq n(x) + n(y)$,

(d) $n(tx) = |t| n(x)$ for all $x \in \mathbb{R}^n$ and all $t \in \mathbb{R}$.

The third of these properties is often referred to as the *triangle inequality.* Norms give a sense of the *length* of a vector. Given any inner product $\langle \cdot, \cdot \rangle$ on $\mathbb{R}^n$, we define a norm by $n(x) = \langle x, x \rangle$. Examples of norms include the well-known Euclidian norm, or ℓ_2-norm

$$\|x\|_2 = (x_1^2 + x_2^2 + \ldots + x_n^2)^{1/2} = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2},$$

the ℓ_1-norm

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

and the ℓ_∞-norm

$$\|x\|_\infty = \max(|x_1|, |x_2|, \dots, |x_n|).$$

Traditionally, norms are denoted $\|\cdot\|$ to signify that they generalize the notion of absolute value for the reals. All functions of the form

$$x \mapsto \|x\|_p = \left(\sum_{i=1}^n x_i^p \right)^{1/p}$$

are norms and are called p-norms or ℓ_p-norms. The ℓ_1 and ℓ_2 norms are obtained with $p = 2$ while the ℓ_∞ norm satisfies $\|x\|_\infty = \lim_{p\to\infty} \|x\|_p$.

The Cauchy-Schwartz inequality says that $|\langle x, y\rangle| \leq \|x\|\|y\|$ for all $x, y \in \mathbb{R}^n$, where the norm is defined by the inner product.

Another category of norms is norms given by a symmetric positive-definite matrix M and defined by

$$\|x\|_M = \langle x, Mx\rangle = x^T Mx.$$

They are sometimes referred to as *elliptical* norms because balls in this M-norm are ellipsoids defined by the eigenvectors of M. With $M = I$, we recover the Euclidian norm.

In $\mathbb{R}^n$, and in all finite-dimensional vector space, all norms are *equivalent*, i.e., if $\|\cdot\|_a$ and $\|\cdot\|_b$ are any two norms on $\mathbb{R}^n$, there exists a constant $\kappa > 0$ such that for all $x, y \in \mathbb{R}^n$,

$$\frac{1}{\kappa}\|x\|_a \leq \|x\|_b \leq \kappa\|x\|_a.$$

All norms induce a *distance* by which we can measure the distance between two points of $\mathbb{R}^n$. The distance between x and y is given by

$$\operatorname{dist}(x, y, =)\|y - x\|,$$

where $\|\cdot\|$ is any norm on $\mathbb{R}^n$.

A.1.2 Balls

Norms and distances induce the notion of neighbourhoods. We will only be concerned with specialized types of neighbourhoods called *balls*. Given $x \in \mathbb{R}^n$ and $\epsilon > 0$, the *ball* with center x and radius ϵ is

$$B(x, \epsilon) = \{z \in \mathbb{R}^n \mid \|z - x\| < \epsilon\}.$$

Each norm gives rise to balls of different "shape", were we to depict them in $\mathbb{R}^2$. While all norms coincide when $n = 1$, balls in the Euclidian norm are circles or spheres when $n = 2$ or $n = 3$, are diamonds in the ℓ_1-norm and rectangles or rectangular parallelipipeds in the ℓ_∞-norm. When $x = 0$ and $\epsilon = 1$, we refer to $B(0, 1)$ as the *unit* ball.

In $\mathbb{R}$, the notion of ball reduces to the notion of open interval, e.g., $B(x, \epsilon) =]x - \epsilon, x + \epsilon[$.

A.1.3 Open and closed sets

Let $A \subseteq \mathbb{R}^n$ be any set in $\mathbb{R}^n$. We say that $x \in A$ lies in the *interior* of A and we write $x \in \operatorname{int}(A)$ if there exists $\epsilon > 0$ such that $B(x, \epsilon) \subset A$. In other words, there is a ball centered at x which is entirely contained in A. The set of all points which are interior to A is called the interior of A and is denoted A° or $\operatorname{int}(A)$.

For intervals of $\mathbb{R}$, we deliberately adopt the French notation

(a) $[a,b] = \{x \in \mathbb{R} \mid a \le x \le b\}$,

(b) $]a,b] = \{x \in \mathbb{R} \mid a < x \le b\}$,

(c) $[a,b[= \{x \in \mathbb{R} \mid a \le x < b\}$,

(d) $]a,b[= \{x \in \mathbb{R} \mid a < x < b\}$,

in order to avoid confusion with the vector (a,b).

Examples of open sets in $\mathbb{R}^n$ are $\mathbb{R}^n$ itself, the empty set $\emptyset$, all open balls and cartesian products of open intervals $]a_1,b_1[\times]a_2,b_2[\times\ldots]a_k,b_k[$. Singletons $\{a\}$ are not open, the set $\{x \in \mathbb{R}^n \mid x_1 \ge 0,\ x_i > 0\ i = 2,\ldots,n\}$ is not open.

The set $A \subseteq$ is said to be *closed* if $\mathbb{R}^n \setminus A$ is open. If $\{x_k\}$ is any sequence of elements of A and if x^* is any limit point of $\{x_k\}$, closedness of A is equivalent to the property that $x^* \in A$. We usually say that a closed set is one that contains the limit points of its sequences. The *closure* of $A \subseteq \mathbb{R}^n$ is the smallest closed set B such that $A \subseteq B$. We have $B = A$ if and only if A is closed. The closure of A is denoted $\mathrm{cl}(A)$

$$\mathrm{cl}(A) = \{x \in \mathbb{R}^n \mid A \cap B(x,\epsilon) \neq \emptyset,\ \forall \epsilon > 0\}.$$

Examples of closed sets in $\mathbb{R}^n$ are $\mathbb{R}^n$ itself, the empty set $\emptyset$, singletons $\{a\}$, the closure of any set, the cartesian product of closed intervals $[a_1,b_1]\times[a_2,b_2]\times\ldots\times[a_k,b_k]$. The sets $B(x,\epsilon)$ and $\{x \in \mathbb{R}^n \mid x_1 \ge 0,\ x_i > 0\ i = 2,\ldots,n\}$ are not closed. The only sets in $\mathbb{R}^n$ that are both closed and open are $\mathbb{R}^n$ and $\emptyset$.

The finite intersection of open sets is open. The union of an arbitrary number of open sets is open. The intersetction of an arbitrary number of closed sets is closed. The finite union of closed sets is closed.

Since all norms on $\mathbb{R}^n$ are equivalent, all norms generate the same interior and closure of A.

The boundary ∂A of $A \subseteq \mathbb{R}^n$ is

$$\partial A = \mathrm{cl}(A) \setminus \mathrm{int}(A).$$

In $\mathbb{R}^n$ with $n \ge 2$, a line segment between two points a and b is a closed set but its interior is empty as the line segment cannot contain any open ball. However, it would seem reasonable to argue that the line segment resembles a closed interval of $\mathbb{R}$ and that as such, we could "open" it by removing the end points. This motivates the following definition. Let $A \subseteq \mathbb{R}^n$. The *affine hull* of A, denoted $\mathrm{aff}(A)$, is the smallest affine subspace of $\mathbb{R}^n$ containing A. The affine hull of the line segment example is the straight line supported by the segment. In other words, $\mathrm{aff}(A)$ is the intersection of all affine subspaces of $\mathbb{R}^n$ containing A. The interior of A when considered a subset of $\mathrm{aff}(A)$ instead of a subset of $\mathbb{R}^n$ is the *relative interior* of A

$$\mathrm{ri}(A) = \{x \in \mathbb{R}^n \mid \exists \epsilon > 0\ B(x,\epsilon) \cap \mathrm{aff}(A) \neq \emptyset\}.$$

With this definition we always have $\mathrm{ri}(A) \subseteq A \subseteq \mathrm{cl}(A)$.

The *relative boundary* of A is

$$\partial_{\mathrm{r}} A = \mathrm{cl}(A) \setminus \mathrm{ri}(A).$$

The relative boundary of the line segment are the two end points. If $\mathrm{int}(A) \neq \emptyset$, then $\mathrm{int}(A) = \mathrm{ri}(A)$ since then, $\mathrm{aff}(A) = \mathbb{R}^n$.

Note that

$$A_1 \subseteq A_2 \Longrightarrow \mathrm{cl}(A_1) \subseteq \mathrm{cl}(A_2) \quad \text{and} \quad \mathrm{int}(A_1) \subseteq \mathrm{int}(A_2),$$

but that

$$A_1 \subseteq A_2 \not\Longrightarrow \mathrm{ri}(A_1) \subseteq \mathrm{ri}(A_2)$$

since $\mathrm{ri}(A_1)$ and $\mathrm{ri}(A_2)$ may be disjoint. For instance, let $A_1 = \{0\} \times [0,1]$ and $A_2 = [0,1] \times [0,1]$ be two subsets of $\mathbb{R}^2$. We have $A_1 \subseteq A_2$ but $\mathrm{ri}(A_1) = \{0\}\times]0,1[$ and $\mathrm{ri}(A_2) =]0,1[\times]0,1[$ so that $\mathrm{ri}(A_1) \cap \mathrm{ri}(A_2) = \emptyset$.

When a set A is given as the level set of a function $f : \mathbb{R}^n \to \mathbb{R}$, e.g., $A = \{x \in \mathbb{R}^n \mid f(x) \leq \alpha\}$ where $\alpha \in \mathbb{R}$, or as an intersection of such sets

$$A = \{x \in \mathbb{R}^n \mid f_i(x) \leq \alpha_i, \; i = 1, \ldots, p\}$$

for some values $\alpha_i \in \mathbb{R}$, the *strict interior* of A is

$$\mathrm{strict}(A) = \{x \in \mathbb{R}^n \mid f_i(x) < \alpha_i, \; i = 1, \ldots, p\}.$$

The strict interior is not to be confused with the relative interior. Consider for instance the function $f : \mathbb{R} \to \mathbb{R}$ defined by

$$f(x) = \begin{cases} 0 & \text{if } x < 0, \\ (x-1)^2 - 1 & \text{if } 0 \leq x \leq 2, \\ 0 & \text{if } x > 2, \end{cases} \tag{I.1}$$

and the level set $A = \{x \in \mathbb{R} \mid f(x) \leq 0\} = \mathbb{R}$. Both the interior and the relative interior of A are equal to $\mathbb{R}$ but the strict interior of A is $\{x \in \mathbb{R} \mid f(x) < 0\} =]0,2[$.

A.1.4 Sequences and convergence

In this section, we review basic properties and real and vector sequences and convergence criteria.

A.1.4.1 Boundedness

Let $S \subseteq \mathbb{R}$ be a subset of $\mathbb{R}$. An upper bound on S is a real number M such that for all $s \in S$, $s \leq M$. If there is no such number M, we say that S is unbounded above. Otherwise, S is bounded above. If $S = \emptyset$, any $M \in \mathbb{R}$ is an upper bound on S. If M is an upper bound on S, any $M' \geq M$ is another upper bound on S.

A lower bound on S is a real number m such that for all $s \in S$, $m \leq s$. If there is no such number m, we say that S is unbounded below. Otherwise, S is bounded below. If $S = \emptyset$, any $m \in R$ is a lower bound on S. If m is a lower bound on S, any $m' \leq m$ is another lower bound on S.

We say that S is *bounded* if it is bounded above and below.

If S is bounded above, the smallest upper bound on S always exists and is called the *supremum* of S, denoted $\sup(S)$. If S is unbounded above, we define $\sup(S) = +\infty$. Note that $\sup(S)$ need not be an element of S. If it so happens that $\sup(S) \in S$, we say that the supremum is attained, we call it the *maximum* of S and denote it $\max(S)$. If $\sup(S) \notin S$, the maximum of S does not exist. If $S = \emptyset$, we let $\sup(S) = -\infty$.

If S is bounded below, the largest lower bound on S always exists and is called the *infimum* of S, denoted $\inf(S)$. If S is unbounded below, we define $\inf(S) = -\infty$. Again, $\inf(S)$ need not be an element of S. When $\inf(S) \in S$, the infimum is attained, we call it the *minimum* of S and denote it $\min(S)$. If $\inf(S) \notin S$, the minimum of S does not exist. If $S = \emptyset$, we let $\inf(S) = +\infty$.

For example, if $S = [0,1[$, all real numbers $M \geq 1$ are upper bounds on S, all $m \leq 0$ are lower bounds on S. We have $\sup(S) = 1 \notin S$, $\max(S)$ does not exist, $\inf(S) = \min(S) = 0 \in S$.

A.1.4.2 Sequences in $\mathbb{R}$ and $\mathbb{R}^n$

All definitions in this section equally apply to sequences in $\mathbb{R}$ by replacing norms with absolute values.

A sequence in $\mathbb{R}^n$ is a set of elements of $\mathbb{R}^n$ indexed by $\mathbb{N}$, the set of nonnegative integers. It is usually denoted $\{x_k\}_{k\in\mathbb{N}}$, $\{x_k\}_{k\geq 0}$, $\{x_0, x_1, \dots, x_k, \dots\}$ or, if there is no possible confusion, simply $\{x_k\}$. If $n = 1$, we say that $\{x_k\}$ is an integer sequence if $x_k \in \mathbb{N}$ for all k.

When $n = 1$, the sequence is *nondecreasing* if $x_{k_1} \leq x_{k_2}$ whenever $k_1 \leq k_2$, it is *nonincreasing* if $x_{k_1} \geq x_{k_2}$ whenever $k_1 \leq k_2$, it is *increasing* if $x_{k_1} < x_{k_2}$ whenever $k_1 < k_2$ and it is *decreasing* if if $x_{k_1} > x_{k_2}$ whenever $k_1 < k_2$. In $\mathbb{R}^n$ with $n \geq 2$, there is no notion of increasing or decreasing sequences. However, if $\{x_k\}$ is a sequence in $\mathbb{R}^n$, the real sequence $\{\|x_k\|\}$ may be (non)increasing or (non)decreasing.

The real sequences $\{1/k\}$, $\{1/k!\}$ and $\{2^{-k}\}$ are decreasing, $\{k\}$, $\{2^k\}$ and $\{k!\}$ are increasing, the sequence $\{max(1/10, 1/k)\}$ is nonincreasing. The sequence $\{(-1)^k\}$ is neither (non)increasing nor (non)decreasing. In $\mathbb{R}^2$, the sequence $\{(1, k)\}$ is increasing in norm.

A *subsequence* of $\{x_k\}$ is a subset of the sequence indexed by $\mathcal{K} \subseteq \mathbb{N}$ such that the indices in $\mathcal{K}$ are increasing, and it is denoted $\{x_k\}_{k\in\mathcal{K}}$. Sometimes, a subsequence of $\{x_k\}_{k\in\mathbb{N}}$ is denoted $\{x_{q_k}\}_{k\in\mathbb{N}}$ where $\{q_k\}_{k\in\mathbb{N}}$ is an increasing integer sequence. Possibly, a sequence or subsequence is finite. Possibly, it is only countable.

The notions of boundedness apply in particular when S is a sequence $\{x_k\}$ in $\mathbb{R}$ or a sequence of the form $\{\|x_k\|\}$ where $\{x_k\}$ is a sequence in $\mathbb{R}^n$.

A.1.4.3 Convergence

If $\{x_k\}$ is a sequence in $\mathbb{R}^n$ and $x^* \in \mathbb{R}^n$, the sequence *converges* to x^* if

$$\forall \epsilon > 0,\ \exists k_\epsilon \text{ such that } \forall k \geq k_\epsilon,\ x_k \in B(x^*, \epsilon).$$

In other words, for any ball centered at x^*, the elements of the sequence $\{x_k\}$ end up falling inside the ball and never leaving it again. Since we can choose $\epsilon > 0$ smaller and smaller, the sequence has no other choice but get closer and closer to x^*. If $\{x_k\}$ converges to x^*, we say that it is a convergent sequence, that x^* is the limit of $\{x_k\}$ and we write

$$\{x_k\} \to x^* \quad \text{or} \quad \lim_{k\to\infty} x_k = x^*.$$

Note that if $\{x_k\}$ converges to x^*, it cannot converge to any other limit. The limit, if it exists, is unique. Note also that the definition of convergence could equivalently be written

$$\forall \epsilon > 0,\ \exists k_\epsilon \text{ such that } \forall k \geq k_\epsilon,\ \|x_k - x^*\| < \epsilon.$$

The sequences $\{1/k\}$, $\{1/k!\}$ and $\{2^{-k}\}$ all converge to 0. The sequences $\{(-1)^k\}$, $\{k\}$ and $\{2^k\}$ are not convergent. The sequence $\{x_k\}$ where $x_k = \sum_{i=0}^{k} 1/i!$ converges to e.

It may be that $\{x_k\}$ is not a convergent sequence but that it is nevertheless possible to extract a convergent subsequence from $\{x_k\}$. For instance $\{(-1)^k\}$ is not convergent but if we only select every other term, starting with $k = 0$, i.e., we choose $q_k = 2k$, we obtain the subsequence $\{(-1)^{2k}\}$, i.e., the constant subsequence $x_{q_k} = 1$ for all k. This subsequence converges to 1. Similarly, if we choose $q_k = 2k+1$ we obtain the constant subsequence $x_{q_k} = -1$ for all k, which converges to -1. Subsequences need not be constant to converge.

More generally, if $\{x_k\}$ is bounded, it necessarily admits a convergent subsequence. This result does not give the subsequence, but only states that there is one. If x^* is the limit of a convergent subsequence $\{x_{q_k}\}$ of $\{x_k\}$, x^* is called a *limit point*, or *cluster point*, of $\{x_k\}$. The sequence $\{(-1)^k\}$ is not convergent, but has two limit points: -1 and 1.

In $\mathbb{R}$ a sequence that is bounded below and nonincreasing always converges. Its limit is its infimum. Similarly, a sequence that is bounded above and nondecreasing always converges to its supremum. A decreasing sequence that is not bounded below does not, strictly speaking, converge. We abuse the definition and say that it converges to $-\infty$. An increasing sequence that is not bounded above is said, similarly, to converge to $+\infty$.

In $\mathbb{R}$, given a sequence $\{x_k\}$ the sequence

$$y_k = \inf \{x_j \mid j > k\}$$

is nondecreasing. Its limit, if it exists, is called the *lower limit* of $\{x_k\}$ and is denoted $\liminf_{k\to\infty} x_k$. If this limit does not exist, we let $\liminf_{k\to\infty} x_k = -\infty$. Similarly, the sequence

$$z_k = \sup \{x_j \mid j > k\}$$

is nonincreasing. Its limit, if it exists, is called the *upper limit* of $\{x_k\}$ and is denoted $\limsup_{k\to\infty} x_k$. If this limit does not exist, we let $\limsup_{k\to\infty} x_k = +\infty$.

A sequence in $\mathbb{R}$ is convergent if and only if its lower and upper limits exist and are equal. In this case we have

$$\liminf_{k\to\infty} x_k = \limsup_{k\to\infty} x_k = \lim_{k\to\infty} x_k.$$

A sequence $\{x_k\}$ in $\mathbb{R}^n$ is called a *Cauchy sequence* if

$$\forall \epsilon > 0 \ \exists k_\epsilon \text{ such that } \forall p, q \geq k_\epsilon \ \|x_p - x_q\| < \epsilon.$$

The definition of a Cauchy sequence resembles that of a convergent sequence but does not involve any fixed vector such as x^*. It specifies that the distance between elements of the sequence shrinks as we progress along the sequence. The interest in Cauchy sequences is that

$$\{x_k\} \text{ converges } \iff \{x_k\} \text{ is a Cauchy sequence.}$$

A *metric space* is a vector space equipped with a norm or distance function. For instance, $(\mathbb{R}, |\cdot|)$, $(\mathbb{R}^n, \|\cdot\|_2)$ are metric spaces. This last equivalence is a major result which only holds in so-called *complete* metric spaces, such as $\mathbb{R}^n$ equipped with any norm $\|\cdot\|$. By definition, a metric space is complete if Cauchy sequences converge. An example of a metric space which is not complete is $\mathbb{Q}$, the set of rational numbers equipped with the absolute value. The rational sequence defined by

$$x_0 = 1, \quad x_{k+1} = x_k/2 + 1/x_k \quad \forall k \geq 0$$

is a Cauchy sequence but is not convergent. Indeed, if $\{x_k\}$ were considered a sequence in $\mathbb{R}$, we would have $\{x_k\} \to \sqrt{2}$. However, $\sqrt{2} \notin \mathbb{Q}$. In general, all finite-dimensional metric spaces are complete and convergence can be established by verifying the Cauchy condition. Difficulties arise in the infinite-dimensional case.

A.1.4.4 Landau notation for sequences

Let $\{x_k\}$ and $\{y_k\}$ be two sequences of positive real numbers.

We say that x_k is asymptotically bounded above by y_k if there exists a constant $\kappa > 0$ and $k_0 \in \mathbb{N}$ such that $x_k \leq \kappa y_k$ for all $k \geq k_0$. In this case we write $x_k = O(y_k)$ and we say that x_k is a big 'O' of y_k.

If $x_k = O(y_k)$ and $y_k = O(x_k)$ we say that the two sequences are asymptotically similar and we write $x_k = \Theta(y_k)$ (or $y_k = \Theta(x_k)$). We say that x_k is a "theta" of y_k.

Assume $\{x_k\} \to 0$ and $\{y_k\} \to 0$. If the ratio x_k/y_k also converges to zero, we say that x_k is asymptotically smaller than y_k and we write $x_k = o(y_k)$. We say that x_k is a little 'o' of y_k.

A.1.4.5 Rate of convergence

Let $\{x_k\}$ be a convergent sequence in $\mathbb{R}^n$ and let x^* denote its limit.

The sequence $\{x_k\}$ converges *Q-linearly* to x^* if there is a constant $\kappa \in]0,1[$ and $k_0 \in \mathbb{N}$ such that

$$\|x_{k+1} - x^*\| \leq \kappa \|x_k - x^*\| \quad \text{for all} \quad k \geq k_0.$$

This definition depends on the norm chosen since we require $\kappa \in]0,1[$.

The sequence $\{x_k\}$ converges *Q-superlinearly* to x^* if

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0,$$

i.e., if $\|x_{k+1} - x^*\| = o(\|x_k - x^*\|)$. This definition is independent of the norm.

The sequence $\{x_k\}$ converges with Q-rate of $\alpha > 1$ to x^* if $\|x_{k+1} - x^*\| = O(\|x_k - x^*\|^\alpha)$. This definition is independent of the norm.

The sequence $\{x_k\}$ converges *Q-quadratically* to x^* if there exists $\kappa > 0$ and $k_0 \in \mathbb{N}$ such that

$$\|x_{k+1} - x^*\| \leq \kappa \|x_k - x^*\|^2 \quad \text{for all} \quad k \geq k_0,$$

i.e., if $\|x_{k+1} - x^*\| = O(\|x_k - x^*\|^2)$. This definition is independent of the norm.

A different but related notion of convergence rate if that of R-rate of convergence. The sequence $\{x_k\}$ converges to x^* with R-rate p if the sequence $\{\|x_k - x^*\|\}$ is bounded above by a sequence converging to zero with a Q-rate of p.

A.2 Approximation of functions

A.2.1 Landau notation for functions

We use the following symbols, sometimes known as the *Landau* symbols. Let $\phi, \psi : \mathbb{R} \to \mathbb{R}$ be two functions of one variable such that there is an x^* for which $\phi(x^*) = \psi(x^*) = 0$. The function ψ is said to be a *little 'o'* of ϕ, denoted $\psi = o(\phi)$ if

$$\lim_{x\to x^*} \frac{\psi(x)}{\phi(x)} = 0,$$

i.e., if ψ converges to zero faster than ϕ as $x \to x^*$. Often, we abuse this notation and simply write

$$\lim_{\phi\to 0} \frac{o(\phi)}{\phi} = 0.$$

A.2.2 Continuity

A function $f : D \subset \mathbb{R}^n \to \mathbb{R}^m$ with domain D is *continuous* at $x^* \in D$ if

$$\forall \epsilon > 0 \; \exists \delta > 0 \quad x \in B(x^*, \delta) \Rightarrow f(x) \in B(f(x^*), \epsilon).$$

Note that this definition can be rewritten

$$\forall \epsilon > 0 \; \exists \delta > 0 \quad \|x - x^*\| < \delta \Rightarrow \|f(x) - f(x^*)\| < \epsilon.$$

In other words, points that are close to each other have images that are close to each other. Continuity is also described in terms of limits. The function f is continuous if and only if

$$\lim_{x \to x^*} f(x) = f(x^*).$$

In particular, this equality holds for all convergent sequences. In this case, the limit of the images is the image of the limit.

The function f is continuous if it is continuous at every point $x \in D$ of its domain.

For example, the function (I.1) is continuous at any $x^* \in \mathbb{R}$. The function $\log(x)$ is not continuous at $x = 0$. The function

$$f(x) = \begin{cases} -1 & \text{if } x < 0, \\ 1 & \text{if } x \geq 0 \end{cases}$$

is not continous at $x = 0$.

A.2.3 Differentiability

A.2.3.1 First-order derivatives

A function $f : D \subset \mathbb{R}^n \to \mathbb{R}$ defined on an open set D is *differentiable* at $x^* \in D$ if there exists an affine function $d_{x^*}(y)$ such that for all $y \in D$,

$$\lim_{y \to x^*} \frac{f(y) - f(x^*) - d_{x^*}(y)}{\|y - x^*\|} = 0.$$

If it exists, d_{x^*} is unique and is called the *differential* of f at x^*. Moreover, the function

$$y \mapsto f(y) - f(x^*) - d_{x^*}(y)$$

is a $o(\|y - x^*\|)$. If f is differentiable at x^*, it is necessarily continuous at x^*. The affine function d_{x^*} is sometimes called the *total derivative* of f at x^*.

If f is differentiable at $x^* \in D$, all the partial derivatives of f, $\partial f / \partial x_i$ $(i = 1, \ldots, n)$, exist at x^*. We gather them into a vector called the *gradient* of f at x^*

$$\nabla f(x^*) = \begin{bmatrix} \partial f(x^*)/\partial x_1 \\ \partial f(x^*)/\partial x_2 \\ \vdots \\ \partial f(x^*)/\partial x_n \end{bmatrix}.$$

When $n = 1$, the gradient reduces to the well-known derivative.

The symbol ∇ is often used to refer to first-order derivatives. It is pronounced *nabla* or *grad*.

Note that the gradient of f can itself be considered a function from $\mathbb{R}^n$ into $\mathbb{R}^n$.

The differential function above can be shown to be $d_{x^*}(y) = \nabla f(x^*)^T (y - x^*)$ for all $y \in \mathbb{R}^n$.

If $F : \mathbb{R}^n \to \mathbb{R}^m$ is a vector-valued function, we can write

$$F(x) = (F_1(x), F_2(x), \ldots, F_m(x)).$$

There is a similar definition of differentiability. The object that corresponds to the gradient in this case is an $m \times n$ matrix called the *Jacobian matrix*, often noted $J_F(x)$, or $J(x)$ if there is no possible confusion, and is defined by

$$J_F(x) = \begin{bmatrix} \partial F_1(x)/\partial x_1 & \partial F_1(x)/\partial x_2 & \dots & \partial F_1(x)/\partial x_n \\ \partial F_2(x)/\partial x_1 & \partial F_2(x)/\partial x_2 & \dots & \partial F_2(x)/\partial x_n \\ \vdots & \vdots & & \vdots \\ \partial F_m(x)/\partial x_1 & \partial F_m(x)/\partial x_2 & \dots & \partial F_m(x)/\partial x_n \end{bmatrix} = \begin{bmatrix} \nabla F_1(x)^T \\ \nabla F_2(x)^T \\ \vdots \\ \nabla F_m(x)^T \end{bmatrix}.$$

A.2.3.2 The chain rule for first-order derivatives

If $f : D \subseteq \mathbb{R}^n \to \mathbb{R}^m$ is differentiable at $x \in D$, if $g : E \subseteq \mathbb{R}^m \to \mathbb{R}^p$, if $f(x) \in E$ and if g is differentiable at $f(x)$, the composition $h = g \circ f : D \subseteq \mathbb{R}^n \to \mathbb{R}^p$ defined by $h(x) = g(f(x))$ for all $x \in D$ such that $f(x) \in E$ is differentiable at x and we have

$$J_h(x) = J_g(f(x))J_f(x).$$

In particular if $m = p = 1$, we obtain

$$\nabla h(x) = g'(f(x))\nabla f(x).$$

As another important special case, suppose f is an affine function of the form $Ax + b$ where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, then $h(x) = g(Ax + b)$ and

$$\nabla h(x) = A^T \nabla g(Ax + b).$$

A.2.3.3 Second-order derivatives

An instance of such a vector-valued function is the gradient of a scalar function $f : \mathbb{R}^n \to \mathbb{R}$. If it exists, its Jacobian matrix is called the *Hessian* matrix of f. By identification with the above Jacobian matrix, we have $F_i(x) = \partial f(x)/\partial x_i$ and therefore $\partial F_i(x)/\partial x_j = \partial^2 f(x)/\partial x_i \partial x_j$. The Hessian matrix of f at x thus gathers all partial second-order derivatives of f at x into a square $n \times n$ matrix

$$\nabla^2 f(x) = \begin{bmatrix} \partial^2 f(x)/\partial x_1^2 & \partial^2 f(x)/\partial x_1 \partial x_2 & \dots & \partial^2 f(x)/\partial x_1 \partial x_n \\ \partial^2 f(x)/\partial x_2 \partial x_1 & \partial^2 f(x)/\partial x_2^2 & \dots & \partial^2 f(x)/\partial x_2 \partial x_n \\ \vdots & \vdots & & \vdots \\ \partial^2 f(x)/\partial x_n \partial x_1 & \partial^2 f(x)/\partial x_n \partial x_2 & \dots & \partial^2 f(x)/\partial x_n^2 \end{bmatrix}.$$

In the special case where all the partial second-order derivatives of f are continuous functions of x, Schwarz's theorem says that for all $i, j = 1, \dots, n$, we have

$$\partial^2 f(x)/\partial x_i \partial x_j = \partial^2 f(x)/\partial x_j \partial x_i$$

and therefore the Hessian matrix $\nabla^2 f(x)$ is symmetric.

A.2.3.4 The chain rule for second-order derivatives

If $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ is twice differentiable at $x \in D$, if $g : E \subseteq \mathbb{R} \to \mathbb{R}$, if $f(x) \in E$ and if g is twice differentiable at $f(x)$, the composition $h = g \circ f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ defined by $h(x) = g(f(x))$ for all $x \in D$ such that $f(x) \in E$ is twice differentiable at x and we have

$$\nabla^2 h(x) = g'(f(x))\nabla^2 f(x) + g''(f(x))\nabla f(x)\nabla f(x)^T.$$

Note that the last term, $\nabla f(x)\nabla f(x)^T$ is an *outer product* of a column vector by a row vector, to produce a rank-1 $n \times n$ matrix.

If f is an affine function of the form $Ax + b$ where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, then $h(x) = g(Ax + b)$ and

$$\nabla^2 h(x) = A^T \nabla^2 g(Ax + b)A.$$

A.2.3.5 Restriction of a function to a direction, directional derivative

Given $f : \mathbb{R}^n \to \mathbb{R}$, a point $x \in \mathbb{R}^n$ and a nonzero direction $d \in \mathbb{R}^n$, the function of one variable

$$\phi : \mathbb{R} \to \mathbb{R}, \quad \phi(t) = f(x + td)$$

is called the restriction of f to the direction d. Its graph is a "slice" of the graph of f (were we able to plot it) along the direction d. If f is differentiable around x, the chain rule yields that ϕ is differentiable for values of t around 0 and

$$\phi'(t) = \nabla f(x + td)^T d.$$

In particular, $\phi'(0) = \nabla f(x)^T d$ is the *directional derivative* of f at x in the direction d.

Even if f is not differentiable at x it may be that the limit

$$\lim_{t\downarrow 0} \frac{\phi(t) - \phi(0)}{t}$$

is well defined and finite. If such is the case, the value of the limit is called the *directional derivative* of f at x in the direction d and is denoted $f'(x; d)$. If f is differentiable at x, all directional derivatives of f at x exist and for all $d \in \mathbb{R}^n$, $f'(x; d) = \nabla f(x)^T d$.

For example $f(x) = |x|$ is not differentiable at $x = 0$ but its directional derivative in the directions -1 and $+1$ is respectively, -1 and $+1$.

Similarly, if f is twice differentiable around x,

$$\phi''(t) = d^T \nabla^2 f(x + td)d$$

and $\phi''(0) = d^T \nabla^2 f(x)d$ is the *curvature* of f at x in the direction d. If $\phi''(0) > 0$, we say that d is a direction of positive curvature, if $\phi''(0) < 0$, we say that d is a direction of negative curvature. If $\phi''(0) = 0$ then, up to first order, f is linear in the direction d.

For example, if $f(x, y) = x^2 - y^2$, $(1, 0)$ is a direction of positive curvature at $(0, 0)$ while $(0, 1)$ is a direction of negative curvature at $(0, 0)$. If $f(x, y) = x^2 + y$, $(0, 1)$ is a direction of "zero curvature".

A.2.4 Expansions

Let I be an open domain of R^n or R^n itself. We say that a function $f : I \subseteq \mathbb{R}^n \to \mathbb{R}$ is continuous on I if it is continuous at each point of I. In this case, we write $f \in \mathcal{C}^0(I)$ or $f \in \mathcal{C}(I)$.

Similarly, f is differentiable on I if it is differentiable at each point of I.

It is continuously differentiable if $\nabla f(x)$ exists and is a continuous function of x. In this case, we write $f \in \mathcal{C}^1(I)$.

It is twice continously differentiable if it is continously differentiable, $\nabla^2 f(x)$ exists and is a continous function of x. In this case, we write $f \in \mathcal{C}^2(I)$.

There are similar definition for derivatives of order $p \in \mathbb{N}$ and we write $f \in \mathcal{C}^p(I)$.

If all successive derivatives of f over I exist and are continous functions, we write $f \in \mathcal{C}^\infty(I)$.

If $D \subset \mathbb{R}^n$ is not an open set, we use the notation $f \in \mathcal{C}^p(D)$ to mean that there exists an open set $I \subseteq \mathbb{R}^n$ such that $D \subset I$ and $f \in \mathcal{C}^p(I)$.

We recall the following two theorems for their importance and constant use in nonlinear optimization.

Theorem A.2.1 (Mean-value theorem)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ and $x, y \in \mathbb{R}^n$. Let $[x, y]$ denote the segment joining x and y. Assume $f \in \mathcal{C}^1([x, y])$. Then*

$$f(y) = f(x) + \int_0^1 \nabla f(x + t(y - x))^T (y - x) dt.$$

If in addition $f \in \mathcal{C}^2([x, y])$, then

$$\nabla f(y) = \nabla f(x) + \int_0^1 \nabla^2 f(x + t(y - x))(y - x) dt.$$

A corollary to the above result is the well-known Taylor theorem

Theorem A.2.2 (Taylor's theorem)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ and $x \in \mathbb{R}^n$. Assume $f \in \mathcal{C}^1(I)$ for some open set I such that $x \in I$. Then for all y sufficiently close to x,*

$$f(y) = f(x) + \nabla f(x)^T (y - x) + o(\|y - x\|).$$

If in addition $f \in \mathcal{C}^2(I)$ for some open set I such that $x \in I$. Then for all y sufficiently close to x,

$$f(y) = f(x) + \nabla f(x)^T (y - x) + \frac{1}{2!}(y - x)^T \nabla^2 f(x)(y - x) + o(\|y - x\|^2).$$

A.2.5 The implicit function theorem

Under certain regularity conditions, if (x^*, y^*) is a solution to an equation of the form $f(x, y) = 0$, it is locally possible to express y as a function of x in the vicinity of (x^*, y^*). The implicit function theorem is largely used in nonlinear optimization and especially in the theory of interior-point methods. When studying systems of nonlinear equations, it is used in most path-following methods.

Theorem A.2.3 (Implicit function theorem)**.** *Let $f : D \subset \mathbb{R}^{n+m} \to \mathbb{R}^n$ be $\mathcal{C}^1(D)$ where D is an open set such that there exists $(a, b) \in D$ satisfying $f(a, b) = 0$. Assume the Jacobian matrix $J_x(a, b)$ of f with respect to x is nonsingular. Then there exist $\epsilon, \delta > 0$ such that*

(a) for all $y \in B(b, \delta)$, there is a unique $x = x(y) \in \mathbb{R}^{n+m}$ such that $(x(y), y) \in B(a, \epsilon)$ and $f(x(y), y) = 0$.

(b) as a function of y, $x(\cdot) : B(b, \delta) \to \mathbb{R}^n$ is $\mathcal{C}^1(B(b, \delta))$. Moreover its gradient is given by

$$\nabla x(y) = -J_x((x(y), y)^{-1} J_y(x(y), y)$$

for all $y \in B(b, \delta)$.

A.3 Linear algebra

A.3.1 Matrices

We denote by $\mathbb{R}^{n\times m}$ the nm-dimensional vector space of $n \times m$ matrices with real coefficients. We usually denote matrices by uppercase latin letters such as A, B, C, etc.

The element at the intersection of the i-th row and j-th column of a matrix A is A_{ij}. The *tranpose* of $A \in \mathbb{R}^{n\times m}$ is a matrix in $\mathbb{R}^{m\times n}$ denoted A^T such that $(A^T)_{ij} = A_{ji}$ for all $1 \le i \le n$ and $1 \le j \le m$.

A square matrix $A \in \mathbb{R}^{n\times n}$ is *symmetric* if $A = A^T$.

A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is *positive semi-definite* if for all $d \in \mathbb{R}^n$, the product $d^T Ad \ge 0$. It is *positive definite* if $d^T Ad > 0$ holds for all $d \in \mathbb{R}^n$, $d \neq 0$. Equivalently, A is positive semi-definite if there exists $\alpha \ge 0$ such that for all $d \in \mathbb{R}^n$, $d^T Ad \ge \alpha \|d\|_2^2$. It is positive definitive if $\alpha > 0$.

There are corresponding definitions for *negative semi-definite* and *negative definite* matrices.

The following notation is typical to denote definiteness of a matrix. A is positive definite is denoted $A \succ 0$, A is positive semi-definite is denoted $A \succeq 0$, A is negative definite is denoted $A \prec 0$ and A is negative semi-definite is denoted $A \preceq 0$.

The set of symmetric positive semi-definite $n \times n$ matrices is a closed cone of $\mathbb{R}^{n\times n}$ denoted $\mathcal{S}_n^+$. The set of symmetric positive definite $n \times n$ matrix is an open cone of $\mathbb{R}^{n\times n}$ denoted $\mathcal{S}_n^{++}$.

Recall that a set K is a *cone* if for all $d \in K$ and all $\lambda \ge 0$, $\lambda d \in K$.

A matrix $A \in \mathbb{R}^{n\times m}$ is said to be *upper triangular* if $A_{ij} = 0$ whenever $i > j$. It is *lower triangular* if $A_{ij} = 0$ whenever $i < j$. It is *diagonal* if $A_{ij} = 0$ whenever $i \neq j$. Note that we also use the term *diagonal* for non-square matrices. Obviously, the "diagonal" of A is then A_{ii} for $i = 1, \dots, \min(n, m)$.

Given a vector $x \in \mathbb{R}^n$, we denote $\text{diag}(x)$ the $n \times n$ diagonal matrix A which has $A_{ii} = x_i$.

A matrix A is *orthogonal* if it is nonsingular and $A^{-1} = A^T$. An example of an orthogonal matrix is the matrix of a Givens rotation.

A.3.2 Subspaces

A subspace $\mathcal{V}$ of $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ that still satisfies the essential properties of $\mathbb{R}^n$ in relation to the addition of vectors and multiplication by a scalar:

(a) $0 \in \mathcal{V}$, and

(b) for all $x, y \in \mathcal{V}$ and all $\lambda \in \mathbb{R}$, $x + \lambda y \in \mathcal{V}$.

The so-called *trivial* subspaces of R^n are $\{0\}$ and $\mathbb{R}^n$ itself. The only subspaces of $\mathbb{R}$ are its trivial subspaces. The non-trivial subspaces of $\mathbb{R}^2$ are all straight lines passing through the origin. The non-trivial subspaces of $\mathbb{R}^3$ are all straight lines passing through the origin and all planes containing the origin.

A set $\{v_1, v_2, \dots, v_p\}$ of p vectors in $\mathbb{R}^n$ is said to be *linearly independent* if and only if the vector equation in the unknowns $\lambda_1, \lambda_2, \dots, \lambda_p \in \mathbb{R}$

$$\lambda_1 v_1 + \lambda_2 v_2 + \ldots + \lambda_p v_p = 0$$

has the unique solution $\lambda_1 = \lambda_2 = \ldots = \lambda_p = 0$. Otherwise it is said to be linearly *dependent.* If one of the vectors v_i is the vector zero, this set can never be linearly independent.

Given a set $\{v_1, v_2, \dots, v_p\}$ of p vectors in $\mathbb{R}^n$, a combination of the form $\lambda_1 v_1 + \lambda_2 v_2 + \ldots + \lambda_p v_p$ is called a *linear combination* of $v_1, v_2, \dots, v_p$. The set obtained by forming all the possible linear combinations of $v_1, v_2, \dots, v_p$ is a subspace of $\mathbb{R}^n$ denoted $\text{span}\{v_1, v_2, \dots, v_p\}$ and called the *subspace generated* by $v_1, v_2, \dots, v_p$. In this case, $\{v_1, v_2, \dots, v_p\}$ is a *generating set* of this subspace. If one of the v_i is a linear combination of the other vectors v_j ($j \neq i$), then

$$\text{span}\{v_1, v_2, \ldots, v_{i-1}, v_{i+1}, \ldots, v_p\} = \text{span}\{v_1, v_2, \ldots, v_p\}.$$

By recursively ridding the set $\{v_1, v_2, \ldots, v_p\}$ of all vectors v_i that are a linear combination of the other remaining vectors, we obtain a minimal set of vectors $\{v_{i_1}, \ldots, v_{i_q}\} \subseteq \{v_1, v_2, \ldots, v_p\}$ such that

$$\text{span}\{v_{i_1}, \ldots, v_{i_q}\} = \text{span}\{v_1, v_2, \ldots, v_p\}.$$

This last set of vectors is not only a generating set of this subspace but is also linearly independent. A set of vectors possessing these two properties is called a *basis*.

The trivial subspace $\{0\}$ possesses no basis since there it contains no linearly independent vector.

If $\mathcal{V}$ is a subspace of $\mathbb{R}^n$ and if $\{v_1, v_2, \ldots, v_p\}$ is a basis of $\mathcal{V}$, every $x \in \mathcal{V}$ is a *unique* linear combination of $v_1, v_2, \ldots, v_p$. If we gather the vectors $v_1, v_2, \ldots, v_p$ in the $n \times p$ matrix V where the i-th column of V is v_i, the elements of $\mathcal{V}$ can always be written in the form Vy for some $y \in \mathbb{R}^p$. Indeed, Vy is the linear combination $y_1 v_1 + \ldots + y_p v_p$ where $y_1, \ldots, y_p$ are the components of y.

The *dimension* of a subspace is the number of elements in any of its bases. By convention, the dimension of $\{0\}$ is 0. The dimension of $\mathbb{R}$ is 1, that of $\mathbb{R}^2$ is 2, that of $\mathbb{R}^3$ is 3, and so on. For any n, the dimension of $\mathbb{R}^n$ is n.

In $\mathbb{R}^n$, subspaces of dimension $n-1$ are usually called *hyperplanes*. In $\mathbb{R}$, the only hyperplane is $\{0\}$. In $\mathbb{R}^2$ the hyperplanes are the straight lines. In $\mathbb{R}^3$, the hyperplanes are the planes, which is the origin of the name.

A.3.3 Orthogonal complements

If $\mathcal{V}$ is a subspace of $\mathbb{R}^n$, its *orthogonal complement* is the set $\mathcal{V}^\perp$ defined by

$$\mathcal{V}^\perp = \{x \in \mathbb{R}^n \mid x^T y = 0 \ \forall y \in \mathcal{V}\}.$$

It is easy to see that $\mathcal{V}^\perp$ is also a subspace of $\mathbb{R}^n$. It is called the *orthogonal complement* of $\mathcal{V}$. In Euclidian spaces like $\mathbb{R}^n$ or $\mathbb{C}^n$, we always have the property $\mathcal{V}^{\perp\perp} \equiv (\mathcal{V}^\perp)^\perp = \mathcal{V}$.

The orthogonal complement of $\{0\}$ is $\mathbb{R}^n$ and the orthogonal complement of $\mathbb{R}^n$ is $\{0\}$. The orthogonal complement of a straight line in $\mathbb{R}^2$ is the straight line that passes through zero and is perpendicular to the first. The orthogonal complement of a straight line in $\mathbb{R}^3$ is the plane containing zero and that is perpendicular to the straight line.

The notion of orthogonality is more general than the intuitive notion of perpendicularity however, since it depends on the inner product. Different inner products give rise to different orthogonal complements.

Orthogonal complements have the property that any $x \in \mathbb{R}^n$ can be written $x = x_1 + x_2$ where $x_1 \in \mathcal{V}$ and $x_2 \in \mathcal{V}^\perp$. Such a decomposition is called an *orthogonal decomposition*. This decomposition is *unique*. To signify this, the notation $\mathbb{R}^n = \mathcal{V} \oplus \mathcal{V}^\perp$ is sometimes used.

A.3.4 Nullspace and image

Given an $m \times n$ real matrix A, the *nullspace* of A denoted Null(A) is the set $\{x \in \mathbb{R}^n \mid Ax = 0\} \subseteq \mathbb{R}^n$. It is easy to see that Null(A) is always a subspace of $\mathbb{R}^n$. Possibly, Null(A) $= \{0\}$. Sometimes, the nullspace of A is called the *kernel* of A. In French, the notation Ker(A) is typically used instead of Null(A).

The *range space* of A, is the set denoted Range(A) defined by $\{y \in \mathbb{R}^m \mid \exists x \in \mathbb{R}^n \; Ax = y\}$ which is equivalently written $\{Ax \mid x \in \mathbb{R}^n\}$. Again, the range space is always a subspace of $\mathbb{R}^m$ which is possibly equal to $\mathbb{R}^m$. It is sometimes called the *image* of A. In French, the notation Im(A) is typically used instead of Range(A).

We can now consider the $n \times m$ matrix A^T. Its nullspace is a subspace of $\mathbb{R}^m$ and its range space is a subspace of $\mathbb{R}^n$.

The so-called *fundamental theorem of linear algebra* states that

$$\text{Null}(A) = \text{Range}(A^T)^\perp,$$

and this has the consequence that $\mathbb{R}^n = \text{Null}(A) \oplus \text{Range}(A^T)$.

A.3.5 Eigenvalues and singular values

A vector $v \in \mathbb{R}^n$, $v \neq 0$ is an *eigenvector* associated to the *eigenvalue* $\lambda \in \mathbb{C}$ of the matrix $A \in \mathbb{R}^{n\times n}$ if $Av = \lambda v$. The eigenvalues of A are the roots of the degree n polynomial $p(\lambda) = \det(A - \lambda I)$ called the characteristic polynomial. The set of all eigenvectors associated to a given eigenvalue λ of A is a suspace of $\mathbb{R}^n$ called the *eigenspace* associated to λ. A is *diagonalizable* if there exists a basis made of eigenvectors. In this case there is a nonsingular matrix P such thati

$$P^{-1}AP = D = \text{diag}(\lambda_1, \ldots, \lambda_n)$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A. This decomposition is called the eigenvector decomposition of A. This occurs when, for all eigenvalue λ of A, λ is real and its multiplicity as a root of $p(\lambda)$ equals the dimension of its associated eigenspace.

When A is not diagonalizable, it is still possible to reduce it to a simpler forms, known as the Jordan form. We do not detail this here.

If A is symmetric its eigenvalues are always real and it is always diagonalizable. Moreover it is always possible to choose P orthogonal so that $P^TAP = \text{diag}(\lambda_1, \ldots, \lambda_n)$.

Using eigenvalues we have the following results. A symmetric matrix A is positive definite if and only if all its eigenvalues are positive. It is positive semi-definite if and only if all its eigenvalues are nonnegative. It is negative definite if and only if all its eigenvalues are negative. It is negative semi-definite if and only if all its eigenvalues are nonpositive.

More generally, any matrix $A \in R^{n\times m}$ may be factored into a product of three matrices

$$A = USV^T = \sum_{i=1}^{\min(n,m)} \sigma_i u_i v_i^T, \tag{I.2}$$

where $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$ are orthogonal and $S \in \mathbb{R}^{n\times m}$ is diagonal. The elements S_{ii}, denoted σ_i are called the *singular values* of A and are the square roots of the eigenvalues of A^TA, which is positive semi-definite. Thus, these singular values are always nonnegative and they can be ordered so that

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\min(n,m)} \geq 0.$$

The factorization (I.2) is called the *singular value decomposition*, or SVD for short. It is usually an expensive decomposition to obtain computationally but a most useful theoretical tool. Singular values have the following property. For all $d \in \mathbb{R}^n$, $\sigma_{\min(n,m)}\|d\|^2 \leq d^TAd \leq \sigma_1\|d\|^2$.

If A is diagonalizable, its singular value decomposition coincides with its eigenvector decomposition. In this case, the matrices A and $\mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ are *similar* and in particular, the determinant of A is given by

$$\det(A) = \Pi_{i=1}^{n} \lambda_i,$$

the trace of A is given by

$$\mathrm{tr}(A) \equiv \sum_{i=1}^{n} A_{ii} = \sum_{i=1}^{n} \lambda_i$$

and the rank of A is the number of nonzero eigenvalues. The determinant, trace and rank and invariant under a similarity transformation. Therefore, A is singular if and only if it has at least one zero eigenvalue.

If P is an orthogonal matrix, it always satisfies $\|Px\| = \|x\|$ in ℓ_2-norm, the eigenvalues of P are either -1 or 1 and the singular values of P are all 1.

A.3.6 Matrix norms

The vector space $\mathbb{R}^{n\times m}$ becomes a normed space if we endow it with a matrix norm. A special class of matrix norms is derived from vector norms using

$$\|A\| = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}.$$

Such matrix norms are said to be *induced* by the vector norm. For the most common vector norms we obtain

$$\begin{array}{rcl} \|A\|_1 & = & \max_{1\leq j\leq m} \sum_{i=1}^{n} |A_{ij}|, \\ \|A\|_\infty & = & \max_{1\leq i\leq n} \sum_{j=1}^{m} |A_{ij}|, \\ \|A\|_2 & = & \sqrt{\sigma_1(A)}. \end{array}$$

The ℓ_1-norm of a matrix is the largest ℓ_1-norm of its columns, its ℓ_∞-norm is the largest ℓ_1-norm of its rows and its ℓ_2-norm is the square root of its largest singular value.

Bibliography

[CGT00] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. SIAM, Philadelphia, USA, 2000.

[CPL98] CPLEX 6.0. *High-performance linear, integer and quadratic programming software*. ILOG SA, Gentilly, France, 1998. `www.cplex.com`.

[Dan63] G. B. Dantzig. *Linear Programming and Extensions*. Princeton Landmarks in Mathematics. Princeton University Press, Princeton, NJ, USA, 1963.

[DS96] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and non-linear equations*. Classics in Applied Mathematics. SIAM, Philadelphia PA, USA, 1996.

[FGK02] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press / Brooks/Cole Publishing Company, second edition, 2002.

[FGW03] A. Forsgren, P. E. Gill, and M. H. Wright. Interior methods for nonlinear optimization. *SIAM Review*, 44(4):525–597, 2003.

[FM68] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. J. Wiley and Sons, Chichester, England, 1968. Reprinted as *Classics in Applied Mathematics*, SIAM, Philadelphia, USA, 1990.

[FMN$^+$04] R. Fourer, C. Maheshwari, A. Neumaier, D. Orban, and H. Schichl. Convexity and concavity detection. Preprint, University of Vienna, Austria, 2004.

[FO07] R. Fourer and D. Orban. The drampl meta solver for optimization. Technical Report G-2007-10, GERAD, Montreal, Canada, 2007.

[GQT66] S. M. Goldfeldt, R. E. Quandt, and H. F. Trotter. Maximization by quadratic hill-climbing. *Econometrica*, 34:541–551, 1966.

[Hei00] L. Hei. A self-adaptive trust region algorithm. Technical Report, ECE Department, Northwestern University, Evanston IL, USA, 2000.

[HJ85] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorics*, 4:373–395, 1984.

[KM72] V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities*, pages 159–175. Academic Press, New-York, 1972.

[KMY89] M. Kojima, S. Mizuno, and A. Yoshise. A primal-dual interior point algorithm for linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point and Related Methods*, pages 29–48. Springer-Verlag, New-York, NY, USA, 1989.

[Lev44] K. Levenberg. A method for the solution of certain problems in least squares. *Quarterly Journal on Applied Mathematics*, 2:164–168, 1944.

[Mar63] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.

[Meg89] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point and Related Methods*, pages 131–158. Springer-Verlag, New-York, NY, USA, 1989.

[Mor60] D. D. Morrison. Methods for nonlinear least squares problems and convergence proofs. In J. Lorell and F. Yagi, editors, *Proceedings of the Seminar on Tracking Programs and Orbit Determination*, pages 1–9, Pasadena, USA, 1960. Jet Propulsion Laboratory.

[Mor83] J. J. Moré. Recent developments in algorithms and software for trust region methods. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming: The State of the Art*, pages 258–287, Heidelberg, Berlin, New York, 1983. Springer Verlag.

[MW93] J. J. Moré and S. J. Wright. *Optimization Software Guide*. SIAM, Philadelphia, 1993.

[Pow70] M. J. D. Powell. A new algorithm for unconstrained optimization. In J. B. Rosen, O. L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65, London, 1970. Academic Press.

[WD04] J. M. B. Walmag and E. J. M. Delhez. A note on trust-region radius update. Technical Report, Modélisation et Méthodes Mathématiques, Université de Liège, Belgium, 2004.

[Wri92] M. H. Wright. Interior methods for constrained optimization. *Acta Numerica*, 1:341–407, 1992.

[Wri98a] M. H. Wright. Ill-conditioning and computational error in interior methods for nonlinear programming. *SIAM Journal on Optimization*, 9:84–111, 1998.

[Wri98b] M. H. Wright. The Interior-Point Revolution in Constrained Optimization. In R. DeLeone, A. Murli, P.M. Pardalos, and G. Toraldo, editors, *High-Performance Algorithms and Software in Nonlinear Optimization*, pages 359–381, Dordrecht, 1998. Kluwer Academic Publishers.

[Wri01] S. J. Wright. Effects of finite-precision arithmetic on interior-point methods for nonlinear programming. *SIAM Journal on Optimization*, 12:36–78, 2001.

[Yua98] Y. Yuan. An example of non-convergence of trust region algorithms. In Y. Yuan, editor, *Advances in Nonlinear Programming*, pages 205–218, Dordrecht, The Netherlands, 1998. Kluwer Academic Publishers.