Manuscript for the course "Numerical Optimization"

Moritz Diehl
Department of Microsystems Engineering and Department of Mathematics,
University of Freiburg, Germany
moritz.diehl@imtek.uni-freiburg.de

May 18, 2026

Preface

This course's aim is to give an introduction into numerical methods for the solution of optimization problems in science and engineering. It is intended for students from two faculties, mathematics and physics on the one hand, and engineering and computer science on the other hand. The course's focus is on *continuous optimization* (rather than discrete optimization) with special emphasis on **nonlinear programming**. For this reason, the course is in large parts based on the excellent text book "Numerical Optimization" by Jorge Nocedal and Steve Wright [4]. This book appeared in Springer Verlag and recommended to the students. Besides nonlinear programming, we discuss important and beautiful concepts from the field of **convex optimization** that we believe to be important to all users and developers of optimization methods. These contents and much more are covered by the equally excellent text book "Convex Optimization" by Stephen Boyd and Lieven Vandenberghe [2], that was published by Cambridge University Press (CUP). Fortunately, this book is also freely available and can be downloaded from the home page of Stephen Boyd in form of a completely legal PDF copy of the CUP book. An excellent textbook on nonlinear optimization that contains also many MATLAB exercises was recently written by Amir Beck [1].
The course is divided into three major parts:

- Fundamental Concepts of Optimization
- Unconstrained Optimization and Newton Type Algorithms
- Equality and Inequality Constrained Optimization

followed by two appendices, the first containing the description of one student project done during the course exercises, and some remarks intended to help with exam preparation (including a list of questions and answers). The writing of this lecture manuscript started at the Optimization in Engineering Center OPTEC of KU Leuven in 2007, with major help of Jan Bouckaert (who did, among other, most of the figures), David Ariens, and Laurent Sorber. In the years from 2007 until today, many students and researchers helped with feedback and with spotting errors, with special thanks to Dr. Carlo Savorgnan, Dr. Dimitris Kouzoupis, Jonathan Frey and Florian Messerer. Particular thanks go to to Léo Simpson for carefully reading and helping to re-edit the manuscript in 2025 and 2026.

Moritz Diehl,
Freiburg,
April 2026

moritz.diehl@imtek.uni-freiburg.de

Contents

Part I

Fundamental Concepts of Optimization

Chapter 1

Introduction

1.1 Optimization in the Real World

Optimization problems are a modeling choice for many real-world problems or questions and can arise in many real-world domains. They are used to model questions such as "What is the resource allocation that induces the best average satisfaction while satisfying some requirements?" or "What are the most likely physical parameters based on these measurements?" We can classify the applications of optimization into two major categories as follows.

a. In **decision making**, we are faced with a set of possible decisions, and we want to find the decision that minimizes some cost. This decision can be made based on the solution to some optimization problem. A couple of examples of applications are:

   - optimal allocation of resources in logistics;
   - optimal scheduling;
   - optimal investments in portfolio management;
   - optimal design of systems in engineering (e.g., bridges, cars, aircraft, digital devices, etc.);
   - optimal control of a system (e.g., temperature control units, self-driving cars, walking robots, etc.).

b. In **model learning**, a set of data is available, and we want to find a model that fits the data and possibly meets some additional requirements. This can be modeled as an optimization problem. Particularly relevant today is the task of *supervised machine learning*, a function that is approximated based on a large amount of input-output samples. There, the function approximation is done by choosing a class of functions that depend only on a finite number of parameters. These parameters are modeled as *optimization variables* that are chosen to *minimize* some metric of how much the function fits to the input-output samples.

These important applications explains why studying optimization problems is such an important topic in applied mathematics. The main question we are interested in is how to characterize the solutions to such problems and how to find these solutions. In this course, we will present the important theoretical results regarding optimization problems and introduce the most important *optimization algorithms* to solve these problems numerically.

Example 1.1: Shipment cost minimization
We regard a typical logistics problem that a production/distribution company might encounter. We want to minimize the transportation costs from the producers to consumers, as visualized in Figure 1.1. More specifically, we model the problem as follows:

- there are n production facilities with production capacities p_i for $i = 1, \dots, n$;
- there are m consumer locations with demands d_j for $j = 1, \dots, m$;
- transporting one unit of production from producer i to consumer j costs c_{ij};
- the quantity transported from producer i to consumer j is the decision variable x_{ij}.

The goal is to *minimize* the total cost $\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$ under the *constraints* that:

i. every transported quantity x_{ij} can not be negative: $x_{ij} \geq 0$;

ii. a producer can not export more than what it produces: $\sum_{j=1}^{m} x_{ij} \leq p_i$;

iii. a consumer must receive at least what it demands: $\sum_{i=1}^{n} x_{ij} \geq d_j$.

Mathematically, this optimization problem is formulated as follows:

$$\underset{x \in \mathbb{R}^{n \times m}}{\text{minimize}} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \tag{1.1a}$$

$$\text{subject to} \quad x_{ij} \geq 0, \quad i = 1, \dots, n, \quad j = 1, \dots, m, \tag{1.1b}$$

$$\sum_{j=1}^{m} x_{ij} \leq p_i, \quad i = 1, \dots, n, \tag{1.1c}$$

$$\sum_{i=1}^{n} x_{ij} \geq d_j, \quad j = 1, \dots, m. \tag{1.1d}$$

There exists some algorithms that can solve this kind problem efficiently as we will see in this course.

1.2 Optimization Problems (qualitative description)

An optimization problem consists of the following three ingredients.

i. The *decision variables* or *optimization variables*, usually stacked into a vector $x \in \mathbb{R}^n$, that can be chosen;

ii. An objective function $f : \mathbb{R}^n \to \mathbb{R}$ that guides the choice of the decision variables. This function shall be minimized or maximized;

iii. Some *equality constraints* in the form $g(x) = 0$ and *inequality constraints* in the form $h(x) \geq 0$ that restrict the choice of the decision variables. We say that a solution candidate is *feasible* if it satisfies these constraints.

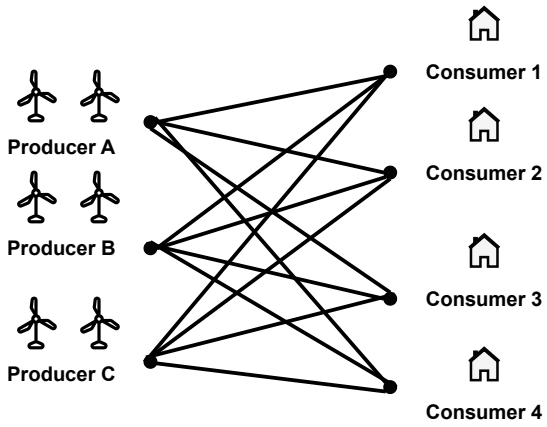


Figure 1.1: A classical optimization problem: minimize the shipment costs while satisfying the demands (on the right) and not exceeding the production capabilities (on the left).

1.3 Optimization Problems (mathematical description)

Definition 1.1: Optimization problems
An optimization problem is the following minimization problem:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(1.2a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(1.2b)} \\ & h(x) \geq 0, && \text{(1.2c)} \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $g : \mathbb{R}^n \to \mathbb{R}^p$ defines the equality constraints, and $h : \mathbb{R}^n \to \mathbb{R}^q$ defines the inequality constraints.

Remark
Note that we write "$h(x) \geq 0$" for the component-wise inequality, i.e., we require $h_j(x) \geq 0$ for each $j = 1, \ldots, q$.

Remark
A maximization problem can also be formulated, with a *utility function* $u(\cdot)$:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{maximize}} \quad & u(x) && \text{(1.3a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(1.3b)} \\ & h(x) \geq 0, && \text{(1.3c)} \end{aligned}$$

In this course, we will prefer reformulating them as a minimization problem: instead of maximizing utility, we minimize the negative utility $f(x) \coloneqq -u(x)$.

Example 1.2: A two-dimensional example
The following tutorial example of an optimization problem is illustrated in Figure 1.2.

$$\begin{aligned} \underset{x \in \mathbb{R}^2}{\text{minimize}} \quad & x_1^2 + x_2^2 && \text{(1.4a)} \\ \text{subject to} \quad & x_2 \geq 1 + x_1^2, && \text{(1.4b)} \\ & x_1 \geq 1. && \text{(1.4c)} \end{aligned}$$

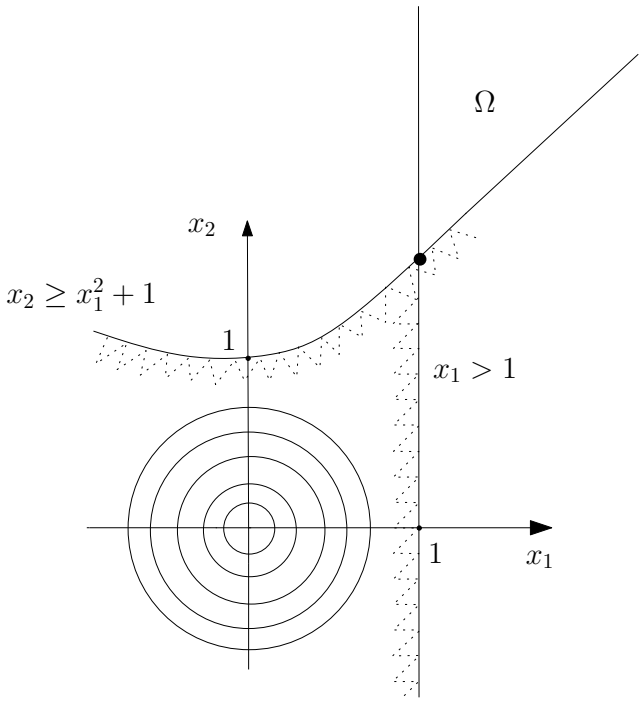


Figure 1.2: Visualization of Example 1.2, Ω is the feasible set.

Note that the inequality constraints of (1.4) can be rearranged into the standard form:

$$\underset{x \in \mathbb{R}^2}{\text{minimize}} \quad \overbrace{x_1^2 + x_2^2}^{f(x)} \tag{1.5a}$$

$$\text{subject to} \quad \underbrace{\begin{bmatrix} x_2 - 1 - x_1^2 \\ x_1 - 1 \end{bmatrix}}_{h(x)} \geq 0. \tag{1.5b}$$

1.4 Definitions of Basic Concepts

Definition 1.2: Feasible sets
The "feasible set" $\Omega \subset \mathbb{R}^n$ is the set of variables that satisfies the constraints, as defined below.

$$\Omega = \left\{ x \in \mathbb{R}^n \;\middle|\; g(x) = 0 \text{ and } h(x) \geq 0 \right\} \tag{1.6}$$

Definition 1.3: Global minimizers
We say that $x^\star$ is a *global minimizer* when it is feasible, i.e. $x^\star \in \Omega$ and optimal, i.e. $\forall x \in \Omega : f(x) \geq f(x^\star)$.

Definition 1.4: Strict global minimizers
We say that $x^\star$ is the *strict global minimizer* when $x^\star \in \Omega$ and it is strictly optimal, i.e. $\forall x \in \Omega \setminus \{x^\star\} : f(x) > f(x^\star)$.

Definition 1.5: Local minimizers
We say that $x^\star$ is a *local minimizer* when there exists a neighborhood $\mathcal{N}$ of $x^\star$ (e.g. an open ball around $x^\star$) such that it is optimal within that neighborhood, i.e. $\forall x \in \Omega \cap \mathcal{N} : f(x) \geq f(x^\star)$.

Definition 1.6: Strict local minimizers
We say that $x^\star$ is a *strict local minimizer* when there exists a neighborhood $\mathcal{N}$ of $x^\star$ such that it is strictly optimal within that neighborhood, i.e., $\forall x \in (\Omega \cap \mathcal{N}) \setminus \{x^\star\} : f(x) > f(x^\star)$.

Example 1.3: A one dimensional example
Consider the following example, illustrated in Figure 1.3.

$$\begin{aligned} \underset{x \in \mathbb{R}}{\text{minimize}} \quad & \sin(x)\exp(x) && \text{(1.7a)} \\ \text{subject to} \quad & x \geq 0, && \text{(1.7b)} \\ & x \leq 4\pi && \text{(1.7c)} \end{aligned}$$

i. $\Omega = \{x \in \mathbb{R} \,|\, \|x\| \geq 0,\ x \leq 4\pi\} = [0, 4\pi]$

ii. Three local minimizers (which?)

iii. One global minimizer (which?)

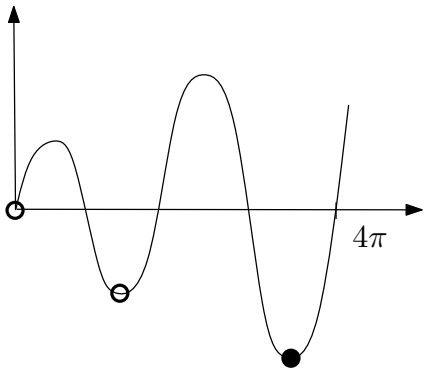


Figure 1.3: Visualization of Example 1.3

1.5 Existence of Solutions

For a given optimization problem, there is unfortunately not always a solution. This is illustrated in the following example.

Example 1.4: Optimization problems with no solution
The following optimization problem has no solution because the value is always positive, but can be arbitrarily small.

$$\underset{x \in \mathbb{R}}{\text{minimize}} \quad \frac{1}{1+x^2} \tag{1.8}$$

The following optimization problem also has no solution, even though the feasible set is bounded because the value can be arbitrarily negative.

$$\underset{x \in (0,1)}{\text{minimize}} \quad -\frac{1}{x} \tag{1.9}$$

That being said, minimizers *usually* exist, as suggested by the two following theorems.

Theorem 1.1: Weierstrass
If the feasible set $\Omega \subset \mathbb{R}^n$ is non-empty and compact (i.e. bounded and closed) and $f : \Omega \to \mathbb{R}$ is continuous, then there exists at least one global minimizer.

Proof. Let us define $f^\star \coloneqq \inf_{x\in\Omega} f(x)$ (a priori, $f^\star \in \mathbb{R} \cup \{-\infty\}$). By definition of the infimum, there exists a sequence $(x_k)_{k\in\mathbb{N}}$ in Ω such that $f(x_k) \underset{k\to+\infty}{\longrightarrow} f^\star$. Since Ω is compact, the sequence x_k has at least one accumulation point $\bar{x} \in \Omega$, i.e. for some increasing sequence of integers $(k_j)_{j\in\mathbb{N}}$, we have $x_{k_j} \underset{j\to+\infty}{\longrightarrow} \bar{x}$ and hence $f(x_{k_j}) \underset{k_j\to+\infty}{\longrightarrow} f(\bar{x})$ (by continuity of $f(\cdot)$). By combining these two limits, we obtain $f(\bar{x}) = f^\star = \inf_{x\in\Omega} f(x)$. This proves that $\bar{x}$ is a minimizer of f over Ω. □

Theorem 1.2: Existence of minimizer for coercive functions
Assume the three following things:

i. the functions f, g and h are continuous,

ii. the function $f(\cdot)$ is *coercive*, i.e. $f(x) \underset{\|x\|\to+\infty}{\longrightarrow} +\infty$,

iii. there is at least one feasible point

Then, the optimization problem (1.2) has at least one global minimizer.

Proof. Let x_0 be a feasible point. The following set is non-empty, closed and bounded:

$$\tilde{\Omega} = \left\{ x \in \mathbb{R}^n \;\middle|\; g(x) = 0,\ h(x) \geq 0,\ \text{and } f(x) \leq f(x_0) \right\}. \tag{1.10}$$

Hence, by applying Theorem 1.1, f has a global minimizer over the set $\tilde{\Omega}$. It follows that this minimizer is also a minimizer over the whole feasible set Ω, which concludes the proof. □

1.6 Mathematical Notation

In this lecture, we use quite standard mathematical notation, but we define them rigorously here.

- The sets $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ are, as usually, the set of natural numbers (excluding 0), integers and real numbers, respectively.
- The space $\mathbb{R}^n$ is the set of vectors with n components, and $\mathbb{R}^{n\times m}$ denotes the set of matrices with n rows and m columns. All vectors are column vectors, i.e. we identify $\mathbb{R}^n = \mathbb{R}^{n\times 1}$.
- For two real numbers $a < b$, we denote by $[a, b]$ the closed interval (that contains a and b), $[a, b)$ is a semi-open interval (that contains a but not b), and so on.
- For $x \in \mathbb{R}^n$, $\|x\|$ is an arbitrary norm. Most of the time, we use the L_1, L_2 or L_∞ norms:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|, \qquad \|x\|_2 \coloneqq \sqrt{\sum_{i=1}^{n} x_i^2}, \qquad \|x\|_\infty = \max_{i=1,\dots,n} |x_i|. \tag{1.11}$$

- We use *Landau notations* "$f(x) = o_{x\to\bar{x}}(g(x))$" and "$f(x) = \mathcal{O}_{x\to\bar{x}}(g(x))$" to compare the assymptotic behavior of the functions f and g when $t \to \bar{x}$ (this is also defined for $\bar{x} = +\infty$):

$$f(x) = o_{x\to\bar{x}}(g(x)) \quad \text{if and only if} \quad \frac{f(x)}{g(x)} \underset{x\to\bar{x}}{\longrightarrow} 0, \tag{1.12a}$$

$$f(x) = \mathcal{O}_{x\to\bar{x}}(g(x)) \quad \text{if and only if} \quad \forall x \in \mathcal{N}_{\bar{x}} : \frac{f(x)}{g(x)} \leq C \text{ for some constant } C\ , \tag{1.12b}$$

  where $\mathcal{N}_{\bar{x}}$ is a neighborhood of $\bar{x}$.

  Usually, $\bar{x}$ is clear from the context, so we will usually omit the index, i.e., we will just write "$f(x) = o(g(x))$" and "$f(x) = \mathcal{O}(g(x))$"

- For a *symmetric* matrix $P \in \mathbb{R}^{n\times n}$ (i.e. $P^\top = P$), we say that P is *positive semi-definite* and write $P \succcurlyeq 0$ when one of the following statements is valid (they are equivalent):

  i. all of the eigenvalues of P are larger or equal to zero
  ii. for any $x \in \mathbb{R}^n$, we have $x^\top P x \geq 0$.

  We will also write $P \succcurlyeq R$ (for another symmetric matrix R) when $P - R \succcurlyeq 0$ and $P \preccurlyeq R$ when $R - P \succcurlyeq 0$.

  It is important to note that *there is no connection* between $P \succcurlyeq 0$ and the positiveness of the components of P.

  It is also important to note that we do not always have either $P \succcurlyeq 0$ or $P \preccurlyeq 0$ (e.g. if $P = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, then neither $P \succcurlyeq 0$ nor $P \prec 0$ is true).

- We say that a symmetric matrix P is positive definite or write $P \succ 0$ when one of the following statements is valid (they are equivalent):

  i. all of the eigenvalues of P are strictly larger than zero
  ii. for any $x \in \mathbb{R}^n$ such that $x \neq 0$, we have $x^\top P x > 0$.

- For a scalar function $f : \mathbb{R}^n \to \mathbb{R}$, we denote *the gradient of* f as follows:

$$\nabla f(x) \coloneqq \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix} \in \mathbb{R}^n, \tag{1.13}$$

  where $\frac{\partial f}{\partial x_i}(x)$ denotes the partial derivative of f with respect to the variable x_i.

- Slightly less standard, when we deal with derivatives of functions F with several real inputs and several real outputs (i.e. $x \in \mathbb{R}^n$ and $F(x) \in \mathbb{R}^m$), we generalize the definition above as follows:

$$\nabla F(x) = \begin{bmatrix} \nabla F_1(x) & \dots & \nabla F_m(x) \end{bmatrix} \in \mathbb{R}^{n\times m}, \tag{1.14}$$

  This is the transpose of the Jacobian: $\frac{\partial F}{\partial x}(x) = \nabla F(x)^\top \in \mathbb{R}^{m\times n}$: Using this notation, the first-order Taylor series is written as:

$$F(x) = F(\bar{x}) + \nabla F(\bar{x})^\top (x - \bar{x}) + \mathcal{O}(\|x - \bar{x}\|^2) \tag{1.15}$$

- If a scalar function $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable (which is written $f \in C^2$), the square matrix containing all second derivatives $\frac{\partial^2 f}{\partial x_i \partial x_j}$ is called the *Hessian matrix* and is denoted by $\nabla^2 f(x) \in \mathbb{R}^{n \times n}$. This matrix is symmetrix (cf. Schwarz's theorem).
- For logical implications and equivalences, we use the symbol $A \implies B$ and $A \iff B$ respectively. In words, we will write "If A then B" and "A if and only if B" respectively.

Chapter 2

Types of Optimization Problems

In this chapter, we present the important classes of optimization problems. In order to choose the right algorithm for a practical problem, it is important to know which kind of problem we are facing. Replacing an inadequate algorithm with a suitable one can make solution times many orders of magnitude shorter.

2.1 Nonlinear Programming (NLP)

In this lecture, we mainly treat algorithms for general NonLinear Programming problems (NLP), which are given in the general form, as before:

Definition 2.1: NonLinear Programming (NLP)
A NonLinear Programming problem (NLP) refers to the general class of optimization problems where the functions involved are smooth, and where the decision variables are vectors of $\mathbb{R}^n$. As before, we write them as follows:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(2.1a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(2.1b)} \\ & h(x) \geq 0, && \text{(2.1c)} \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^p$, $h : \mathbb{R}^n \to \mathbb{R}^q$, are assumed to be continuously differentiable at least once. Sometimes we will need them to be twice differentiable.

The differentiability of the functions allows us to use algorithms that are based on first and second derivatives, in particular the so-called "Newton type optimization methods," which are the main topic of this course. On the other hand, many problems have more specific structures, which one can exploit to solve them more efficiently.

2.2 Linear Programming (LP)

A much more restrictive class of problems is the class of Linear Programs (LP), defined as follows.

Definition 2.2: Linear Programming (LP)
A Linear Program (LP) is a special case of an NLP, where the functions f, g, and h are affine. Explicitly, we can write it as follows:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c_0 + c^\top x && \text{(2.2a)} \\ \text{subject to} \quad & Ax = b, && \text{(2.2b)} \\ & Cx \geq d, && \text{(2.2c)} \end{aligned}$$

where A, b, C, and d are matrices and vectors of appropriate dimensions. Note that the constant part of the objective "c_0+" is not important, since it does not affect the solution.

For solving LPs, the *only* difficulty comes from the inequality constraints, which cannot be solved using basic linear algebra only (as for the equality constraints).
The first efficient algorithm dates back to the 1940s, when the American mathematician George Dantzig invented the famous "simplex method," which is still widely used today. This method is based on a well-known property about LPs: an optimal solution can always be found at a vertex of the feasible set. Using this fact, the simplex method is an efficient way to traverse the vertices of the feasible set until an optimal one is found. In this lecture, we will not treat the simplex method in detail, but instead, we will discuss *active set methods* in Section 14.1, a more general class of algorithms that includes the simplex method. Later, in the 1980s, the Indian mathematician Narendra Karmarkar invented an algorithm for solving LPs: the "interior point method". This was the first algorithm that could solve LPs in polynomial time. The idea is to iterate within the interior of the feasible set (even though for LPs, the solution is at the boundary). This is done by adding a *barrier term* to the objective function, which pushes the iterates away from the constraints. Interestingly, this method is not specific to LPs; it can actually be used in a similar way for general NLPs, as we will see in Section 14.4.
Nowadays, LPs with *millions* of variables and constraints can be solved quickly. Most business students know how to use them, and they arise in myriads of applications. For example, note that the optimization problem (1.1) from Example 1.1 is an LP.

Software for solving LPs: When you encounter an LP, you should recognize it and use the right software. There are many solvers for LPs, both commercial and free: CPLEX, SOPLEX, lp_solve, lingo, linprog (for MATLAB), etc.

2.3 Quadratic Programming (QP)

A slightly more general class of problems are the Quadratic Programs (QP). These are similar, but some *curvature* (i.e., second derivatives) is allowed in the objective function.

Definition 2.3: Quadratic Programming (QP)
A Quadratic Program (QP) is a special case of an NLP, where the functions g and h are affine and the function f is quadratic. Explicitly, we can write it as follows:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^\top x + \frac{1}{2} x^\top B x && \text{(2.3a)} \\ \text{subject to} \quad & Ax = b, && \text{(2.3b)} \\ & Cx \geq d, && \text{(2.3c)} \end{aligned}$$

where A, B, b, C, and d are matrices and vectors of appropriate dimensions. We will also impose B to be symmetric, such that $\nabla^2 f(x) = B$.

Definition 2.4: Convex QP
If the Hessian matrix B is positive semi-definite, i.e., $B \succcurlyeq 0$, we call the QP (2.3) a "convex QP". If the Hessian is positive definite, i.e., $B \succ 0$, we call it a "strictly convex QP".

Convex QPs are tremendously easier to solve than general QPs, mainly because every local minimizer is also a global minimizer. In the next chapter, we will see a more general definition of convexity.

Example 2.1: A non-convex QP

$$\begin{aligned} \underset{x \in \mathbb{R}^2}{\text{minimize}} \quad & 2x_2 + \frac{5}{2} x_1^2 - \frac{1}{2} x_2^2 && \text{(2.4a)} \\ \text{subject to} \quad & -1 \leq x_1 \leq 1, && \text{(2.4b)} \\ & -1 \leq x_2 \leq 10. && \text{(2.4c)} \end{aligned}$$

This QP is not convex (which makes it difficult to solve). Here, the problem has two local minimizers: $x_a^\star = (0, -1)^\top$ and $x_b^\star = (0, 10)^\top$, but only $x_b^\star$ is a global minimizer.

Example 2.2: A strictly convex QP

$$\begin{aligned} \underset{x \in \mathbb{R}^2}{\text{minimize}} \quad & 2x_2 + \frac{5}{2} x_1^2 + \frac{1}{2} x_2^2 && \text{(2.5a)} \\ \text{subject to} \quad & -1 \leq x_1 \leq 1, && \text{(2.5b)} \\ & -1 \leq x_2 \leq 10. && \text{(2.5c)} \end{aligned}$$

The following QP is strictly convex (which makes it easy to solve). This problem has only one (strict) local minimizer at $x^\star = (0, -1)^\top$. That is also the global minimizer.

Software for solving quadratic programs: Like for LPs, there are many solvers for QPs, both commercial and free: CPLEX, MOSEK, qpOASES (open), OOQP (open), quadprog (for

MATLAB).

2.4 General Convex Optimization Problems

Both, LPs and convex QPs, are part of an important class of optimization problems, namely the "convex optimization problems". These are problems where the objective function is convex and the feasible set is a convex set. The definition of these concepts, and the properties of convex optimization problems, will be treated in Chapter 3. In order to define them and understand why they are so important, we first recall what is a convex set and a convex function.

2.5 Unconstrained Optimization Problems

Any NLP without constraints is called an "unconstrained optimization problem".

Definition 2.5: Unconstrained Optimization Problem
An Unconstrained Optimization Problem (UOP) is a special case of an NLP where there are no constraints, i.e., the feasible set is $\Omega = \mathbb{R}^n$. Explicitly, we can write it as follows:

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f(x) \end{array} \tag{2.6}$$

Unconstrained optimization will be the focus of Part II. Also, note that many algorithms are designed especially for this case, especially in Machine Learning. For example, the *gradient descent* algorithm is a very popular method for solving unconstrained optimization problems. However, as we will see in this lecture, some other algorithms are more efficient, such as the *Newton type methods*.

2.6 Non-Differentiable Optimization Problems

If one or more of the problem functions f, g, h are not differentiable in an optimization problem (2.1), we speak of a "non-differentiable" or "non-smooth" optimization problem. Non-differentiable optimization problems are much harder to solve than general NLPs.

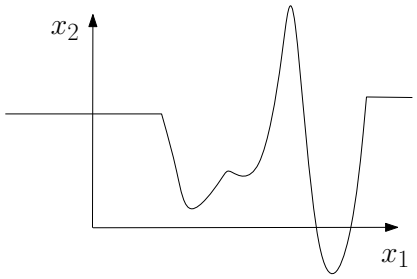


Figure 2.1: Visualization of a non-smooth objective.

A few gradient-free solvers exist (e.g., the Nelder-Mead method, genetic algorithms, etc.), but they are typically orders of magnitude slower than derivative-based methods, which are the focus of this course. On the other hand, some algorithms (such as the *subgradient method* or *proximal methods*)

can solve specific non-smooth problems, such as linear regression problems with L_1-regularization. In this course, we will not cover these methods, because we recommend reformulating nonsmooth problems into constrained but smooth problems. This is done by introducing *slack variables* and additional inequality constraints. An example of such a reformulation is given below.

Example 2.3: Reformulation of a smooth problem
Consider the following nonsmooth problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \max\{f_1(x), f_2(x)\}, \tag{2.7}$$

where $f_1(\cdot)$ and $f_2(\cdot)$ are two functions. Then, (2.7) can be reformulated as follows:

$$\begin{aligned} \underset{x\in\mathbb{R}^n, s\in\mathbb{R}}{\text{minimize}} \quad & s \\ \text{subject to} \quad & s \geq f_1(x), \\ & s \geq f_2(x). \end{aligned} \tag{2.8}$$

A lot of non-smooth problems can be reformulated similarly. In fact, many non-smooth functions can be expressed as the maximum between some smooth functions. It is the case for example for the absolute value: $|\varphi(x)| = \max\{\varphi(x), -\varphi(x)\}$.
However, there are also some non-smooth problems that are more difficult to solve. A prominent class of them is described below.

Definition 2.6: Mathematical programs with complementarity constraints
A Mathematical Programs with Complementarity Constraints (MPCC) is a problem where one of the constraints is a *complementarity constraint*. This type of constraint are written "$0 \leq p(x) \perp q(x) \geq 0$", which means that the two functions $p(x)$ and $q(x)$ have to take non-negative values, and for each component, at least one of the them should be zero:

$$0 \leq p(x) \perp q(x) \geq 0 \iff \begin{cases} & p(x) \geq 0, \\ \text{and} & q(x) \geq 0, \\ \text{and} & p(x)^\top q(x) = 0, \end{cases} \iff \forall i, \begin{cases} & p_i(x) = 0 \text{ and } q_i(x) \geq 0 \\ \text{or} & p_i(x) \geq 0 \text{ and } q_i(x) = 0. \end{cases} \tag{2.9}$$

MPCC are also considered as non-smooth problems. Numerical methods for solving these problems is an active research field. The solver `LCQpow` is practically very successful for solving quadratic programs with linear complementarity constraints.

2.7 Mixed-Integer Programming (MIP)

A Mixed-Integer Programming problem or Mixed-Integer Program (MIP) is a problem with both real and integer decision variables.

Definition 2.7: Mixed-Integer Programming (MIP)
A Mixed-Integer Program (MIP) is an optimization problem of the following form:

$$\begin{aligned} \underset{x \in \mathbb{R}^n,\, z \in \mathbb{Z}^m}{\text{minimize}} \quad & f(x, z) && \text{(2.10a)} \\ \text{subject to} \quad & g(x, z) = 0, && \text{(2.10b)} \\ & h(x, z) \geq 0. && \text{(2.10c)} \end{aligned}$$

A problem with only integer variables z is called an *Integer Program* (IP).

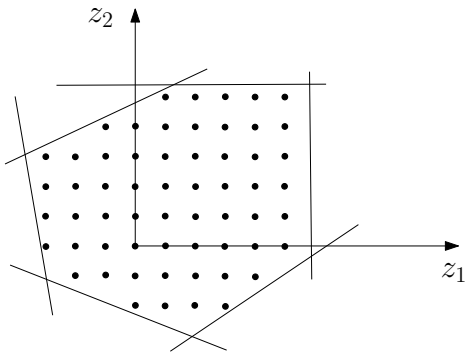


Figure 2.2: Visualization of the feasible set of an integer problem with linear constraints.

Mixed-integer programming problems are very common because they pop up every time there is a binary decision to be made. For example, in the problem of scheduling a set of tasks, we might have to decide whether to assign a task to a machine or not. We can classify MIPs into more specific classes:

i. A Mixed-Integer Linear Program (MILP) is a MIP where the functions f, g, h are affine in x and z. In this class, one might find a lot of real-world problems in logistics, such as the famous "travelling salesman problem". It is one of the largest research areas in discrete optimization.

ii. A Mixed-Integer Quadratic Program (MIQP) is a MIP where g, h are affine and f is convex quadratic in both x and z.

iii. A Mixed-Integer Nonlinear Program (MINLP) is a MIP that has no specific structure.

In general, all MIP problems are very hard to solve due to the combinatorial nature of the integer variables z. Common algorithms use the technique of *branch-and-bound* to solve them. *Branching* refers to splitting the feasible set into smaller branches (e.g., either $z_1 \geq 1$ or $z_1 \leq 0$), and *bounds* are lower-bounds based on *problem relaxations* (e.g. $\min_{z \in \mathbb{Z}^m} f(z) \geq \min_{z \in \mathbb{R}^m} f(z)$). Using these lower-bounds, one can often discard entire branches when the lower-bound on this branch is worse than the best solution found so far. Such algorithms have an exponential complexity in the worst case, but they are considerably better than pure enumeration of the possible integer points.

Software for solving MIPs: For MILPs or MIQPs, the commercial solvers `CPLEX` and `Gurobi` have shown impressive results. There are also some good free solvers such as `lp_solve`, `Cbc` or `HIGHS` for MILPs, and `SHOT`, `SCIP` or `CAMINO` for MINLPs.

Chapter 3

Convex Optimization

"The great watershed in optimization is not between linearity and nonlinearity, but convexity and nonconvexity"
Ralph Tyrrell Rockafellar.

An important class of optimization problems is the class of *convex optimization problems*. This includes LPs and convex QPs that we have seen in the previous chapter, but also more general problems as well. In order to define convex optimization problems and explain why they are so important, we first recall what a convex set and a convex function are.

3.1 Convex Functions

Definition 3.1: Convex Function
A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex when all secants are above the graph, i.e.,

$$\forall x, y \in \mathbb{R}^n, t \in [0,1] : \ f(x + t(y - x)) \leq f(x) + t(f(y) - f(x)). \tag{3.1}$$

Remark
Here we only defined convex functions on the whole space $\mathbb{R}^n$, but the same definition can be used when the domain is a convex subset of $\mathbb{R}^n$, which will be defined in the next section.

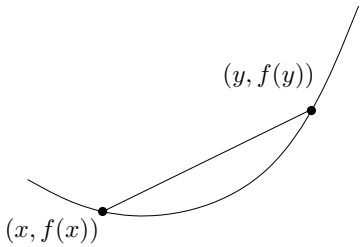


Figure 3.1: For a convex function, the line segment between any two points on the graph lies above the graph.

Definition 3.2: Concave Function
A function $f : \mathbb{R}^n \to \mathbb{R}$ is called "concave" if $x \mapsto -f(x)$ is convex.

Now that we defined convex functions, we will describe how to check if a function is convex. The following properties are useful for that purpose.

Theorem 3.1: Convexity for C^1 Functions
Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable. Then, f is convex if and only if its tangents lie below the graph, i.e.,

$$\forall x, y \in \mathbb{R}^n : \quad f(y) \geq f(x) + \nabla f(x)^\top (y - x). \tag{3.2}$$

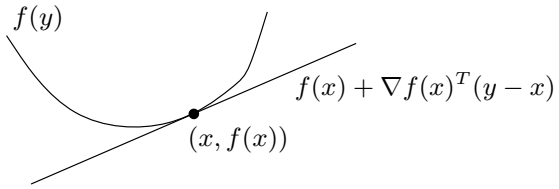


Figure 3.2: If f is convex and differentiable, then $f(y) \geq f(x) + \nabla f(x)^\top (y - x)$ for all $x, y \in \mathbb{R}^n$.

Proof. "$\Longrightarrow$" Simple manipulations of the definition of convexity in Eq. (3.1) yield:

$$\underbrace{\frac{f(x + t(y - x)) - f(x)}{t}}_{\xrightarrow[t\to 0]{} \nabla f(x)^\top (y-x)} \leq f(y) - f(x). \tag{3.3}$$

"$\Longleftarrow$" For $x, y \in \mathbb{R}^n$ and $t \in [0, 1]$, let us define $z \coloneqq x + t(y - x)$. Now apply Eq. (3.2) twice at z:

$$f(x) \geq f(z) + \nabla f(z)^\top (x - z), \tag{3.4a}$$
$$f(y) \geq f(z) + \nabla f(z)^\top (y - z). \tag{3.4b}$$

Now taking the linear combination "$(1 - t)$ (3.4a) $+ t$ (3.4b)", we find:

$$(1 - t)f(x) + tf(y) \geq f(z) + \nabla f(z)^\top \underbrace{\left[(1 - t)(x - z) + t(y - z)\right]}_{=(1-t)x+ty-z=0}. \tag{3.5}$$

□

Before diving into the next property, we will simply recall that for a symmetric matrix $P \in \mathbb{R}^{n\times n}$ we write $P \succcurlyeq 0$ when it is *positive semi-definite*, i.e., $\forall z \in \mathbb{R}^n : \; z^\top P z \geq 0$. For more details, we invite the reader to read the corresponding point in Section 1.6.

Theorem 3.2: Convexity for C^2 Functions
Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. Then f is convex if and only if its Hessian matrix is positive semi-definite everywhere, i.e.,

$$\forall x \in \mathbb{R}^n : \quad \nabla^2 f(x) \succcurlyeq 0. \tag{3.6}$$

Proof. For this proof, we will simply show the equivalence between the two conditions (3.2) and (3.6).

(3.2) $\implies$ (3.6): To prove this, we use a second-order Taylor expansion of f at x in an arbitrary direction p:

$$f(x+tp) = f(x) + t\nabla f(x)^\top p + \frac{1}{2}t^2 p^\top \nabla^2 f(x) p + o(t^2 \|p\|^2). \tag{3.7}$$

From this we obtain

$$p^\top \nabla^2 f(x) p = \lim_{t\to 0} \frac{2}{t^2} \underbrace{\Big(f(x+tp) - f(x) - t\nabla f(x)^\top p\Big)}_{\geq 0, \text{ because of (3.2)}}. \tag{3.8}$$

(3.6) $\implies$ (3.2): To prove this, we use the Taylor remainder term formula with some $\theta \in [0,1]$:

$$f(y) = f(x) + \nabla f(x)^\top (y-x) + \underbrace{\frac{1}{2}(y-x)^\top \nabla^2 f(x+\theta(y-x))(y-x)}_{\geq 0, \text{ due to (3.6)}}. \tag{3.9}$$

□

Example 3.1: Exponential Function
The function $f(x) = \exp(x)$ is convex because $f''(x) = \exp(x) \geq 0 \ \forall x \in \mathbb{R}$.

Example 3.2: Quadratic Function
The function $f(x) = c^\top x + \frac{1}{2}x^\top B x$ is convex if and only if $B \succcurlyeq 0$, because $\forall x \in \mathbb{R}^n$: $\nabla^2 f(x) = B$.

Example 3.3: An Interesting Convex Function
The function $f(x,t) = \frac{x^\top x}{t}$ is convex on the restricted domain $\Omega = \mathbb{R}^n \times (0,\infty)$ because its Hessian $\nabla^2 f(x,t) = \begin{bmatrix} \frac{2}{t}\mathbb{I}_n & -\frac{2}{t^2}x \\ -\frac{2}{t^2}x^\top & \frac{2}{t^3}x^\top x \end{bmatrix}$ is positive definite. To see this, multiply it from left and right with $v = (z^\top, s)^\top \in \mathbb{R}^{n+1}$ which yields $v^\top \nabla^2 f(x,t) v = \frac{2}{t^3}\|tz - sx\|_2^2 \geq 0$ if $t > 0$.

3.2 Convex Sets

Definition 3.3: Convex Set
A set $\Omega \subset \mathbb{R}^n$ is convex when all connecting lines lie inside the set, i.e.,

$$\forall x, y \in \Omega, t \in [0,1] : \ x + t(y-x) \in \Omega. \tag{3.10}$$

Now that we have defined what convex sets are, we will describe how to check if a set is convex. Note that in the next section, in Theorem 3.3, you will encounter another convenient way to check if a set is convex by checking if it is a sublevel set of a convex function.

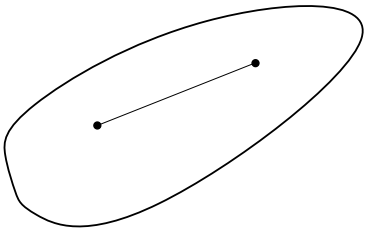


Figure 3.3: An example of a convex set

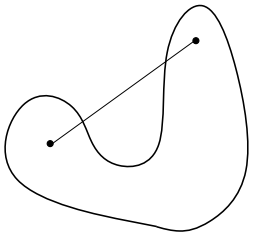


Figure 3.4: An example of a non-convex set

Connections between Convex Sets and Convex Functions

Remark
So far, we only defined convex functions on the domain $\mathbb{R}^n$, but the same definitions can be used when the domain is a convex set $\Omega \subset \mathbb{R}^n$.

Proposition 3.1: Characterization using Epigraphs
A function $f : \Omega \to \mathbb{R}$ is convex if and only if its Epigraph $\left\{(x, s) \in \Omega \times \mathbb{R} \;\middle|\; s \geq f(x)\right\}$ is a convex set.

This proposition is illustrated in Figure 3.5.

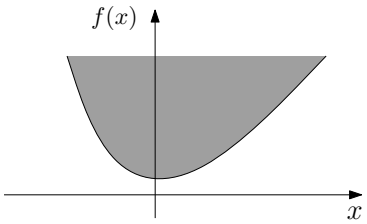


Figure 3.5: For a convex function, the Epigraph of the function (grey color) is convex.

Theorem 3.3: Convexity of Sublevel Sets
The sublevel sets of a convex function $f(\cdot)$, i.e., $\left\{x \in \Omega \;\middle|\; f(x) \leq c\right\}$ for $c \in \mathbb{R}$, are convex sets.

This theorem is illustrated in Figure 3.6.

Proof. If $f(x) \leq c$ and $f(y) \leq c$, then for any $t \in [0, 1]$ it also holds that

$$f((1-t)x + ty) \leq (1-t)f(x) + tf(y) \leq (1-t)c + tc = c. \tag{3.11}$$

□

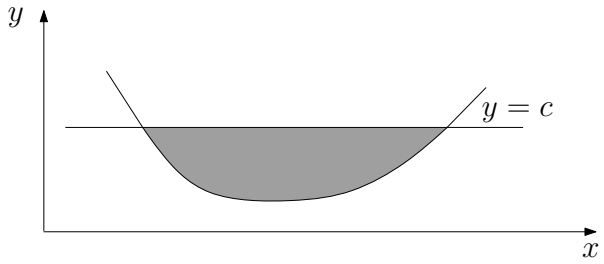


Figure 3.6: Convexity of sublevel sets.

3.3 Examples of convex sets

Example 3.4: The Set of Positive Semi-Definite Matrices
Let us define the vector space of symmetric matrices in $\mathbb{R}^{n\times n}$ as $\mathcal{S}^n = \left\{P \in \mathbb{R}^{n\times n} \;\middle|\; P = P^\top\right\}$ and the subset of positive semi-definite matrices as $\mathcal{S}^n_+ = \left\{P \in \mathcal{S}^n \;\middle|\; P \succcurlyeq 0\right\}$. This set is convex because it is the intersection of (infinitely many) sets of the form $\left\{P \in \mathcal{S}^n \;\middle|\; z^\top P z \geq 0\right\}$, which are convex.

Example 3.5: Linear Matrix Inequalities (LMI)
When $B_0, \ldots, B_m$ are some symmetric matrices, the inequality

$$B_0 + \sum_{i=1}^{m} B_i x_i \succcurlyeq 0, \tag{3.12}$$

is called a "linear matrix inequality (LMI)". It defines a convex set, as the pre-image of $\mathcal{S}^n_+$ under the affine map $x \mapsto B_0 + \sum_{i=1}^m B_i x_i$.

3.4 Convexity-preserving operations

The following properties are useful to manipulate convex functions and to construct new ones from old ones.

Proposition 3.2: Operations that Preserve Convexity of Functions
The following operations on convex functions preserve convexity:

- *Affine input transformations:* If $f(\cdot)$ is convex, then the function $x \mapsto f(Ax + b)$ is also convex.
- *Composition with a monotone convex function:* If $\Phi : \mathbb{R} \to \mathbb{R}$ is a convex and monotonically increasing function, and $f(\cdot)$ is a convex function, then $x \mapsto \Phi(f(x))$ is also convex.
- *Supremum of convex functions:* The supremum of (finitely or infinitely) many convex functions: if $f_p(x)$ are convex for all p in some set S, then the function $x \mapsto \sup_{p \in S} f_p(x)$ is also convex.

Proof. The proof is left as an exercise. □

Proposition 3.3: Operations that Preserve Convexity of Sets
Several operations on convex sets preserve their convexity:

- The intersection of finitely or infinitely many convex sets is convex.
- Affine image: if Ω is convex, then for $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$, the set $A\Omega + b \coloneqq \left\{Ax + b \;\middle|\; x \in \Omega\right\} = \left\{y \in \mathbb{R}^m \;\middle|\; \exists x \in \Omega : \; y = Ax + b\right\}$ is convex.
- Affine pre-image: if Ω is convex, then for $A \in \mathbb{R}^{n \times m}, b \in \mathbb{R}^n$, the set $\left\{z \in \mathbb{R}^m \;\middle|\; Az + b \in \Omega\right\}$ is convex.

Proof. The proof is left as an exercise. □

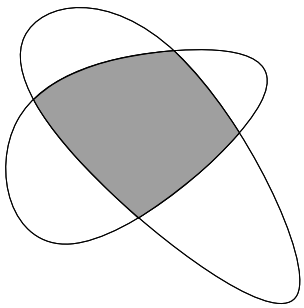


Figure 3.7: The intersection of finitely or infinitely many convex sets is convex

3.5 Convex Optimization Problems

Now that we have the mathematical tools for it, let us dive into convex optimization problems with their definitions and the most important properties.

3.5.1 What is a convex optimization problem?

Definition 3.4: Convex Optimization Problem
A convex optimization problem is a problem in the general form

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(3.13a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(3.13b)} \\ & h(x) \geq 0, && \text{(3.13c)} \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function, $g : \Omega \to \mathbb{R}^m$ is an affine function, and each component of $h : \Omega \to \mathbb{R}^p$ is a concave function (i.e., each function $x \mapsto -h_i(x)$ is convex).

Remark
In the literature, convex optimization problems are often written in a slightly different form, as follows:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f_0(x) && \text{(3.14a)} \\ \text{subject to} \quad & Ax = b, && \text{(3.14b)} \\ & f_i(x) \leq 0, \quad i = 1, \dots, q, && \text{(3.14c)} \end{aligned}$$

where $f_0(\cdot), f_1(\cdot), \dots, f_q(\cdot)$ are convex functions.

Proposition 3.4: Convexity of the feasible set
The feasible set of a convex problem $\Omega \coloneqq \left\{x \in \mathbb{R}^n \;\middle|\; g(x) = 0, h(x) \geq 0\right\}$ is convex.

Proof. Let us decompose the feasible set into an intersection of sublevel sets:

$$\Omega = \left(\textstyle\bigcap_{j=1}^{p} \left\{x \in \mathbb{R}^n \;\middle|\; g_j(x) \leq 0\right\}\right) \cap \left(\textstyle\bigcap_{j=1}^{p} \left\{x \in \mathbb{R}^n \;\middle|\; -g_j(x) \leq 0\right\}\right) \cap \left(\textstyle\bigcap_{i=1}^{q} \left\{x \in \mathbb{R}^n \;\middle|\; -h_i(x) \leq 0\right\}\right). \tag{3.15}$$

Then, note that the functions $g_j(\cdot)$ and $-g_i(\cdot)$ are affine so they are convex. The functions $-h_i(\cdot)$ are also convex because $h_i(\cdot)$ is concave. Therefore, the feasible set Ω is convex because it is the intersection of sublevel sets of convex functions. □

Remark
A more general definition is to only impose that the feasible set Ω is convex and that the objective is convex. For example, $(x_1 + x_2)^3 = 1$ is, strictly speaking, a convex constraint but only by "accident". Because this definition would be more difficult to manipulate, we prefer the one given above.

Remark
A maximization problem of the form

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{maximize}} \quad & f(x) \\ \text{subject to} \quad & x \in \Omega, \end{aligned}$$

is called a "convex maximization problem" if Ω is convex and f concave. In general, we will prefer reformulating them as a minimization problem as follows.

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & -f(x) \\ \text{subject to} \quad & x \in \Omega. \end{aligned}$$

Example 3.6: Quadratically Constrained Quadratic Program (QCQP)
A convex optimization problem of the form (3.14) with $f_i(x) = d_i + c_i^\top x + \frac{1}{2} x^\top B_i x$ with $B_i \succcurlyeq 0$ for $i = 0, 1, \ldots, m$ is called a "Quadratically Constrained Quadratic Program (QCQP)".

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c_0^\top x + \frac{1}{2} x^\top B_0 x && \text{(3.16a)} \\ \text{subject to} \quad & Ax = b, && \text{(3.16b)} \\ & d_i + c_i^\top x + \frac{1}{2} x^\top B_i x \leq 0, \quad i = 1, \ldots, m. && \text{(3.16c)} \end{aligned}$$

By choosing $B_1 = \ldots = B_m = 0$ we would obtain a usual QP, and by also setting $B_0 = 0$ we would obtain an LP. Therefore, the class of QCQPs contains both LPs and QPs as subclasses.

CVX: an important tool for convex optimization An excellent tool to formulate and solve convex optimization problems is `CVX`, which is an open-source software available in Python (`CVXPY`) or MATLAB.

3.6 Semidefinite Programming (SDP)

An interesting class of convex optimization problems makes use of LMI (see Example 3.5) as inequality constraints.

Definition 3.5: Semidefinite Programming (SDP)
A Semidefinite Program (SDP) is an optimization problem of the following form:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^\top x && \text{(3.17a)} \\ \text{subject to} \quad & Ax - b = 0, && \text{(3.17b)} \\ & B_0 + \sum_{i=1}^{n} B_i x_i \succcurlyeq 0, && \text{(3.17c)} \end{aligned}$$

where the matrices $B_0, \ldots, B_m \in \mathbb{R}^{k \times k}$ are all symmetric.

SDPs also draw a lot of attention in the literature, mainly because of the two following properties.

Proposition 3.5: SDPs are convex
SDP problems are convex optimization problems.

Proposition 3.6: A lot of problems can be reformulated as SDPs
All LPs, convex QPs, and convex QCQPs can be reformulated as SDPs.

As mentioned above, an algorithm that solves SDPs can also be used to solve LPs and convex QPs, but not only. Indeed, many other problems are SDPs but not necessarily a QP or a QCQP, as the following example suggests.

Example 3.7: Minimizing Largest Eigenvalue
Consider the following optimization problem, where the goal is to minimize the largest eigenvalue of a symmetric matrix that depends affinely on the optimization variable $x \in \mathbb{R}^n$:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \lambda_{\max}\left(B_0 + \sum_{i=1}^{n} B_i x_i \right) \tag{3.18}$$

We can reformulate this problem as an SDP by adding a slack variable $s \in \mathbb{R}$, as follows:

$$\begin{aligned} \underset{s \in \mathbb{R},\, x \in \mathbb{R}^n}{\text{minimize}} \quad & s && \text{(3.19a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(3.19b)} \\ & \mathbb{I}_k s - \sum_{i=1}^{n} B_i x_i - B_0 \succcurlyeq 0. && \text{(3.19c)} \end{aligned}$$

3.7 Optimality Characterization for Convex Problems

Theorem 3.4: Local Implies Global Optimality for Convex Problems
For a convex optimization problem, every local minimizer is also a global minimizer.

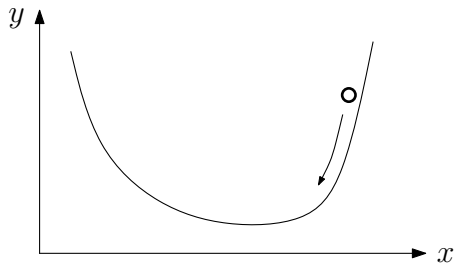


Figure 3.8: Every local minimum is also a global one for a convex function.

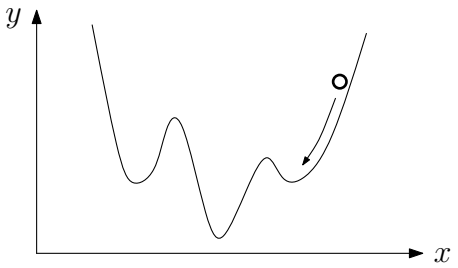


Figure 3.9: Not every local minimum is also a global one for this nonconvex function.

Proof. Regard Figure 3.10 for a visualization of the proof.
Let $x^\star$ be a local minimizer of the problem:

$$\begin{aligned} &\underset{x \in \mathbb{R}^n}{\text{minimize}} && f(x) && \text{(3.20a)} \\ &\text{subject to} && x \in \Omega. && \text{(3.20b)} \end{aligned}$$

Because of local optimality, there exists a neighborhood $\mathcal{N}$ of $x^\star$ so that for all $\tilde{x} \in \Omega \cap \mathcal{N}$, it holds that $f(\tilde{x}) \geq f(x^\star)$.
Now, let $y \in \Omega$ be an arbitrary feasible point, and consider the connecting line between $x^\star$ and y (choose $y \neq x^\star$). This line is completely contained in Ω due to its convexity. Let $\tilde{x} = x^\star + t(y - x^\star)$ be on this line and also in the neighborhood $\mathcal{N}$ (i.e., $t > 0$ is small enough). Due to local optimality, we have $f(x^\star) \leq f(\tilde{x})$, and due to convexity we have $f(\tilde{x}) \leq f(x^\star) + t(f(y) - f(x^\star))$. Putting together these inequalities, we find $t(f(y) - f(x^\star)) \geq 0$, which implies $f(y) \geq f(x^\star)$. Since this is the case for any $y \in \Omega$, we have proven global optimality. □

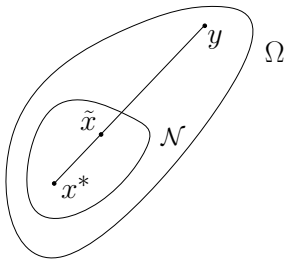


Figure 3.10: Visualization for the proof of Theorem 3.4.

Theorem 3.5: Optimality Characterization for Smooth Convex Problems
Consider the convex optimization problem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & x \in \Omega, \end{aligned}$$

with a continuously differentiable objective function f. A point $x^\star \in \Omega$ is a global optimizer if and only if

$$\forall y \in \Omega : \quad \nabla f(x^\star)^\top (y - x^\star) \geq 0. \tag{3.21}$$

Proof. "(3.21) $\implies$ global optimality": Due to the C^1 characterization of convexity of f in Eq. (3.2), we have for any feasible $y \in \Omega$

$$f(y) \geq f(x^\star) + \underbrace{\nabla f(x^\star)^\top (y - x^\star)}_{\geq 0} \geq f(x^\star). \tag{3.22}$$

"optimality $\implies$ (3.21)": Assume for contradiction that there exists a $y \in \Omega$ with $\nabla f(x^\star)^\top (y - x^\star) < 0$. Then, we could consider a Taylor expansion

$$f(x^\star + t(y - x^\star)) = f(x^\star) + t \underbrace{\nabla f(x^\star)^\top (y - x^\star)}_{<0} + \underbrace{o(t)}_{\to 0}, \tag{3.23}$$

yielding $f(x^\star + t(y - x^\star)) < f(x^\star)$ for $t > 0$ small enough to let the last term be dominated by the second last one. Thus, $x^\star$ would not be a minimizer.

□

Corollary 3.1: Unconstrained Convex Problems
Consider the unconstrained convex problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \tag{3.24}$$

where f is convex and continuously differentiable. Then $x^\star$ is a minimizer if and only if it satisfies

$$\nabla f(x^\star) = 0. \tag{3.25}$$

Example 3.8: Unconstrained Quadratic
Consider the unconstrained problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad c^\top x + \frac{1}{2} x^\top B x \tag{3.26}$$

with $B \succ 0$. The condition $0 = \nabla f(x^\star) = c + Bx^\star$ allows us to solve this problem:

$$x^\star \coloneqq \arg\min_{x \in \mathbb{R}^n} c^\top x + \frac{1}{2} x^\top B x = -B^{-1} c, \tag{3.27a}$$

$$f(x^\star) \coloneqq \min_{x \in \mathbb{R}^n} c^\top x + \frac{1}{2} x^\top B x = -\frac{1}{2} c^\top B^{-1} c. \tag{3.27b}$$

Note that these basic results will often be used in the following chapters.

Chapter 4

The Lagrangian Function and Duality

In this chapter, we deal with a general NLP (not necessarily convex)

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(4.1a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(4.1b)} \\ & h(x) \geq 0, && \text{(4.1c)} \end{aligned}$$

with functions $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^p$, and $h : \mathbb{R}^n \to \mathbb{R}^q$:

The Lagrangian Function

Definition 4.1: Lagrangian Function and Lagrange Multipliers
We define the so-called "Lagrangian function" as follows:

$$\mathcal{L}(x, \lambda, \mu) \coloneqq f(x) - \lambda^\top g(x) - \mu^\top h(x), \tag{4.2}$$

where $\lambda \in \mathbb{R}^p$ and $\mu \in \mathbb{R}^q$ are the so-called "Lagrange multipliers" or "dual variables". We typically require the inequality multipliers μ to be non-negative, $\mu \geq 0$, while the sign of the equality multipliers λ is arbitrary.

The Lagrangian function plays a crucial role in both convex and general nonlinear optimization. We will also make the following lemma that justifies why we require μ to be non-negative.

Lemma 4.1: Lower Bound Property of Lagrangian
If $\tilde{x}$ is a feasible point of (4.10) and $\mu \geq 0$, then

$$\mathcal{L}(\tilde{x}, \lambda, \mu) \leq f(\tilde{x}). \tag{4.3}$$

Proof.

$$\mathcal{L}(\tilde{x}, \lambda, \mu) = f(\tilde{x}) - \lambda^\top \underbrace{g(\tilde{x})}_{=0} - \underbrace{\mu^\top}_{\geq 0} \underbrace{h(\tilde{x})}_{\geq 0} \leq f(\tilde{x}). \tag{4.4}$$

□

This lemma allows us to prove the following theorem, which draws a connection between the Lagrangian and the original optimization problem.

Theorem 4.1: The supremum of the Lagrangian
The following identity holds:

$$\sup_{\lambda,\mu\geq 0} \mathcal{L}(x,\lambda,\mu) = \begin{cases} f(x) & \text{if } x \text{ is feasible for (4.1),} \\ +\infty & \text{else.} \end{cases} \tag{4.5}$$

Proof. Let $x \in \mathbb{R}^n$ be a vector. We prove equation (4.5) by separating the problem into the three (possibly intersecting) following cases.

i. If x violates one of the inequality constraints, i.e., $h_i(x) < 0$ for some i, then $\mathcal{L}(x,\lambda,\mu) \underset{\mu_i\to+\infty}{\longrightarrow} +\infty$, which implies that $\sup_{\lambda,\mu\geq 0} \mathcal{L}(x,\lambda,\mu) = +\infty$, i.e., equation (4.5) holds.

ii. If x violates one of the equality constraints, i.e., $g_j(x) \neq 0$ for some j, then depending on the sign of $g_j(x)$, we have either $\mathcal{L}(x,\lambda,\mu) \underset{\lambda_j\to-\infty}{\longrightarrow} +\infty$ or $\mathcal{L}(x,\lambda,\mu) \underset{\lambda_j\to+\infty}{\longrightarrow} +\infty$. Again, this implies that $\sup_{\lambda,\mu\geq 0} \mathcal{L}(x,\lambda,\mu) = +\infty$, hence equation (4.5) holds.

iii. If x is feasible, then using Lemma 4.1 and the fact that $\mathcal{L}(x,0,0) = f(x)$, we find $\sup_{\lambda,\mu\geq 0} \mathcal{L}(x,\lambda,\mu) = \mathcal{L}(x,0,0) = f(x)$. Therefore, equation (4.5) still holds.

Since every case is covered, this concludes the proof. □

Corollary 4.1: Connection between the Lagrangian and the original optimization problem
The original optimization problem (4.1) is equivalent to the following "min-max" problem:

$$\underset{x\,\in\,\mathbb{R}^n}{\text{minimize}} \quad \sup_{\lambda,\mu\geq 0} \mathcal{L}(x,\lambda,\mu) \tag{4.6}$$

4.1 The Dual Problem and Weak Duality

The concept of *duality* is a very powerful concept in optimization. Some claim that the Hungarian/American economist and mathematician John von Neumann conjectured the principle of duality for linear programming, but the first rigorous and published proofs of the main principles were published by the Canadian mathematician Albert William Tucker in 1948.

Definition 4.2: Lagrange Dual Function
We define the so-called "Lagrange dual function" as the unconstrained infimum of the Lagrangian over x for fixed multipliers λ, μ:

$$q(\lambda,\mu) \coloneqq \inf_{x\in\mathbb{R}^n} \mathcal{L}(x,\lambda,\mu) \in \mathbb{R} \cup \{-\infty\}. \tag{4.7}$$

This function will often take the value $-\infty$, in which case we will say that the pair (λ, μ) is "dual infeasible" for reasons that we motivate in the last example of this subsection.

Lemma 4.2: Lower Bound Property of Lagrange Dual
If $\mu \geq 0$, then

$$q(\lambda, \mu) \leq p^\star. \tag{4.8}$$

Proof. The lemma is an immediate consequence of Eq. (4.3), which implies that for any feasible $\tilde{x}$ holds $q(\lambda, \mu) \leq f(\tilde{x})$. This inequality holds in particular for the global minimizer $x^\star$ (which must be feasible), yielding $q(\lambda, \mu) \leq f(x^\star) = p^\star$. □

Theorem 4.2: Concavity of Lagrange Dual
The function $q : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R} \cup \{-\infty\}$ is concave (even if the NLP is not convex).

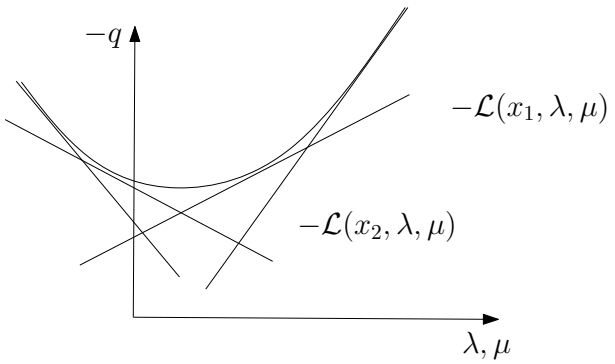


Figure 4.1: The negative dual function $-q$ is the supremum of linear functions in λ and μ, hence convex.

Proof. We will show that $-q$ is convex. The Lagrangian $\mathcal{L}$ is an affine function in the multipliers λ and μ, which in particular implies that $-\mathcal{L}$ is convex in (λ, μ). Thus, the function $-q(\lambda, \mu) = \sup_x -\mathcal{L}(x, \lambda, \mu)$ is the supremum of convex functions in (λ, μ) that are indexed by x, and therefore convex. □

A natural question to ask is what is the best lower-bound that we can get from the Lagrange dual function. We obtain it by maximizing the Lagrange dual over all possible multiplier values, yielding the so-called "dual problem".

Definition 4.3: Dual Problem
The "dual problem" is defined as the convex maximization problem:

$$\underset{\lambda \in \mathbb{R}^p, \mu \in \mathbb{R}^q}{\text{maximize}} \quad \underbrace{\inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu)}_{=q(\lambda,\mu)} \tag{4.9a}$$

$$\text{subject to} \quad \mu \geq 0. \tag{4.9b}$$

The optimal value $d^\star$ of this optimization problem is called the "dual optimal value".

Remark
We will denote the globally optimal value of the objective function subject to the constraints as the "primal optimal value" by $p^\star$:

$$p^\star = f(x^\star), \tag{4.10}$$

where $x^\star$ is a solution of (4.1). We will also denote the original problem as the "primal optimization problem".

Remark
It is interesting to notice that the dual problem looks similar to the reformulated primal problem in (4.6) except that in the dual problem, we *first minimize over x then maximize over λ, μ* while in the primal problem, we *first maximize over λ, μ then minimize over x*.

It is also interesting to note that the dual problem is always convex, even if the so-called "primal problem" is not. As an immediate consequence of the last lemma, we obtain a very fundamental result that is called "weak duality".

Theorem 4.3: Weak Duality

$$d^\star \leq p^\star. \tag{4.11}$$

This theorem holds for any arbitrary optimization problem, but does only unfold its full strength in convex optimization, where a stronger result holds as we will see in the next section.

4.2 Strong Duality for Convex Problems

In order to state the strong duality theorem, we need to introduce an additional assumption, which is satisfied for *almost all* feasible convex optimization problems.

Definition 4.4: The Slater condition
The "Slater condition" for a convex problem states that there exists a feasible point for which the *nonlinear inequality constraints are inactive*, i.e.:

$$\text{there exists } \bar{x} \in \mathbb{R}^n \text{ such that:} \tag{4.12}$$

$$g(\bar{x}) = 0 \quad \text{and} \quad \text{for all } i = 1, \dots, q : \begin{cases} h_i(\bar{x}) \geq 0, & \text{when } h_i(\cdot) \text{ is affine,} \\ h_i(\bar{x}) > 0, & \text{when } h_i(\cdot) \text{ is concave but nonlinear.} \end{cases}$$

The Slater condition is notably satisfied for all feasible LPs and QPs, and for any convex problem with affine constraints only.
We will just cite the important result here, without proof.

Theorem 4.4: Strong Duality
If the primal optimization problem (4.10) is convex and the so-called "Slater condition" holds, then primal and dual objective are equal to each other,

$$d^\star = p^\star. \tag{4.13}$$

Furthermore, the solutions of the primal problem can be retrieved from the solution of the dual problem:

$$x^\star = \arg\min_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda^\star, \mu^\star), \tag{4.14}$$

where $(\lambda^\star, \mu^\star)$ is a solution of the dual problem (4.9).

The proof of the theorem can be found for example in [2].
Strong duality allows us to reformulate a convex optimization problem into its dual, which looks very different but gives the same solution. We will look at this at hand of two examples.

Dual of an LP

Let us now regard the following LP:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^\top x && \text{(4.15a)} \\ \text{subject to} \quad & Ax - b = 0, && \text{(4.15b)} \\ & Cx - d \geq 0. && \text{(4.15c)} \end{aligned}$$

Its Lagrangian function is given by

$$\begin{aligned} \mathcal{L}(x, \lambda, \mu) &= c^\top x - \lambda^\top (Ax - b) - \mu^\top (Cx - d), \\ &= \lambda^\top b + \mu^\top d + \left(c - A^\top \lambda - C^\top \mu\right)^\top x. \end{aligned}$$

Thus, the Lagrange dual is

$$q(\lambda, \mu) = \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu) = \lambda^\top b + \mu^\top d + \begin{cases} 0 & \text{if } \; c - A^\top \lambda - C^\top \mu = 0, \\ -\infty & \text{else.} \end{cases}$$

Thus, the objective function $q(\lambda, \mu)$ of the dual optimization problem is $-\infty$ at all points that do not satisfy the linear equality $c - A^\top \lambda - C^\top \mu = 0$. As we want to maximize, these points can be regarded as infeasible points of the dual problem (that is why we call them "dual infeasible"), and we can explicitly write the dual of the above LP (4.15) as

$$\begin{aligned} \underset{\lambda \in \mathbb{R}^p, \mu \in \mathbb{R}^q}{\text{maximize}} \quad & \lambda^\top b + \mu^\top d && \text{(4.16a)} \\ \text{subject to} \quad & A^\top \lambda + C^\top \mu = c, && \text{(4.16b)} \\ & \mu \geq 0. && \text{(4.16c)} \end{aligned}$$

This is again an LP, and strong duality holds for all LPs for which at least one feasible point exists, i.e., $d^\star = p^\star$.

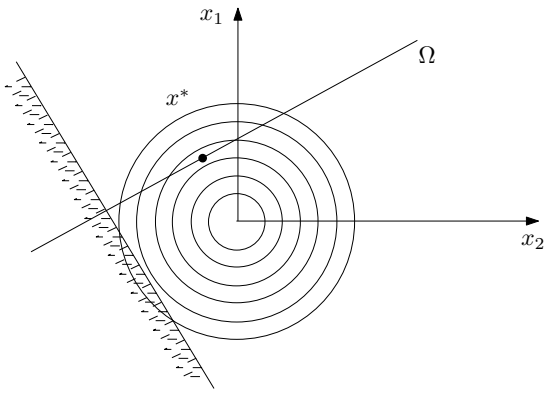


Figure 4.2: Illustration of a QP in two variables with one equality and one inequality constraint.

Dual of a strictly convex QP

We regard the following strictly convex QP (i.e., with $B \succ 0$):

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^\top x + \frac{1}{2} x^\top B x && \text{(4.17a)} \\ \text{subject to} \quad & Ax - b = 0, && \text{(4.17b)} \\ & Cx - d \geq 0. && \text{(4.17c)} \end{aligned}$$

This is illustrated in Figure 4.2.
Its Lagrangian function is given by

$$\begin{aligned} \mathcal{L}(x, \lambda, \mu) &= c^\top x + \frac{1}{2} x^\top B x - \lambda^\top (Ax - b) - \mu^\top (Cx - d), \\ &= \lambda^\top b + \mu^\top d + \frac{1}{2} x^\top B x + \left(c - A^\top \lambda - C^\top \mu\right)^\top x. \end{aligned}$$

The Lagrange dual function is the infimum value of the Lagrangian with respect to x, which only enters the last two terms in the above expression. We obtain

$$\begin{aligned} q(\lambda,\mu) &= \lambda^\top b + \mu^\top d + \inf_{x\in\mathbb{R}^n} \left(\frac{1}{2}x^\top Bx + \Big(c - A^\top\lambda - C^\top\mu\Big)^\top x\right) \\ &= \lambda^\top b + \mu^\top d - \frac{1}{2}\Big(c - A^\top\lambda - C^\top\mu\Big)^\top B^{-1}\Big(c - A^\top\lambda - C^\top\mu\Big). \end{aligned}$$

Therefore, the dual optimization problem of the QP (4.17) is given by

$$\begin{aligned} \underset{\lambda\in\mathbb{R}^p,\,\mu\in\mathbb{R}^q}{\text{maximize}} \quad & \lambda^\top b + \mu^\top d - \frac{1}{2}\Big(c - A^\top\lambda - C^\top\mu\Big)^\top B^{-1}\Big(c - A^\top\lambda - C^\top\mu\Big) && \text{(4.18a)} \\ \text{subject to} \quad & \mu \geq 0. && \text{(4.18b)} \end{aligned}$$

Due to the fact that the objective is concave, this problem is again a convex QP, but not a strictly convex one.
Note that one should carefully keep the first term even though it is constant, to keep the equation $d^\star = p^\star$ correct.

Interpretations of the Dual Problem

In this section, we will illustrate the meaning of the dual problem with one property and one example.

Interpretation of the multipliers as the "price induced by the constraint"

Consider the following problem, where $h(x)$ is scalar, i.e., there is only one constraint:

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(4.19a)} \\ \text{subject to} \quad & h(x) \geq 0. && \text{(4.19b)} \end{aligned}$$

Then, the solution $\mu^\star$ of the dual problem can be interpreted as *the change in the objective value if one would perturb the constraint by one unit.*
More precisely, consider the value $p^\star$ of problem 4.19, and compare it with the value $p^\star_\varepsilon$ of the following perturbation of the problem:

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(4.20a)} \\ \text{subject to} \quad & h(x) \geq \varepsilon. && \text{(4.20b)} \end{aligned}$$

Then, the multiplier $\mu^\star$ has the following interpretation:

$$\mu^\star = \lim_{\varepsilon\to 0} \frac{p^\star_\varepsilon - p^\star}{\varepsilon} \geq 0. \tag{4.21}$$

Note that similar properties can be derived when there are many constraints.

The instructive example of resource allocation

We consider a problem of resource allocation among n agents. For example, this would emerge in the real-world when many agents share a common resource (such as water, oil, electricity, etc.) and a *global agent* looks for the resource distribution that minimizes the average cost of the agent. The problem of finding the best allocation is modeled with the following minimization problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \sum_{i=1}^{n} c_i(x_i) \tag{4.22a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_i \leq r, \tag{4.22b}$$

where $r > 0$ is the total amount of resources, $x_i \in \mathbb{R}$ is the amount of resource allocated to agent i, and $c_i(x_i)$ is the cost function of agent i, which is assumed to be convex.
Depending on the size of n and the complexity of the individual cost functions $c_i(\cdot)$, this can be a difficult problem to solve. Moreover, it might be difficult for a central distributor to know the individual cost functions. It turns out that duality theory delivers an elegant way to facilitate a decentralized solution to the optimization problem that is seamlessly used in real life. The Lagrangian of the problem is given by:

$$\mathcal{L}(x, \mu) = \sum_{i=1}^{n} c_i(x_i) - \mu \left(r - \sum_{i=1}^{n} x_i \right) = \sum_{i=1}^{n} \left(c_i(x_i) + \mu x_i \right) - r\mu, \tag{4.23}$$

where $\mu \geq 0$ is the Lagrange multiplier for the constraint $\sum_{i=1}^{n} x_i \leq r$.
The Lagrange dual function $q(\cdot)$ is given by:

$$q(\mu) = \inf_x \mathcal{L}(x, \mu) = \inf_x \left(\sum_{i=1}^{n} \left(c_i(x_i) + \mu x_i \right) \right) - r\mu = \sum_{i=1}^{n} \inf_{x_i} \left(c_i(x_i) + \mu x_i \right) - r\mu. \tag{4.24}$$

Since strong duality holds (the problem is convex and the constraints are affine), the solutions $x_i^\star$ are given by the individual unconstrained optimization problems:

$$\underset{x_i \in \mathbb{R}}{\text{minimize}} \quad c_i(x_i) + \mu^\star x_i, \tag{4.25}$$

where $\mu^\star$ is the solution of the dual problem $\max_{\mu \geq 0} q(\mu)$.
It is interesting that the allocation given by (4.25) is exactly what would happen if each actor chose their amount of resource freely, but have to pay a price of $\mu^\star$ for each unit of resource. For this reason, the Lagrange multipliers are sometimes called *shadow prices* in economics.
A good algorithm for solving (4.22) would be to iteratively adjust the multiplier μ until the solution given by the individual problems (4.25) minimizes $\mathcal{L}(x, \mu)$. This algorithm (sometimes known as the *dual decomposition method*) would "magically" find the optimal solution that satisfies the global constraint $\sum_{i=1}^{n} x_i \leq r$ while this constraint would never explicitly be checked. Note that this is often what happens seamlessly in real life, where the price of goods is adjusted such that demand and supply match.
It is remarkable that at each iteration of this algorithm, the problems (4.25) can be solved in parallel by the individual agents, which can make the algorithm computationally efficient.

Part II

Unconstrained Optimization and Newton Type Algorithms

Chapter 5

Optimality Conditions

In this part of the course, we regard unconstrained smooth optimization problems, i.e., problems that are of the following form

$$\underset{x \in D}{\text{minimize}} \quad f(x), \tag{5.1a}$$

where the objective function $f : D \to \mathbb{R}$ is continuously differentiable on an open domain $D \subset \mathbb{R}^n$. Most of the time, we choose $D = \mathbb{R}^n$ but not always, we might consider examples as follows:

$$\underset{x \in (0,\infty)}{\text{minimize}} \quad \frac{1}{x} + x. \tag{5.2}$$

Note that we are only interested in minimizers that lie *inside* D. This simplifies considerably the optimality conditions and the optimization algorithms. In the next part (i.e., Part III), we will analyze the case where the minimizers may lie on the *border* of the feasible set.

5.1 Necessary Optimality Conditions

First, let us state some *necessary* conditions, i.e., some conditions that are satisfied for *every* minimizer.

Theorem 5.1: First Order Necessary Conditions (FONC)
If $x^\star \in D$ is a local minimizer of (5.1), then:

$$\nabla f(x^\star) = 0. \tag{5.3}$$

Proof. Since $x^\star$ is a local minimizer (and D is open), for any direction $p \in \mathbb{R}^n$, and for $t \in \mathbb{R}$ small enough, we have $f(x^\star + tp) \geq f(x^\star)$. Now, using the definition of the derivative, we can write:

$$\nabla f(x^\star)^\top p = \lim_{t \to 0^+} \frac{f(x^\star + tp) - f(x^\star)}{t} \geq 0. \tag{5.4}$$

By choosing $p = -\nabla f(x^\star)$, we obtain $-\|\nabla f(x^\star)\|^2 = -\nabla f(x^\star)^\top \nabla f(x^\star) \geq 0$, which proves equation (5.3). □

Definition 5.1: Stationary points
In this part of the course, we will call *stationary points* the points $x^\star$ that satisfy the first-order necessary condition (5.3). Note that a more general definition of stationary points exists for constrained optimization problems.

Theorem 5.2: Second Order Necessary Conditions (SONC)
If $x^\star \in D$ is a local minimizer of $f : D \to \mathbb{R}$ and $f \in C^2$, then:

$$\nabla f(x^\star) = 0, \tag{5.5a}$$
$$\nabla^2 f(x^\star) \succcurlyeq 0. \tag{5.5b}$$

Proof. If $x^\star$ is a local minimizer, then it is a stationary point, i.e., $\nabla f(x^\star) = 0$. This allows us to write, using Taylor's expansion:

$$\forall p \in \mathbb{R}^n, \quad \frac{1}{2} t^2 p^\top \nabla^2 f(x^\star) p = \lim_{t \to 0} \frac{f(x^\star + tp) - f(x^\star)}{t^2} \geq 0. \tag{5.6}$$

By definition of positive semi-definite matrices, this is equivalent to (5.5b). □

Note that the second-order necessary conditions (5.5) are not sufficient for $x^\star$ to be a minimizer. This is illustrated by the function $f(x) = x^3$ or $f(x) = -x^4$, which are saddle points and maximizers respectively, both fulfilling SONC.

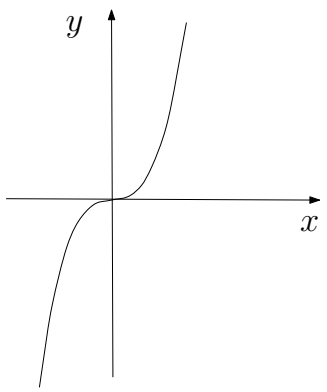


Figure 5.1: Stationary points are not always optimal.

5.2 Sufficient Optimality Conditions

For convex functions, we have already proven in Corollary 3.1 that not only the first-order optimality condition is necessary, but it is also sufficient for optimality.
The question remains: for general nonlinear but smooth functions f, how can we certify that a point $x^\star \in D$ is a minimizer of f?
Unfortunately, for nonconvex problems, it is very hard to certify that a point is a global minimizer. However, the following theorem allows us to certify *local optimality.*

Theorem 5.3: Second Order Sufficient Conditions (SOSC)
If the following equations hold:

$$\nabla f(x^\star) = 0, \tag{5.7a}$$
$$\nabla^2 f(x^\star) \succ 0, \tag{5.7b}$$

then $x^\star$ is a strict local minimizer of $f(\cdot)$.

Proof. Let $x^\star \in D$ be a point that satisfies the conditions (5.7). Then there exists a neighborhood $\mathcal{N}$ of $x^\star$ so that for all $x \in \mathcal{N}$, $\nabla^2 f(x) \succ 0$. Now let $x \in \mathcal{N} \backslash \{x^\star\}$ be in that neighborhood. A second-order Taylor expansion gives that for some $\theta \in (0, 1)$, we have:

$$f(x) = f(x^\star) + \underbrace{\nabla f(x^\star)^\top (x - x^\star)}_{=0} + \frac{1}{2} \underbrace{(x - x^\star)^\top \nabla^2 f(x^\star + \theta(x - x^\star))(x - x^\star)}_{>0} > f(x^\star), \tag{5.8}$$

where we used the fact that $x^\star + \theta(x - x^\star) \in \mathcal{N}$. This is true for any $x \in \mathcal{N} \backslash \{x^\star\}$, hence, $x^\star$ is a strict local minimizer of f. □

Note that the second-order sufficient condition (5.7) is not necessary for a stationary point $x^\star$ to be a strict local minimizer. This is illustrated by the function $f(x) = x^4$, for which $x^\star = 0$ is a strict local minimizer with $\nabla^2 f(x^\star) = 0$.

5.3 Perturbation Analysis

In this part, we analyze the *sensitivity* of the solution of some optimization problem with respect to the objective function. More precisely, we are interested in how the solution changes when we slightly perturb the problem. We consider the solution $x_\varepsilon^\star$ of the perturbed problem:

$$\begin{array}{ll} \text{minimize} \\ x \in D \end{array} \quad f(x) + \varepsilon \phi(x), \tag{5.9}$$

where $\phi : D \to \mathbb{R}$ is a function, and $\varepsilon > 0$ is a small perturbation parameter. The goal is to analyze the behavior of the solution $x_\varepsilon^\star$ as $\varepsilon \to 0$.

Remark
When computing a function numerically, we often have to deal with *numerical errors*. Indeed, computers can evaluate (most of the) functions only up to a certain precision, which is called the *machine precision*. This motivates the study of the sensitivity of the solution with respect to perturbations of the problem. Indeed, the inaccuracy of the numerical evaluations of $f(x)$ can be viewed as a perturbation of the problem.

The following theorem gives a condition that ensures that the solution of the perturbed problem (5.9) stays close to the solution of the original problem (5.1) as $\varepsilon \to 0$. This is the case for strict local minimizers that satisfy the second-order sufficient condition (5.7).

Theorem 5.4: Stability of the Solutions
Consider the perturbed problem (5.9) where $f(\cdot)$ and $\phi(\cdot)$ are twice continuously differentiable functions. Let $x^\star$ be a local minimizer that satisfies the second-order sufficient conditions (5.7) for the unperturbed problem (i.e., $\varepsilon = 0$). Then, for ε small enough, there exists a local minimizer $x_\varepsilon^\star$ of the perturbed problem (5.9) such that:

$$x_\varepsilon^\star = x^\star - \nabla^2 f(x^\star)^{-1} \nabla \phi(x^\star)\varepsilon + o(\varepsilon). \tag{5.10}$$

Proof. Let us apply the *implicit function theorem* to the nonlinear equation given by the first-order conditions:

$$\nabla f(x) + \varepsilon \nabla \phi(x) = 0, \tag{5.11}$$

around the solution $(x, \varepsilon) = (x^\star, 0)$. This implies that, for ε small enough, a smooth function $\varepsilon \mapsto x_\varepsilon^\star$ satisfies the first-order necessary condition for the perturbed problem (5.9). For ε small enough, the second-order sufficient condition (5.7b) is also satisfied, which ensures that the solution $x_\varepsilon^\star$ is a strict local minimizer of the perturbed problem (5.9). Furthermore, to compute the derivative $\frac{\mathrm{d}x_\varepsilon^\star}{\mathrm{d}\varepsilon}\big|_{\varepsilon=0}$, we differentiate the stationarity condition (5.11) with respect to ε, which gives:

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big(\nabla f(x_\varepsilon^\star) + \varepsilon \nabla \phi(x_\varepsilon^\star)\Big)\Big|_{\varepsilon=0} = 0, \tag{5.12a}$$

$$\implies \left(\nabla^2 f(x_\varepsilon^\star)\frac{\mathrm{d}x_\varepsilon^\star}{\mathrm{d}\varepsilon} + \nabla \phi(x_\varepsilon^\star) + \varepsilon \nabla^2 \phi(x_\varepsilon^\star)\frac{\mathrm{d}x_\varepsilon^\star}{\mathrm{d}\varepsilon}\right)\Big|_{\varepsilon=0} = 0, \tag{5.12b}$$

$$\implies \nabla^2 f(x^\star)\frac{\mathrm{d}x_\varepsilon^\star}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} + \nabla \phi(x^\star) = 0, \tag{5.12c}$$

$$\implies \frac{\mathrm{d}x_\varepsilon^\star}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0} = -\nabla^2 f(x^\star)^{-1} \nabla \phi(x^\star). \tag{5.12d}$$

Finally, we can write the Taylor expansion of the smooth function $\varepsilon \mapsto x_\varepsilon^\star$ around $\varepsilon = 0$:

$$x_\varepsilon^\star = x^\star + \underbrace{\frac{\mathrm{d}x_\varepsilon^\star}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}}_{=-\nabla^2 f(x^\star)^{-1}\nabla\phi(x^\star)} \varepsilon + o(\varepsilon), \tag{5.13}$$

which proves the desired result (5.10). □

Remark
Similar analysis can be performed for parametric optimization problems:

$$\underset{x \in D}{\text{minimize}} \quad f(x, p), \tag{5.14}$$

where $f : D \times \mathbb{R}^m \to \mathbb{R}$ depends on a parameter $p \in \mathbb{R}^m$. Hence, the solution $x^\star(p)$ depends on the parameter p. The function $p \mapsto x^\star(p)$ is called the *solution map*. *Parametric optimization* is an active research area. For example, one problem is to find a solution for some parameter value p_1 when some solution for another parameter value p_0 is already known. A popular approach is the *homotopy method*, which consists of changing slowly the parameter p from p_0 to p_1 while updating the solution $x^\star(p)$.

Chapter 6

Estimation and Fitting Problems

In this chapter, we will study optimization problems that have a special structure, namely the least-squares structure[1]:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \|\eta - \Phi(x)\|_2^2 . \tag{6.1}$$

Here, $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ is a "model" that depends on "model parameters" $x \in \mathbb{R}^n$ and tries to predict the vector $\eta \in \mathbb{R}^m$ which contains some "measurements". This formulation is widely present in *estimation* or *fitting* problems. Typically, one uses this least-squares fitting to approximate the relationship between the input samples and the output measurements. More formally, one might need to derive an approximation of a function $u \mapsto \varphi^\star(u)$ from a couple of samples $(u_1, \eta_1), \ldots, (u_m, \eta_m)$ where $\eta_i = \varphi^\star(u_i)$. The first step is to approximate $u \mapsto \varphi^\star(u)$ with a model $u \mapsto \varphi_{\text{m.}}(u; x)$, which depends on an (unknown, but) finite-dimensional parameter-vector $x \in \mathbb{R}^n$. For example, one could choose to approximate $\varphi^\star(u)$ by a polynomial of degree 3, as illustrated below, in Fig. 6.2. The parameter-vector $x \in \mathbb{R}^n$ is estimated by solving the least-squares problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{m} (\eta_i - \varphi_{\text{m.}}(u_i; x))^2. \tag{6.2}$$

Note that problem (6.2) can be formulated as (6.1) by defining the following vectors:

$$\Phi(x) = \begin{bmatrix} \varphi_{\text{m.}}(u_1; x) \\ \varphi_{\text{m.}}(u_2; x) \\ \vdots \\ \varphi_{\text{m.}}(u_m; x) \end{bmatrix}, \quad \eta = \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_m \end{bmatrix}. \tag{6.3}$$

These types of problems arise in many applications, for example:

- Estimation of physical parameters from measurements;
- Training of neural networks;
- Data reconciliation in weather forecasting;
- System identification in control engineering;

[1] Recall the definition of the Euclidean norm: $\|x\|_2 = \left(\sum_{i=1}^n x_i^2\right)^{1/2} = (x^\top x)^{1/2}$.

- Fitting of curves to data points in statistics.

In the machine learning terminology, we say that we *learn* the function $\varphi^\star(\cdot)$ by training the model $\varphi_{\text{m.}}(\cdot; x)$ on the training dataset $\{(u_1, \eta_1), \ldots, (u_m, \eta_m)\}$. A remarkable fact is that this type of optimization problem already attracted the attention of the German mathematician Carl Friedrich Gauss in *1809* for estimating the parameters of the orbit of the asteroid Ceres.
When the computation of $\Phi(x)$ is itself a very complex algorithm (e.g., involving the solution of a differential equation), its computation is sometimes called the "forward problem": it represents the task of predicting the outputs given the parameters of the model. On the other hand, solving the least-squares problem (6.1) is often called the "inverse problem": given some actual outputs, we look for the parameters that would give some predictions as close as possible to these outputs.

6.1 Linear Least Squares

The most convenient case (regarding optimization at least) is to choose a model $\Phi(x)$ that is linear in the parameters x:

Definition 6.1: Linear Least Squares (LLS) Problem
A *Linear Least Squares problem* (LLS) is a special case of the least squares problem (6.1), where the model $\Phi(x)$ is linear in x:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \|\eta - Jx\|_2^2, \tag{6.4}$$

for some matrix $J \in \mathbb{R}^{m \times n}$.

Example 6.1: Affine Regression
Consider the following fitting problem where we try to fit an affine function to a set of input-output data points $(u_i, \eta_i) \in \mathbb{R}^2$:

$$\underset{(a,b) \in \mathbb{R}^2}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{m} (\eta_i - (au_i + b))^2. \tag{6.5}$$

This is illustrated in Fig. 6.1.

Example 6.2: Polynomial fitting
Consider the following fitting problem where we try to fit a polynomial of degree 3 to a set of input-output data points $(u_i, \eta_i) \in \mathbb{R}^2$:

$$\underset{(a,b,c,d) \in \mathbb{R}^4}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{m} \left(\eta_i - \left(au_i^3 + bu_i^2 + cu_i + d\right)\right)^2. \tag{6.6}$$

This is illustrated in Fig. 6.2.

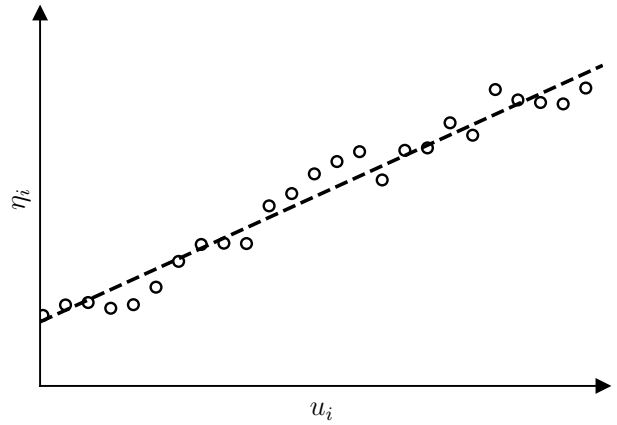


Figure 6.1: Affine regression for a set of data points (u_i, η_i).

Example 6.3: Sinusoidal fitting
Consider the following fitting problem where we try to fit a sum of sinusoidal functions to a set of input-output data points $(u_i, \eta_i) \in \mathbb{R}^2$:

$$\underset{x \in \mathbb{R}^3}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{m} (\eta_i - (x_1 \sin(u_i) + x_2 \sin(2u_i) + x_3 \sin(3u_i)))^2. \tag{6.7}$$

This is illustrated in Fig. 6.2.

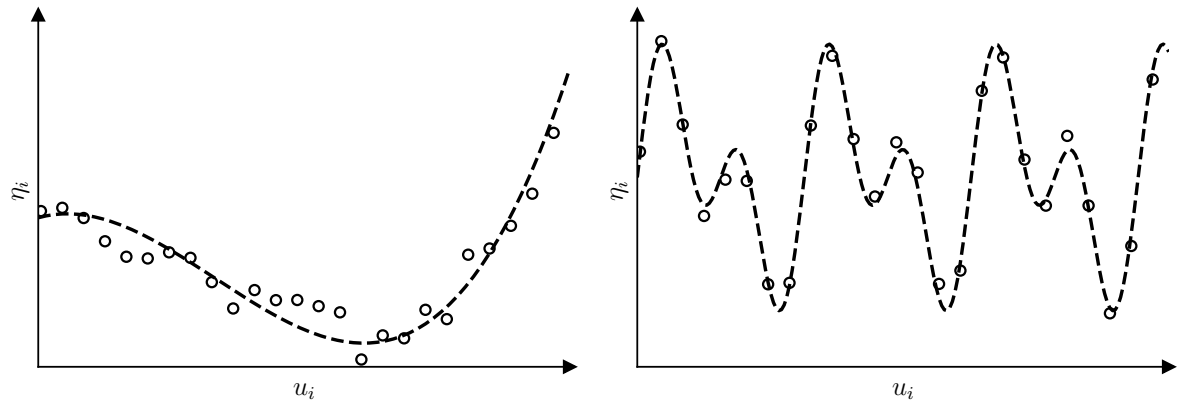


Figure 6.2: Fitting a polynomial (on the left) or a sinusoidal function (on the right) to a dataset (u_i, η_i). The circles represent the data points (u_i, η_i), and the dotted line represents the polynomial/sinusoidal model that has been *fitted* to the data.

Proposition 6.1: Solutions of the linear least squares problem
The LLS problem (6.4) is a convex QP problem. Its solutions are the solutions of the following linear equation:

$$J^\top J x^\star = J^\top \eta. \tag{6.8}$$

Proof. The function $f(x) = \frac{1}{2}\|\eta - Jx\|_2^2$ is convex because it is the composition of a convex function (the squared norm) with an affine function (the linear function Jx). Therefore, the solutions are the stationary points, i.e., the solutions of $\nabla f(x^\star) = 0$. This equation leads to the result (6.8). □

Corollary 6.1: Closed-form solution for linear least squares
If $J^\top J$ is invertible, then the unique solution of the LLS problem (6.4) is given by

$$x^\star = \underbrace{\left(J^\top J\right)^{-1} J^\top}_{=:J^\dagger} \eta. \tag{6.9}$$

Example 6.4: Average linear least squares
Consider the simplest example, where the model is an unknown constant:

$$\underset{x \in \mathbb{R}}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{m}(\eta_i - x)^2. \tag{6.10}$$

This corresponds to the case where $J = \begin{bmatrix} 1 & 1 & \vdots & 1 \end{bmatrix}^\top$. Applying the formula (6.9), we obtain the intuitive result that the solution of (6.10) is the average of the measurements η_i:

$$x^\star = \left(J^\top J\right)^{-1} J^\top \eta = \frac{1}{m}\sum_{i=1}^{m}\eta_i. \tag{6.11}$$

Example 6.5: Solution of the Affine Regression problem
Consider the affine regression problem in Example 6.1. This can be formulated as (6.4) with $x = (a, b)$ and $J = \begin{bmatrix} u_1 & 1 \\ u_2 & 1 \\ \vdots & \vdots \\ u_m & 1 \end{bmatrix} \in \mathbb{R}^{m\times 2}$. The solution is given by equation (6.9), which can be written explicitly in this case:

$$\begin{bmatrix} a^\star \\ b^\star \end{bmatrix} = \left(J^\top J\right)^{-1} J^\top \eta, \tag{6.12a}$$

$$= \left(\begin{bmatrix} u_1 & \cdots & u_m \\ 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_1 & 1 \\ u_2 & 1 \\ \vdots & \vdots \\ u_m & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} u_1 & \cdots & u_m \\ 1 & \cdots & 1 \end{bmatrix}^\top \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_m \end{bmatrix}, \tag{6.12b}$$

$$= \left(\begin{bmatrix} \sum_{i=1}^m u_i^2 & \sum_{i=1}^m u_i \\ \sum_{i=1}^m u_i & m \end{bmatrix} \right)^{-1} \begin{bmatrix} \sum_{i=1}^m u_i\eta_i \\ \sum_{i=1}^m \eta_i \end{bmatrix}, \tag{6.12c}$$

$$= \frac{1}{m\sum_{i=1}^m u_i^2 - \left(\sum_{i=1}^m u_i\right)^2} \begin{bmatrix} \sum_{i=1}^m u_i^2 & -\sum_{i=1}^m u_i \\ -\sum_{i=1}^m u_i & m \end{bmatrix} \begin{bmatrix} \sum_{i=1}^m u_i\eta_i \\ \sum_{i=1}^m \eta_i \end{bmatrix}. \tag{6.12d}$$

Remark
In equation (6.9), we have defined the matrix $J^\dagger = \left(J^\top J\right)^{-1} J^\top$. This matrix is called the *pseudo-inverse* of J. This is due to the fact that it is a generalization of the inverse, because if J is square and invertible, then $J^\dagger = J^{-1}$. This is consistent with the fact that if J is square and invertible, then the solution of (6.4) is obviously $x^\star = J^{-1}\eta$.
Note that this definition only makes sense for well-posed problems. A more general definition of the pseudo-inverse is given as follows.

6.2 Well-posed vs. Ill-posed Linear Least Squares Problems

Definition 6.2: Well-posed vs. Ill-posed problems
The problem is said to be well-posed when the solution is unique, i.e., when $J^\top J$ is invertible. This is illustrated in Fig. 6.3.
On the contrary, it is said to be *ill-posed* when $J^\top J$ is not invertible. This is illustrated in Fig. 6.4.

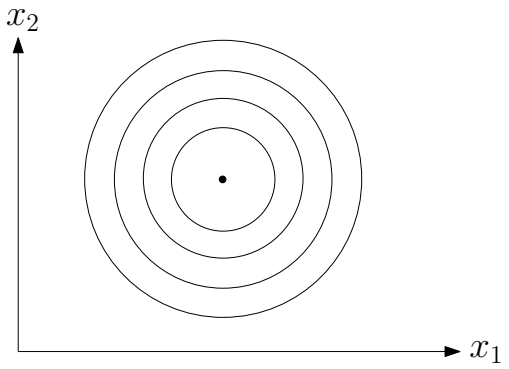


Figure 6.3: $J^\top J$ is invertible, resulting in a unique minimizer (well-posed).

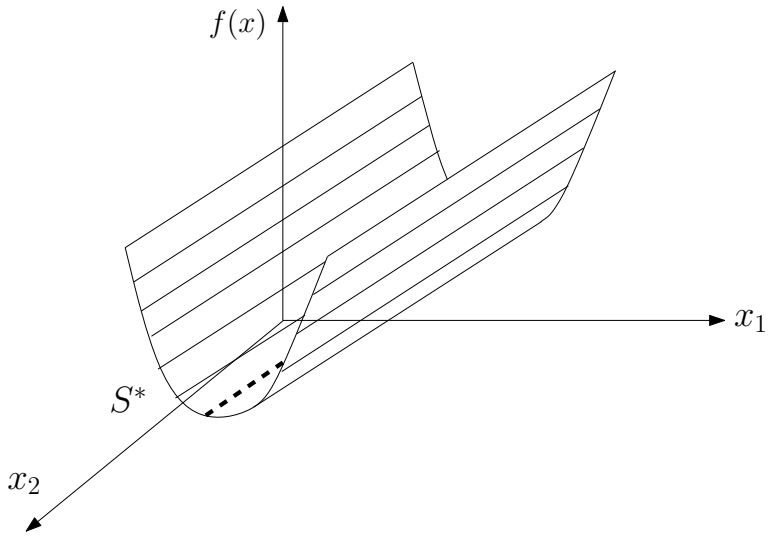


Figure 6.4: $J^\top J$ is not invertible, resulting in a non-unique minimizer (ill-posed).

Remark
Note that a problem is well-posed *if and only if* the matrix J has full column rank, i.e., $\operatorname{rank}(J) = n$. An interesting observation is that this can happen only if the number of measurements m is at least as large as the number of parameters n, i.e., $m \geq n$. However, this condition is not sufficient, because it can be that some columns of J are linearly dependent by "accident".

6.3 Regularization for Least Squares

The minimum norm solution can be approximated by a "regularized problem"

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \|\eta - Jx\|_2^2 + \frac{\varepsilon}{2} \|x\|_2^2 , \tag{6.13a}$$

with small $\epsilon > 0$, to get a unique solution

$$x_\varepsilon^\star = \left(J^\top J + \varepsilon \mathbb{I}\right)^{-1} J^\top \eta. \tag{6.14}$$

In the remainder of this part, we study how the solution of this problem behaves when $\varepsilon \to 0$.

Definition 6.3: Moore-Penrose Pseudo-Inverse
Let $J \in \mathbb{R}^{m \times n}$ be a rectangular matrix. Then, the Moore-Penrose pseudo-inverse $J^\dagger$ is defined as follows:

$$J^\dagger \coloneqq \lim_{\varepsilon \to 0} \left(J^\top J + \varepsilon \mathbb{I}_n\right)^{-1} J^\top \tag{6.15}$$

This limit always exists, which is justified by the numerical method 6.1 below.
The nice thing with the pseudo-inverse is that it allows us to compute a solution to the LLS problem (6.4) even when the problem is ill-posed. This is highlighted in the following proposition.

Proposition 6.2: Interpretation of the pseudo-inverse
Consider the LLS problem (6.4). For ill-posed problems, the pseudo-inverse can be used to find the minimum-norm solution. More precisely, not only is $x^\star \coloneqq J^\dagger \eta$ *a* solution of (6.4), but it is also *the* solution of the following problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \|x\|_2 \tag{6.16a}$$

$$\text{subject to} \quad x \in \underset{\tilde{x} \in \mathbb{R}^n}{\arg\min} \ \|\eta - J\tilde{x}\|_2^2 . \tag{6.16b}$$

Proof. Left as an exercise. □

Proposition 6.3: Pseudo-Inverse for well-posed problems
If $J^\top J$ is invertible, then the pseudo-inverse $J^\dagger$ is given by $J^\dagger = \left(J^\top J\right)^{-1} J^\top$. The pseudo-inverse $J^\dagger$ is a generalization of the inverse, because if J is square and invertible, then $J^\dagger = J^{-1}$.

Proof. This is a direct consequence of the continuity of the matrix inverse function on the set of invertible matrices. □

It is usually preferable to compute the pseudo-inverse $J^\dagger$ using the *Singular Value Decomposition* (SVD) of J, which is given by the following theorem (from numerical linear algebra).

Theorem 6.1: Singular Value Decomposition (SVD)
Any rectangular matrix $J \in \mathbb{R}^{m\times n}$ admits a Singular Value Decomposition (SVD):

$$J = U\Sigma V^\top, \tag{6.17}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices, and $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix with non-negative entries.

Remark
The SVD can be computed efficiently, for example using Gauss elimination.

Numerical Method 6.1: Computing the pseudo-inverse for ill-posed problems
The pseudo-inverse $J^\dagger$ can be computed even when $J^\top J$ is not invertible using the SVD of J:

$$J^\dagger = V\Sigma^\dagger U^\top, \tag{6.18}$$

where $\Sigma^\dagger$ is the diagonal matrix containing the inverse of the non-zero entries of Σ and zeros in place of the zero entries of Σ:

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & & & \\ & \sigma_2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_r & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{bmatrix}, \qquad \Sigma^\dagger = \begin{bmatrix} \sigma_1^{-1} & & & & & & \\ & \sigma_2^{-1} & & & & & \\ & & \ddots & & & & \\ & & & \sigma_r^{-1} & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{bmatrix}. \tag{6.19}$$

Proof.

$$\begin{aligned} J^\dagger &\coloneqq \lim_{\varepsilon\to 0} \left(J^\top J + \varepsilon \mathbb{I}_n\right)^{-1} J^\top && \text{(6.20a)}\\ &= \lim_{\varepsilon\to 0} \left(V\Sigma U^\top U\Sigma V^\top + \varepsilon \mathbb{I}_n\right)^{-1} V\Sigma U^\top && \text{(6.20b)}\\ &= \lim_{\varepsilon\to 0} V\left(\Sigma^2 + \varepsilon \mathbb{I}_n\right)^{-1}\Sigma U^\top && \text{(6.20c)}\\ &= V\Bigg(\underbrace{\lim_{\varepsilon\to 0} \left(\Sigma^2 + \varepsilon \mathbb{I}_n\right)^{-1}\Sigma}_{=\Sigma^\dagger}\Bigg)U^\top, && \text{(6.20d)} \end{aligned}$$

where the last equality is obtained by noting that the matrix $\left(\Sigma^2 + \varepsilon \mathbb{I}_n\right)^{-1}\Sigma$ is a diagonal matrix with entries $\frac{\sigma_i}{\sigma_i^2+\varepsilon}$, which converges to σ_i^{-1} if $\sigma_i > 0$ and is equal to 0 if $\sigma_i = 0$. □

Remark
The SVD is a very powerful tool for computing the pseudo-inverse, because it allows us to compute the pseudo-inverse even when $J^\top J$ is not invertible, i.e., when the problem is ill-posed. Even for well-posed problems, it is often more numerically stable to compute the pseudo-inverse using the SVD rather than using the formula in Proposition 6.3.

6.4 Statistical Interpretation of Least Squares Problems

A least squares problem (6.1) can be interpreted as finding the x that "explains as best" the noisy measurements η.
To clarify what this means, we need to introduce a probabilistic framework for the measurements η. Here, we assume that it is produced by the following process:

$$\eta_i = \varphi_{\mathrm{m.}}(u_i; x^{\mathrm{tr.}}) + \varepsilon_i, \quad i = 1, \dots, m, \tag{6.21}$$

where $x^{\mathrm{tr.}}$ is the *true* parameter (unknown to us) and ε_i are *random variables* with zero mean, i.e., $\mathbb{E}[\varepsilon_i] = 0$.
We can also formulate this in the vectorized form:

$$\eta = \Phi(x^{\mathrm{tr.}}) + \varepsilon, \tag{6.22}$$

where $\varepsilon \in \mathbb{R}^m$ is a *multivariate random variable*.
In this stochastic framework, a good tool to estimate the unknown parameter $x^{\mathrm{tr.}}$ is the *maximum-likelihood estimate* (MLE):

Definition 6.4: Maximum-Likelihood Estimate
The MLE is the value of x for which the probability of observing what we observed is the highest. More formally, the MLE is the solution of the following problem:

$$\underset{x \in \mathbb{R}^n}{\text{maximize}} \quad p(\eta \mid x) \quad = p\big(\{\eta = \Phi(x^{\mathrm{tr.}}) + \varepsilon\} \mid \{x^{\mathrm{tr.}} = x\}\big) \quad = p(\{\varepsilon = \eta - \Phi(x)\}) \tag{6.23}$$

In order to formulate the MLE problem, one needs to choose a probability distribution for the noise ε. One popular choice is the *normal distribution*, a.k.a. the *Gaussian distribution*[2].
With this specific choice, the MLE gives an interpretation of the least-squares problem (6.1) as a statistical problem, as proven in the following proposition.

Proposition 6.4: MLE and Least Squares Problem
Assume that the noise ε follows a *Gaussian distribution* with zero mean and with a covariance matrix $\sigma_\varepsilon^2 \mathbb{I}$. Then, the MLE problem (6.23) is equivalent to the least-squares problem (6.1).

[2]Recall that a Gaussian distribution with mean $\mu \in \mathbb{R}^n$ and covariance matrix $\Sigma \succ 0$ is the following probability density function:

$$p(z) = \frac{1}{\sqrt{(2\pi)^m \det(\Sigma)}} \exp\left(-\frac{1}{2}(z-\mu)^\top \Sigma^{-1}(z-\mu)\right).$$

Remark
For Gaussian noise, having a covariance matrix of the form $\sigma_\varepsilon^2 \mathbb{I}$ is equivalent to having components ε_i *independent* and *identically distributed* (i.i.d.).

Proof. Applying the definition of the Gaussian distribution to $\mu = 0$ and $\Sigma = \sigma_\varepsilon^2 \mathbb{I}$, we find:

$$p(\varepsilon) = \underbrace{\frac{1}{\sqrt{(2\pi)^m \det(\sigma_\varepsilon^2 \mathbb{I})}}}_{=:C} \exp\left(-\frac{\|\varepsilon\|_2^2}{2\sigma_\varepsilon^2}\right). \tag{6.24}$$

Plugging this into the MLE problem (6.23), we obtain:

$$\underset{x \in \mathbb{R}^n}{\text{maximize}} \quad C \exp\left(-\frac{1}{2\sigma_\varepsilon^2} \|\eta - \Phi(x)\|_2^2\right) \tag{6.25}$$

Furthermore, since $p \mapsto -\log(p)$ is strictly monotonically decreasing, maximizing $p(\eta \mid x)$ is equivalent to minimizing $-\log(p(\eta \mid x))$. Hence, (6.25) is equivalent to:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad -\log(C) + \frac{1}{2\sigma_\varepsilon^2} \|\eta - \Phi(x)\|_2^2 \tag{6.26}$$

Finally, the multiplicative and additive constants do not affect the solution of this problem, so we can deduce that this problem is equivalent to the least-squares problem (6.1). □

Remark
If, instead, the noise covariance has a diagonal covariance with different values on the diagonal, i.e.:

$$\mathrm{Cov}[\varepsilon] = \mathbb{E}\left[\varepsilon\varepsilon^\top\right] = \begin{bmatrix} \sigma_{\varepsilon_1}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{\varepsilon_2}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{\varepsilon_m}^2 \end{bmatrix} \tag{6.27}$$

Then the MLE is equivalent to the weighted least-squares problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{m} \frac{1}{\sigma_{\varepsilon_i}^2} (\eta_i - \Phi_i(x))^2. \tag{6.28}$$

Remark
The L_1-estimation problem can also be interpreted as a maximum-likelihood estimate, but where the noise follows a *Laplace distribution* instead of a *Gaussian distribution.*

Another important statistical tool is the *Maximum A Posteriori* (MAP) estimate.

Definition 6.5: Maximum A Posteriori (MAP)
In the Maximum A Posteriori problem (MAP), we look for the parameter x that maximizes the *posterior probability* $p(x \mid \eta)$, which is the probability of x given the measurements η. More formally, the MAP problem is defined as:

$$\underset{x \in \mathbb{R}^n}{\text{maximize}} \quad p(x \mid \eta) = p\big(\{x^{\text{tr.}} = x\} \mid \{\eta = \Phi(x^{\text{tr.}}) + \varepsilon\}\big) \tag{6.29}$$

Using Bayes' theorem, this can be rewritten as:

$$\underset{x \in \mathbb{R}^n}{\text{maximize}} \quad p(x \mid \eta) = \frac{p(\eta \mid x)p(x)}{p(\eta)} \propto p(\eta \mid x)p(x) \tag{6.30}$$

where $p(x)$ is called the *prior distribution* on the parameter x because it models the knowledge on this parameter *before* observing the measurements η. This is why the MAP problem is also sometimes called the *Bayesian inference problem.*
In the case of the assumption from Proposition 6.4, (i.e., that ε_i are zero-mean i.i.d. Gaussians), and assuming a Gaussian prior on x with mean $\mathbb{E}[x] = \bar{x}$ and covariance matrix $\text{Cov}[x] = \sigma_x^2 \mathbb{I}$, the MAP problem (6.29) is equivalent to the *regularized least-squares* problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2\sigma_\varepsilon^2} \|\eta - \Phi(x)\|_2^2 + \frac{1}{2\sigma_x^2} \|x - \bar{x}\|_2^2 \tag{6.31}$$

6.5 L_1 Estimation (Least Absolute Deviations)

Instead of using the metric $\|\cdot\|_2^2$ to evaluate the distance between the predicted values $\Phi(x)$ and the measurements η, one could also use the L_1-norm, i.e., $\|\eta - \Phi(x)\|_1 = \sum_{i=1}^m |\eta_i - \Phi_i(x)|$. This leads to the so-called L_1-*estimation* problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \|\eta - \Phi(x)\|_1 \coloneqq \sum_{i=1}^m |\eta_i - \varphi(u_i; x)| \,. \tag{6.32}$$

The motivation behind this is that the L_1-norm is less sensitive to outliers than the L_2-norm. This is why L_1-estimation is sometimes called "robust" parameter estimation.
Note that problem (6.32) is non-smooth because the absolute value function $|x|$ is not differentiable at $x = 0$. However, there is a way to reformulate this problem into a smooth optimization problem by introducing inequality constraints and auxiliary variables:

$$\begin{aligned} \underset{x \in \mathbb{R}^n,\, s \in \mathbb{R}^m}{\text{minimize}} \quad & \sum_{i=1}^m s_i \\ \text{subject to} \quad & \eta_i - \varphi(u_i; x) \geq -s_i, \quad i = 1, \ldots, m, \\ & \eta_i - \varphi(u_i; x) \leq s_i, \quad i = 1, \ldots, m. \end{aligned} \tag{6.33}$$

An interesting observation is that for a constant model, i.e., $\varphi_{\mathrm{m.}}(u_i, x) = x$, the optimal L_2-fit is the mean of the measurements while the optimal L_1-fit is the median of the measurements:

$$\arg\min_{x\in\mathbb{R}} \sum_{i=1}^{m} (\eta_i - x)^2 \quad = \quad \text{mean of } \{\eta_1, \ldots, \eta_m\}, \tag{6.34a}$$

$$\arg\min_{x\in\mathbb{R}} \sum_{i=1}^{m} |\eta_i - x| \quad = \quad \text{median of } \{\eta_1, \ldots, \eta_m\}. \tag{6.34b}$$

6.6 The Gauss-Newton Method

It is astonishing that when the German mathematician Carl Friedrich Gauss was estimating some quantities in astrophysics based on measurements in *1809*, not only did he *formulate* a nonlinear least squares problem but also *invented an algorithm* to solve it: the *Gauss-Newton method*. What is even more astonishing is that this algorithm is still a state-of-the-art method for solving nonlinear least squares problems today. The main idea of this method is to iteratively linearize the residuals and solve the resulting linear least squares problem. It happens that the resulting method ends up being closely related to another method developed *even earlier* for solving nonlinear equations, named after the English physicist and mathematician Isaac Newton. Therefore, this iterative method for solving nonlinear least squares problems is called the *Gauss-Newton method*.

Note that from a theoretical point of view, there is no clear answer because general nonlinear least squares problems of the form (6.1) can be nonconvex, with possibly many local minima. However, in practice, the structure of these problems makes them easier to solve. This is the case because the linearizations of the inner nonlinear function $\Phi(x)$ often yield good approximations for the original problem, which are easy to solve.

Before introducing the Gauss-Newton method formally, we reformulate the nonlinear least squares problem (6.1) into a more compact form:

$$\begin{array}{ll} \underset{x \,\in\, \mathbb{R}^n}{\text{minimize}} & \dfrac{1}{2} \left\| F(x) \right\|_2^2, \end{array} \tag{6.35}$$

where we defined the residual function $F(x) \coloneqq \eta - \Phi(x)$.

Numerical Method 6.2: The Gauss-Newton Method (for unconstrained problems)
The Gauss-Newton method is an iterative method for solving nonlinear least square problems where, at each iteration, the following computations are performed:

i. Linearize the residual function $F(\cdot)$ using a first-order Taylor expansion at the current iterate;

ii. Solve the resulting linear least squares problem analytically to compute the next iterate.

More precisely, when facing a problem of the form (6.35), we construct the iterates $x_0, \dots, x_k$ based on the following rule:

$$\begin{aligned} x_{k+1} &= \arg\min_{x \in \mathbb{R}^n} \frac{1}{2} \left\| F(x_k) + \nabla F(x_k)^\top (x - x_k) \right\|_2^2 \\ &= x_k - \left(\nabla F(x_k)^\top \right)^\dagger F(x_k), \\ &= x_k - \left(\nabla F(x_k) \nabla F(x_k)^\top \right)^{-1} \underbrace{\nabla F(x_k) F(x_k)}_{\nabla f(x_k)}. \end{aligned} \tag{6.36}$$

As you might have noticed, the last equality is only valid when the matrix $\nabla F(x_k)\nabla F(x_k)^\top$ is invertible. Furthermore, if this matrix is *almost* singular, computing its inverse is "numerically dangerous". Another interesting observation is that in the Gauss-Newton method, the linearization of $F(x_k)$ is only accurate for $x \approx x_k$. Therefore, it would be interesting to avoid taking steps where x_{k+1} deviates *too much* from the current iterate x_k.

6.7 Levenberg-Marquardt Method

The last two observations motivated the two American statisticians Kenneth Levenberg and Donald Marquardt to develop the so-called *Levenberg-Marquardt* method (K. Levenberg published the method for the first time in 1944, and D. Marquardt analyzed it more in depth in 1963). In this method, we add a penalization term to the approximated objective function, which penalizes solutions that are too far from the current iterate x_k. This penalization term takes the form pen. $= \frac{\alpha}{2} \|x - x_k\|_2^2$ with some parameter $\alpha > 0$.

Numerical Method 6.3: The Levenberg-Marquardt method (for unconstrained problems)
The Levenberg-Marquardt method is an iterative method for solving nonlinear least squares problems where, at each iteration, the following computations are performed:

i. Linearize the residual function $F(\cdot)$ using a first-order Taylor expansion at the current iterate;

ii. Add a penalization term to the objective function;

iii. Solve the resulting linear least squares problem analytically to compute the next iterate.

More precisely, when facing a problem of the form (6.35), we construct the iterates $x_0, \dots, x_k$ based on the following rule:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^n} \frac{1}{2} \left\| F(x_k) + \nabla F(x_k)^\top (x - x_k) \right\|_2^2 + \frac{\alpha}{2} \|x - x_k\|_2^2 \tag{6.37a}$$

$$= x_k - \left(\nabla F(x_k) \nabla F(x_k)^\top + \alpha \mathbb{I} \right)^{-1} \underbrace{\nabla F(x_k) F(x_k)}_{\nabla f(x_k)}. \tag{6.37b}$$

It is interesting to see that both the Gauss-Newton and the Levenberg-Marquardt methods take the more general form:

$$x_{k+1} = x_k - {B_k}^{-1} \underbrace{\nabla F(x_k) F(x_k)}_{\nabla f(x_k)}, \tag{6.38}$$

for some matrix $B_k \succ 0$. In the next chapter, we will study exactly this more general family of methods, which is called *Newton type optimization methods*. There, we will present the important results regarding the convergence of these methods in general, and in particular, the convergence properties of the Gauss-Newton algorithm.

Chapter 7

Newton Type Methods

In this chapter, we will study a powerful method: *Newton's method*. It is an *iterative* algorithm to compute a solution of a nonlinear root finding problem, which can be applied to optimization problems by formulating the associated first-order optimality conditions as a root-finding problem.

Definition 7.1: Iterative Algorithm
We call an "iterative algorithm" any procedure that starts from an *initial guess* $w_0 \in \mathbb{R}^n$, and improves it incrementally through multiple *iterations*. The current estimate of the solution, at iteration k, is denoted by $w_k \in \mathbb{R}^n$, and is called an "iterate" of the algorithm. The algorithm usually stops when some *stopping criterion* is met, i.e., a condition that quantifies how good our solution is.

Initially Newton's method is a method for solving root-finding problems, i.e., finding $w \in \mathbb{R}^n$ such that $F(w) = 0$, where $F : \mathbb{R}^n \to \mathbb{R}^n$ is a function representing some equations that one wants to solve. Later, we will see how to apply this method to optimization problems instead.
The Newton method is named after the English physicist and mathematician *Isaac Newton*. However, he only used this method in the very specific case of root-finding for a polynomial one-dimensional function, at the end of the 17th century. Other scientists over the years successively improved the method. Notably, the English mathematician Joseph Raphson reformulated the method in a more compact way by making use of the concept of *derivatives*. In 1740, the English mathematician *Thomas Simpson* presented this method for solving general one-dimensional nonlinear root-finding problems. He even generalized the method to two-dimensional problems. Today, we have generalized Newton's method for n-dimensional problems, using linear algebra.
Let us discuss the problem of finding a solution to some nonlinear equations, which we call a *root-finding problem*. This might seem a bit different from the main goal of this course, which is numerical optimization. But root-finding algorithms can also be applied to first-order optimality conditions, yielding an optimization algorithm, as we will see below, in Section 7.1.1.

Definition 7.2: Root-finding Problems
We call a *root-finding problem* a problem of the following form:

$$\text{find } w \in \mathbb{R}^n \quad \text{such that } F(w) = 0, \tag{7.1}$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear function that has the *same number* of inputs and outputs.

Note that we only consider root-finding problems where we have as many variables as equations. This is due to the following observations:

i. if there are more equations than variables, *in most cases* there is no solution (apart from special cases, e.g., the equation $0 = 0$). In this case, we say that the problem is *overdetermined*.

ii. if there are more variables than equations, *in most cases* there are many solutions (apart from special cases, e.g., the equation $w_1^2 + w_2^2 = 0$). In this case, we say that the problem is *underdetermined*.

7.1 (Exact) Newton's Method

Newton's Method for Root-Finding Problems

Now we will present the (exact) Newton method for root-finding problems, a method where we iteratively linearize the function $F(\cdot)$ and solve the resulting linear system to compute the next iterate.

Numerical Method 7.1: (Exact) Newton's Method for Root-Finding Problems
The Newton method for solving root-finding problems of the form (7.1) is an iterative method where, at each iteration, we solve a linearized version of (7.1):

$$F(w_k) + \frac{\partial F}{\partial w}(w_k)(w - w_k) = 0. \tag{7.2}$$

More precisely, we construct the iterates based on the fact that w_{k+1} solves (7.2), which yields the explicit formula:

$$w_{k+1} = w_k - \left(\frac{\partial F}{\partial w}(w_k)\right)^{-1} F(w_k). \tag{7.3}$$

Example 7.1: Root-finding example, Exact Newton method
Regard the example $F(w) = w^4 - 1$. Here the method iterates as follows:

$$w_{k+1} = w_k - \frac{1}{4w^3}(w^4 - 1). \tag{7.4}$$

This is illustrated in Figure 7.1, where we see that the iterates quickly converge to the solution $w^\star = 1$.

Remark
It is interesting to see that the formula (7.3) (Newton) for nonlinear root-finding is very similar to the formula (6.36) (Gauss-Newton) that we had for solving nonlinear least-square problems. Indeed, the only difference is that in the Gauss-Newton method, it is a pseudo-inverse instead of the actual inverse. This is due to the fact that if there is a solution to the root-finding problem $F(x) = 0$, then this is also a solution to the least-square problem $\min_x \|F(x)\|^2$.

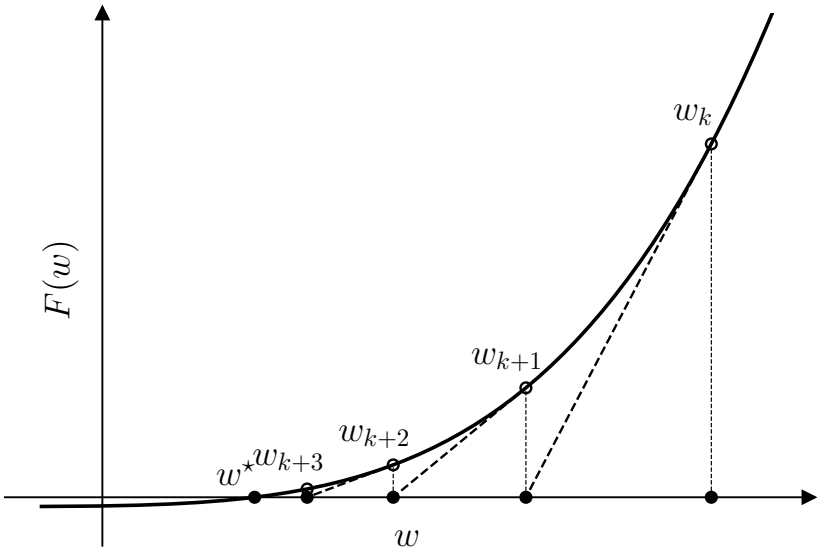


Figure 7.1: An illustration of the Newton method for the root-finding problem $w^4 - 1 = 0$

7.1.1 (Exact) Newton's Method for Unconstrained Optimization

In this part, we return to the case of optimizing a *continuously twice differentiable* function $f : \mathbb{R}^n \to \mathbb{R}$, without any constraints:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x). \tag{7.5}$$

We already saw in Chapter 5 that a necessary condition for $x^\star$ to be a solution is the following equation:

$$\nabla f(x^\star) = 0. \tag{7.6}$$

The idea of Newton's method for optimization problems is to use our root-finding method 7.1 for the equation (7.6). Replacing F with ∇f in (7.3), we obtain the following equation:

$$x_{k+1} = x_k - \left(\frac{\partial \nabla f}{\partial x}(x_k)\right)^{-1} \nabla f(x_k) = x_k \underbrace{- \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)}_{=:p_k}. \tag{7.7}$$

Note that $\nabla^2 f(x_k)$ is called *the Hessian of f at x_k*, and it is the Jacobian of the gradient ∇f. Changing slightly the notations here, we prefer, for optimization algorithms, to define the step p_k explicitly, and then writing the update on x_k in a separate equation. This will be further motivated when we will see *backtracking line-search* algorithms in Chapter 9, a variant of this algorithm where p_k is only used as a *direction*, but not necessarily as a full step.
Let us write the method more formally.

Numerical Method 7.2: The Newton Method for Optimization
When facing an unconstrained optimization problem of the form (7.5), the Newton method can be applied to find a solution. This method is an iterative method where the update rule takes the form:

$$x_{k+1} = x_k \underbrace{-\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)}_{:=p_k}. \tag{7.8}$$

An interesting observation is that whenever $\nabla^2 f(x_k)$ is positive definite, i.e., $\nabla^2 f(x_k) \succ 0$, then the step p_k is a *descent direction* for f at x_k. This means that if a sufficiently small step is taken in the direction of p_k, then the value of f will necessarily decrease. Here is a more formal and specific definition of a descent direction.

Definition 7.3: Descent Direction
A vector $p \in \mathbb{R}^n$ is called a *descent direction* for the point x when:

$$\nabla f(x)^\top p < 0 \tag{7.9}$$

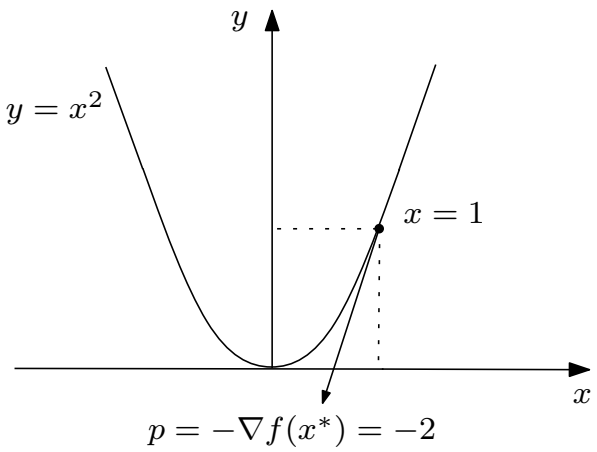


Figure 7.2: An illustration of a descent direction for $f(x) = x^2$

A second interpretation of Newton's method for optimization is that at each step, we minimize a *quadratic approximation* of the objective function f. This quadratic approximation is obtained by a second-order Taylor expansion of f at the point x_k. More precisely, define the following quadratic model m_k of objective f:

$$m_k(x_k + p) = f(x_k) + \nabla f(x_k)^\top p + \frac{1}{2} p^\top \nabla^2 f(x_k) p \quad \approx f(x_k + p) \tag{7.10}$$

Then, the direction p_k defined in equation (7.8) is *also* the minimizer of this quadratic model m_k:

$$p_k = \arg\min_p m_k(x_k + p). \tag{7.11}$$

This is illustrated in Figure 7.3. A very important remark is that this interpretation only makes

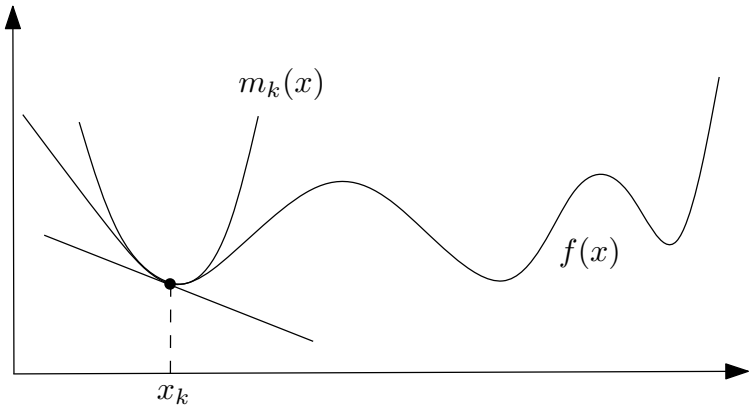


Figure 7.3: The second interpretation of Newton's method.

sense when $\nabla^2 f(x_k) \succ 0$. In general, if this positiveness condition is not fulfilled, the Newton step (7.8) should not be used.
We highlight that it is important to keep in mind these two interpretations of Newton's method. Indeed, sometimes it is more convenient to think about the Newton step as a root-finding step on the first-order optimality conditions, and sometimes it is better to view it as optimizing an approximation of the objective function.
Later, when we will study algorithms for constrained optimization, we will see that these two interpretations lead to two different algorithms: on one hand the primal-dual interior-point method, where *the first-order optimality conditions are approximated* and on the other hand, Sequential Quadratic Programming (SQP) where *the optimization problem is approximated*. In the next chapter, (Chapter 8), we will prove the local convergence of Newton's method, i.e., the convergence of the iterates for a solution if it is initialized sufficiently close to it. We will also provide details on the *speed of convergence* of the iterates. In order to do so, in the section below we classify some types of convergence speed.

7.2 Convergence Rates

To compare the convergence speed of different classes of methods, we introduce the notion of *convergence rate* for a numerical method.

Definition 7.4: Convergence Rates
Let $\{w_k\}_{k\in\mathbb{N}}$ be a sequence generated by some algorithm such that $w_k \underset{k\to+\infty}{\longrightarrow} w^\star$ for some $w^\star \in \mathbb{R}^n$.
Then, we say that the sequence $\{w_k\}_{k\in\mathbb{N}}$:

- converges Q-linearly to $w^\star$ if:

$$\|w_{k+1} - w^\star\| \leq C \|w_k - w^\star\| \text{ for } k \text{ sufficiently large,} \tag{7.12}$$

  where $C < 1$ is a constant (often called the *contraction factor*);

- converges Q-superlinearly to $w^\star$ if:

$$\|w_{k+1} - w^\star\| \leq C_k \|w_k - w^\star\| \text{ for } k \text{ sufficiently large,} \tag{7.13}$$

  where C_k is a sequence of numbers such that $C_k \underset{k\to+\infty}{\longrightarrow} 0$;

- converges Q-quadratically to $w^\star$ if:

$$\|w_{k+1} - w^\star\| \leq C \|w_k - w^\star\|^2 \text{ for } k \text{ sufficiently large,} \tag{7.14}$$

  where $C < \infty$ is a constant.

Note that in the definitions above, we use the phrase "for k sufficiently large." For a more rigorous definition, one could state that there exists a $k_0 \in \mathbb{N}$ such that the inequality holds for all $k \geq k_0$.

Remark
.

i. When w_k converges Q-linearly, the error decreases *exponentially*: $\|w_k - w^\star\| = \mathcal{O}\big(e^{-\lambda k}\big)$;

ii. When w_k converges Q-superlinearly, then: $\|w_k - w^\star\| = \mathcal{O}\big(e^{-\lambda_k k}\big)$ with $\lambda_k \underset{k\to+\infty}{\longrightarrow} +\infty$;

iii. If the convergence is Q-quadratic, then $\|w_k - w^\star\| = \mathcal{O}\Big(e^{-\lambda 2^k}\Big)$.

Remark
Let us consider the case where one desires to estimate $w^\star$ up to d digits of accuracy (e.g., $d = 12$), i.e., $\|w_k - w^\star\| \leq 10^{-d}$. Then:

i. If w_k converges Q-linearly, we will need $k = \mathcal{O}(d)$ iterations.

ii. If w_k converges Q-superlinearly, we will need $k = o(d)$ iterations.

iii. If the convergence is Q-quadratic, then we will need $k = \mathcal{O}(\log(d))$ iterations.

Example 7.2: Convergence Rates
Consider examples with $w_k \in \mathbb{R}$, $w_k \to 0$.

- $w_k = \frac{1}{2^k}$ converges Q-linearly: $\frac{w_{k+1}}{w_k} = \frac{1}{2}$.
- $w_k = 0.99^k$ also converges Q-linearly: $\frac{w_{k+1}}{w_k} = 0.99$. This example converges very slowly to $w^\star$. In practice, we desire C in equation (7.12) to be smaller than, say, $\frac{1}{2}$.
- $w_k = \frac{1}{k!}$ converges Q-superlinearly, as $\frac{w_{k+1}}{w_k} = \frac{1}{k+1}$.
- $w_k = \frac{1}{2^{2^k}}$ converges Q-quadratically, because $\frac{w_{k+1}}{(w_k)^2} = \frac{(2^{2^k})^2}{2^{2^{k+1}}} = 1 < \infty$. For $k = 6$, $w_k = \frac{1}{2^{64}} \approx 0$, so in practice, convergence up to machine precision is reached after roughly 6 iterations.

Definition 7.5: R-Convergence
If the norm sequence $\|w_k - w^\star\|$ is upper bounded by some sequence $y_k \to 0$, $y_k \in \mathbb{R}$, i.e., $\|w_k - w^\star\| \leq y_k$, and if y_k converges with a given Q-rate, i.e., Q-linearly, Q-superlinearly, or Q-quadratically, then w_k is said to converge "R-linearly, R-superlinearly, or R-quadratically" to $w^\star$. Here, R indicates "root," because, e.g., R-linear convergence can also be defined via the root criterion $\sqrt[k]{\|w_k - w^\star\|} < C < 1$ for k sufficiently large.

Example 7.3: R-Convergence
.

$$w_k = \begin{cases} \frac{1}{2^k} & \text{if } k \text{ is even} \\ 0 & \text{otherwise} \end{cases} \tag{7.15}$$

This is fast R-linear convergence, but it is not monotonically decreasing like Q-linear convergence.

The three different Q-convergence and three different R-convergence rates have the following relationships[1]:

$$\begin{array}{ccccc} Q\text{-quadratically} & \Longrightarrow & Q\text{-superlinearly} & \Longrightarrow & Q\text{-linearly} \\ \Downarrow & & \Downarrow & & \Downarrow \\ R\text{-quadratically} & \Longrightarrow & R\text{-superlinearly} & \Longrightarrow & R\text{-linearly} \end{array}$$

7.3 Newton Type Methods

In this section, we discuss a generalization of Newton's method: *the Newton type methods*. These include many methods that take the same form as Newton's method, but where the Jacobian $\frac{\partial F}{\partial w}(w_k)$ (for root-finding) or the Hessian $\nabla^2 f(x_k)$ (for optimization) is replaced by some other matrix M_k (for root-finding) or B_k (for optimization).

[1] In the diagram, $X \implies Y$ should be read as "If a sequence converges with rate X, this implies that the sequence also converges with rate Y."

7.3.1 Newton Type Methods for Root-Finding Problems

For the root-finding problem, a Newton type method is a method similar to Newton's method, but where the Jacobian $\frac{\partial F}{\partial w}(w_k)$ is replaced by some other invertible matrix M_k, yielding the following general class of methods.

Numerical Method 7.3: Newton Type Methods for Root-Finding Problems
A Newton type method is a method of the following form:

$$w_{k+1} = w_k \underbrace{-M_k^{-1} F(w_k)}_{:=p_k}, \tag{7.16}$$

where M_k is some invertible matrix.

Example 7.4: Newton Type Method on a Root-Finding Problem
Consider again Example 7.1 where we applied the exact Newton method to solve the root-finding problem $w^4 - 1 = 0$. Let us now use a Newton type method, with Jacobian approximation $M_k = 100$, which yields the following update rule:

$$w_{k+1} = w_k - 0.01(w^4 - 1). \tag{7.17}$$

This is illustrated in Figure 7.4. In general, such an approximation yields slower convergence (or no convergence at all), as we will see in the next chapter.

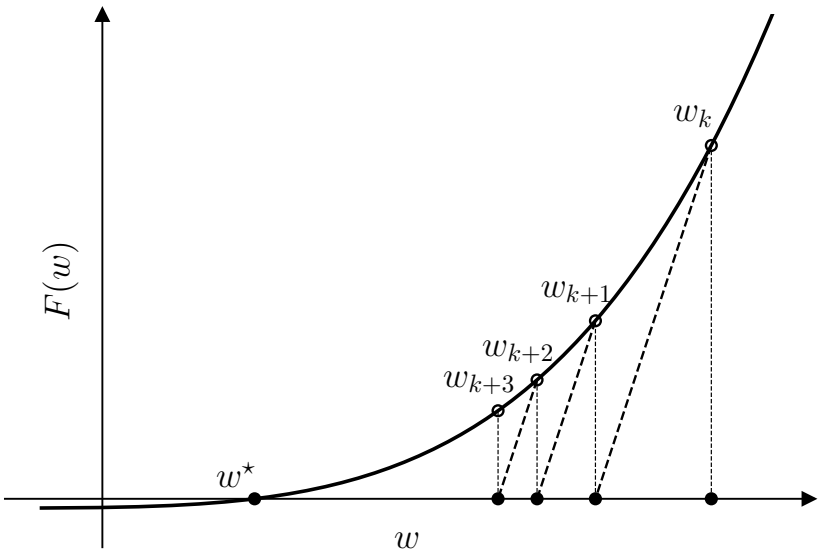


Figure 7.4: A Newton type method for the root-finding problem $w^4 - 1 = 0$, where the Jacobian is approximated by a constant value $M_k = 100$.

7.3.2 Newton Type Methods for Unconstrained Optimization

Applying the same principle, but for the special case of unconstrained optimization, where the equation to solve is $\nabla f(x) = 0$, we obtain the following method.

Numerical Method 7.4: Newton Type Methods for (unconstrained) Optimization
A Newton type method is a method of the following form:

$$x_{k+1} = x_k \underbrace{-B_k^{-1}\nabla f(x_k)}_{:=p_k}, \tag{7.18}$$

where B_k is some invertible matrix.

Note that the Newton method defined in (7.8) is a special case of Newton type methods, where $B_k = \nabla^2 f(x_k)$. This is why the matrix B_k is often called the *Hessian approximation* at the point x_k, and we try to construct a method such that this approximation holds to *some extent*: $B_k \approx \nabla^2 f(x_k)$.
This fits well with the "second interpretation" of Newton's method, where we minimize a quadratic model of the objective function f. In this case, the quadratic model m_k is defined as follows:

$$m_k(x_k + p) = f(x_k) + \nabla f(x_k)^\top p + \frac{1}{2} p^\top B_k p. \tag{7.19}$$

Now, the approximation $m_k(x_k + p) \approx f(x_k + p)$ does not come from a second-order Taylor expansion anymore, but can come from different considerations, e.g., a quadratic upper bound of f. It is important, however, that the gradient and the value of the model m_k match with the actual function f at the point $x = x_k$ (or $p = 0$).
We conclude this part with an important lemma.

Lemma 7.1: Descent direction
If $B_k \succ 0$, and $\nabla f(x_k) \neq 0$, then $p_k = -B_k^{-1}\nabla f(x_k)$ is a descent direction.

Proof.

$$\nabla f(x_k)^\top p_k = -\nabla f(x_k)^\top \underbrace{B_k^{-1}}_{\succ 0} \nabla f(x_k) < 0 \tag{7.20}$$

□

Now let us discuss some examples of Newton type methods for unconstrained optimization.

a) Exact Newton's Method

As we saw, this corresponds to the case $B_k = \nabla^2 f(x_k)$.

b) Gauss-Newton / Levenberg-Marquardt

For nonlinear least squares objectives, i.e., $f(x) = \frac{1}{2}\|F(x)\|_2^2$, the Gauss-Newton method —defined by equation (6.36) from Chapter 6— is a Newton type method, where the Hessian approximation

B_k^{GN} is defined as follows:

$$B_k^{\mathrm{GN}} = \nabla F(x_k)(\nabla F(x_k))^\top, \tag{7.21}$$

This method has the advantage to always satisfy the condition $B_k^{\mathrm{GN}} \succcurlyeq 0$, i.e., the Hessian approximation is always positive semi-definite, even when it is not necessarily the case for the actual Hessian $\nabla^2 f(x_k)$. An interesting remark is that for such objectives, the Hessian is given by the following formula:

$$\nabla^2 f(x) = \nabla^2 \left(\frac{1}{2} \sum_{i=1}^m F_i(x)^2 \right) = \underbrace{\sum_{i=1}^m \nabla F_i(x) \nabla F_i(x)^\top}_{=B_k^{\mathrm{GN}}} + \sum_{i=1}^m F_i(x) \nabla^2 F_i(x) \tag{7.22}$$

This means that the Gauss-Newton method matches with the Newton method whenever each residual $F_i(x)$ has either zero curvature (e.g., linear) or zero value.
Similarly, the Levenberg-Marquardt method is also a Newton type method, where the Hessian approximation is defined as follows:

$$B_k^{\mathrm{LM}} = \nabla F(x_k)(\nabla F(x_k))^\top + \alpha \mathbb{I}, \tag{7.23}$$

where $\alpha > 0$ is a parameter that can be tuned.

c) Steepest Descent Method / Gradient Descent Method

The famous gradient descent method $x_{k+1} = x_k - \alpha \nabla f(x_k)$ can also be viewed as a Newton type method. Indeed, it corresponds to the case of a constant Hessian approximation $B_k = \alpha^{-1}\mathbb{I}$ with some parameter $\alpha > 0$.
This method is very popular, especially in machine learning, where the dimension n can be very large. In this case, each iteration stays computationally cheap, as it only requires the computation of the gradient $\nabla f(x_k)$, and there is no need for computing the inverse of a matrix of size $n \times n$. On the other hand, this method is usually slower than the other presented methods, and therefore, more iterations might be needed to find a solution that is good enough.

d) Quasi-Newton Methods

Quasi-Newton methods are Newton type methods where $B_k \underset{k \to +\infty}{\longrightarrow} \nabla^2 f(x^\star)$. Here, the idea is to use the values of the gradients $\nabla f(x_0), \nabla f(x_1), \ldots, \nabla f(x_k)$ only to build the Hessian approximation B_k instead of computing it explicitly. This way, the computational burden of computing the Hessian $\nabla^2 f(x_k)$ is avoided. The approximation is usually based on the following Taylor expansion of the gradient ∇f:

$$\nabla f(x_k) \approx \nabla f(x_{k-1}) + \nabla^2 f(x_k)(x_k - x_{k-1}). \tag{7.24}$$

Imposing this equation for B_k instead of $\nabla^2 f(x_k)$, we obtain the following equation:

$$B_k \underbrace{(x_k - x_{k-1})}_{=:s_k} = \underbrace{\nabla f(x_k) - \nabla f(x_{k-1})}_{=:y_k}. \tag{7.25}$$

which is the so-called "secant condition". One of the most popular methods to construct a Hessian approximation B_k that satisfies the secant condition (7.25) is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method below.

Numerical Method 7.5: The BFGS Method
The BFGS method is a Newton type method where the Hessian approximation B_k is updated as follows:

$$B_k = B_{k-1} - \frac{B_{k-1}s_k s_k^\top B_{k-1}}{s_k^\top B_{k-1} s_k} + \frac{y_k y_k^\top}{s_k^\top y_k}, \tag{7.26}$$

where we defined, as above, $s_k \coloneqq x_k - x_{k-1}$ and $y_k \coloneqq \nabla f(x_k) - \nabla f(x_{k-1})$.
The rest of the algorithm is the same as for all Newton type methods, i.e.:

$$x_{k+1} = x_k \underbrace{- B_k^{-1}\nabla f(x_k)}_{=p_k}. \tag{7.27}$$

This update rule ensures that B_k satisfies the secant condition (7.25), i.e., $B_k s_k = y_k$. This can be proven with direct computations:

$$B_k s_k = \left(B_{k-1} - \frac{B_{k-1}s_k s_k^\top B_{k-1}}{s_k^\top B_{k-1} s_k} + \frac{y_k y_k^\top}{s_k^\top y_k}\right)s_k, \tag{7.28a}$$

$$= B_{k-1}s_k - \frac{B_{k-1}s_k \cancel{s_k^\top B_{k-1}s_k}}{\cancel{s_k^\top B_{k-1}s_k}} + \frac{y_k \cancel{y_k^\top s_k}}{\cancel{s_k^\top y_k}}, \tag{7.28b}$$

$$= B_{k-1}s_k - B_{k-1}s_k + y_k, \tag{7.28c}$$

$$= y_k. \tag{7.28d}$$

As we saw earlier, it is desired that $B_k \succ 0$. The following property gives us a necessary and sufficient condition for this to be guaranteed.

Proposition 7.1: Positiveness of B_k in the BFGS method
Assume that $B_{k-1} \succ 0$ and that $s_k^\top y_k > 0$. Then B_k is also positive semi-definite: $B_k \succ 0$.

Note that this property is not very restrictive, because this condition holds for any strictly convex problem. Furthermore, any update rule that satisfies the secant condition (7.25) can preserve positiveness only when $s_k^\top y_k > 0$ because otherwise: $s_k^\top B_k s_k = s_k^\top y_k < 0$ which would contradict the positiveness of B_k.

Proof. Let $p \in \mathbb{R}^n$ be a direction such that $p \neq 0$. Let us prove that $p^\top B_k p > 0$ using the definition of B_k:

$$p^\top B_k p = p^\top \left(B_{k-1} - \frac{B_{k-1}s_k s_k^\top B_{k-1}}{s_k^\top B_{k-1} s_k} + \frac{y_k y_k^\top}{s_k^\top y_k}\right)p, \tag{7.29a}$$

$$= p^\top B_{k-1}p - \frac{\left(s_k^\top B_{k-1}p\right)^2}{s_k^\top B_{k-1}s_k} + \frac{\left(y_k^\top p\right)^2}{s_k^\top y_k}, \tag{7.29b}$$

$$\geq p^\top B_{k-1}p - \frac{\left(s_k^\top B_{k-1}p\right)^2}{s_k^\top B_{k-1}s_k} > 0, \tag{7.29c}$$

where the last inequality comes from a Cauchy-Schwarz[2]inequality on the scalar product defined by the matrix B_{k-1}, i.e., $s \cdot p \coloneqq s^\top B_{k-1} p$. □

Based on this property, a practical implementation of the BFGS method would check this condition, and make the update only if the condition is satisfied. In this case, we can guarantee that the Hessian approximation B_k is positive semi-definite. We call this variant the *careful BFGS method*.

Numerical Method 7.6: The Careful BFGS Method
The careful BFGS method is a variant of the BFGS method, where the Hessian approximation B_k is updated as follows:

$$B_{k+1} = \begin{cases} B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{s_k^\top y_k}, & \text{if } s_k^\top y_k > 0, \\ B_k, & \text{otherwise,} \end{cases} \tag{7.30}$$

where we defined, as above, $s_k \coloneqq x_k - x_{k-1}$ and $y_k \coloneqq \nabla f(x_k) - \nabla f(x_{k-1})$.
The rest of the algorithm is the same as before.

To conclude, note that another quasi-Newton method is the Davidon-Fletcher-Powell (DFP) method. It is very similar to the BFGS method, so we do not present it here.

e) Inexact Newton

When not only the Hessian, but also the gradient is inexact, we call the method *inexact Newton*. An example of such methods is when the dimension n is very large, hence the linear system below is only solved approximately:

$$\nabla^2 f(x_k) p = -\nabla f(x_k) \tag{7.31}$$

Another context in which an inexact gradient is used is for large-scale machine learning problems, where the objective function is a sum of *many* (e.g., billions) of terms. In this case, the gradient is approximated by considering only a few of these terms, which is called *mini-batching*. This approach is typically present in *Stochastic Gradient Descent* (SGD), an optimization algorithm that is widely used in machine learning.
In the next two chapters, we will consider the two following questions:

1. How fast is the convergence when the iterates are starting to approach the solution $x^\star$? See Chapter 8 about *local convergence*.

2. Can we guarantee convergence for any initial guess x_0? See Chapter 9 which is about *globalization strategies*: some tools to ensure global convergence of an algorithm without affecting its behavior for $x_k \approx x^\star$.

[2]The Cauchy-Schwarz inequality states that for any scalar product $(s, p) \mapsto s \cdot p$:

$$\forall s, p \in \mathbb{R}^n, \quad (s \cdot p)^2 \leq (s \cdot s)(p \cdot p)$$

Chapter 8

Local Convergence of Newton Type Methods

In this chapter, we discuss the *local convergence* of Newton type methods introduced earlier. To keep the analysis more general, we consider the Newton type method in its more general form as in the numerical method 7.3, to solve the root-finding problem $F(w) = 0$ as in Definition (7.2). Recall that this method is given by the following iteration:

$$w_{k+1} = w_k - M_k^{-1} F(w_k), \quad k = 0, 1, 2, \ldots, \tag{8.1}$$

8.1 Local Convergence of Newton Type for Root-Finding

Theorem 8.1: Local Contraction
Let $F(\cdot)$ be differentiable, and let $w^\star$ be a solution to the root-finding problem $F(w) = 0$. Regard the iterates given by a Newton type method, i.e. $w_{k+1} = w_k - M_k^{-1} F(w_k)$. Assume that there exist $\omega \in [0, \infty)$ and $\kappa \in [0, 1)$ such that for all w_k and $p \in \mathbb{R}^n$, the following holds

$$\left\| M_k^{-1} \left(\frac{\partial F}{\partial w}(w_k + p) - \frac{\partial F}{\partial w}(w_k) \right) \right\|_{\text{op}} \leq \omega \|p\| \qquad \text{("Lipschitz-like", or "omega", condition),}$$

$$\left\| \mathbb{I} - M_k^{-1} \frac{\partial F}{\partial w}(w_k) \right\|_{\text{op}} \leq \kappa_k \leq \kappa \qquad \text{(compatibility, or "kappa", condition).}$$

Then, the following contraction property holds:

$$\|w_{k+1} - w^\star\| \leq \left(\kappa_k + \frac{\omega}{2} \|w_k - w^\star\| \right) \|w_k - w^\star\| . \tag{8.2}$$

Remark
For exact Newton, we have $\kappa = 0$.

Remark
We used the matrix operator norm $\|\cdot\|_{\text{op}}$ defined as follows:

$$\|A\|_{\text{op}} = \max_{x\in\mathbb{R}^n, x\neq 0} \frac{\|Ax\|}{\|x\|}. \tag{8.3}$$

Remark
If the Jacobian $\frac{\partial F}{\partial w}(\cdot)$ is Lipschitz continuous with Lipschitz constant L, and the inverse M_k^{-1} is bounded by a constant $m_{\max}$, then the "Lipschitz-like" condition is satisfied with $\omega = m_{\max}L$.

Proof. First, write the first-order Taylor expansion of $F(w^\star)$ at the point w_k:

$$\begin{aligned}
&\overbrace{F(w^\star)}^{=0} = F(w_k) + \frac{\partial F}{\partial w}(w_k)(w^\star - w_k) + \overbrace{\int_0^1 \left(\frac{\partial F}{\partial w}(w_k + t(w^\star - w_k)) - \frac{\partial F}{\partial w}(w_k)\right)(w^\star - w_k)\mathrm{d}t}^{=:r_k} \\
&\iff F(w_k) = \frac{\partial F}{\partial w}(w_k)(w_k - w^\star) - r_k \\
&\iff w_{k+1} = w_k - M_k^{-1}\frac{\partial F}{\partial w}(w_k)(w_k - w^\star) + M_k^{-1}r_k.
\end{aligned} \tag{8.4}$$

The remainder r_k can be bounded using the Lipschitz-like condition as follows:

$$\begin{aligned}
\left\|M_k^{-1}r_k\right\| &= \left\|\int_0^1 M_k^{-1}\left(\frac{\partial F}{\partial w}(w_k + t(w^\star - w_k)) - \frac{\partial F}{\partial w}(w_k)\right)(w^\star - w_k)\mathrm{d}t\right\| \\
&\leq \int_0^1 \left\|M_k^{-1}\left(\frac{\partial F}{\partial w}(w_k + t(w^\star - w_k)) - \frac{\partial F}{\partial w}(w_k)\right)(w^\star - w_k)\right\| \mathrm{d}t \\
&\leq \int_0^1 \left\|M_k^{-1}\left(\frac{\partial F}{\partial w}(w_k + t(w^\star - w_k)) - \frac{\partial F}{\partial w}(w_k)\right)\right\|_{\text{op}} \mathrm{d}t \left\|w^\star - w_k\right\| \\
&\leq \int_0^1 \omega \left\|t(w^\star - w_k)\right\| \mathrm{d}t \left\|w^\star - w_k\right\| = \frac{\omega}{2}\left\|w_k - w^\star\right\|^2.
\end{aligned} \tag{8.5}$$

Combining (8.4) and (8.5), we obtain

$$\begin{aligned}
\left\|w_{k+1} - w^\star\right\| &= \left\|(w_k - w^\star) - M_k^{-1}\frac{\partial F}{\partial w}(w_k)(w_k - w^\star) + M_k^{-1}r_k\right\| \\
&\leq \left\|\left(\mathbb{I} - M_k^{-1}\frac{\partial F}{\partial w}(w_k)\right)(w_k - w^\star)\right\| + \left\|M_k^{-1}r_k\right\| \\
&\leq \kappa_k \left\|w_k - w^\star\right\| + \frac{\omega}{2}\left\|w_k - w^\star\right\|^2,
\end{aligned} \tag{8.6}$$

which proves the desired result (8.2). □

Remark
The above contraction theorem could work with slightly weaker assumptions, namely, for all $t \in [0,1]$, the following holds for $\tilde{p}_k := w^\star - w_k$:

$$\left\| M_k^{-1} \left(\frac{\partial F}{\partial w}(w_k + t\tilde{p}_k) - \frac{\partial F}{\partial w}(w_k) \right) \tilde{p}_k \right\| \leq \omega t \left\| \tilde{p}_k \right\|^2 \qquad \text{(weaker omega condition)},$$

$$\left\| \left(\mathbb{I} - M_k^{-1} \frac{\partial F}{\partial w}(w_k) \right) \tilde{p}_k \right\| \leq \kappa_k \left\| \tilde{p}_k \right\| \qquad \text{(weaker kappa condition)}.$$

This weaker omega condition is actually satisfied whenever $t \mapsto F(w_k + t\tilde{p}_k)$ is twice continuously differentiable and:

$$\forall t \in [0,1], \quad \left\| M_k^{-1} \frac{\mathrm{d}^2}{\mathrm{d}t^2} F(w_k + t\tilde{p}_k) \right\| \leq \omega \qquad \text{(weaker omega condition bis)}.$$

Corollary 8.1: Local Convergence of Newton Type Methods
Let the "omega" and "kappa" conditions of Theorem 8.1 hold, and assume, in addition, that $\|w_0 - w^\star\| < \frac{2(1-\kappa)}{\omega}$. Then the iterates w_k converge to $w^\star$.

Proof. From Theorem 8.1, we can also easily deduce that if $\|w_0 - w^\star\| < \frac{2(1-\kappa)}{\omega}$, then the iterates w_k converge to $w^\star$. Indeed, by induction, one can show that it implies a contraction with factor $\delta := \kappa + \frac{\omega}{2}\|w_0 - w^\star\| < 1$ for all k, which implies that $\|w_k - w^\star\| \leq \delta^k \|w_0 - w^\star\|$, and thus that w_k converges to $w^\star$. □

8.2 Affine Invariance

Let us now discuss the *affine invariance* of Newton type methods. Although this concept is not directly related to local convergence, it shares the same spirit: it serves as a good theoretical "sanity check" for a method.

Definition 8.1: Affine Invariance
We say that an algorithm is *affine invariant* if the iterates of the algorithm remain invariant under affine transformations of the problem data.
More rigorously, an algorithm is said to be affine invariant if it behaves similarly when applied to the following variation of the initial problem:

$$AF(By + b) = 0 \tag{8.7}$$

where $A, B \in \mathbb{R}^{n \times n}$ are invertible matrices, and $b \in \mathbb{R}^n$ is a vector.

Remark
Note that when we say the algorithm behaves similarly, we mean that $Ay_k + b = w_k$ holds if $y_0, \ldots, y_k$ are the iterates of the algorithm applied to the transformed problem (given that the initial guess y_0 corresponds to w_0).

This property is very important in practice, and it seems reasonable to seek algorithms that are affine invariant. One of the advantages of affine invariant methods is that they usually work well without requiring rescaling of the problem data. In contrast, methods that lack this property often require careful and problem-specific rescaling of the optimization variables and/or the objective function to ensure good convergence properties.

Example 8.1: Optimizing Over Temperatures
Let us consider the optimization problem of finding the optimal temperature T in a chemical reaction system.
You can formulate your equations measuring the temperature in Kelvin, Celsius, or Fahrenheit, each of which will give different numerical values denoting the same physical temperature.
Fortunately, the three values can be obtained by affine transformations from each other[1]. Therefore, the behavior of an affine invariant method will be agnostic to the choice of convention for the temperature. On the other hand, this might not be the case for other types of methods, such as the gradient descent method.

[1] $T^{\mathrm{K}} = T^{\mathrm{C}} + 273.15 = \frac{5}{9}(T^{\mathrm{F}} - 32) + 273.15$

Proposition 8.1: Affine Invariance of Several Methods
The following methods are affine invariant:

- The exact Newton method,
- The Gauss-Newton method,
- The BFGS method (provided that the initial guess B_0 is chosen in an affine invariant way).

However, the gradient descent method and the Levenberg-Marquardt method are not affine invariant.

Proof. This property follows almost directly from the construction of these methods. However, one could also prove this property explicitly, which is left as an exercise for the reader. □

8.3 Local Convergence of Newton Type for (unconstrained) Optimization

Let us now specialize the general contraction results to the case of Newton type unconstrained optimization methods.

Corollary 8.2: Local Convergence of Exact Newton for (unconstrained) Optimization
Let $x^\star$ satisfy SOSC for $f \in C^2$, i.e. $\nabla f(x^\star) = 0$ and $\nabla^2 f(x^\star) \succ 0$, and regard the iterates given by the exact Newton method, i.e., $x_{k+1} = x_k - \big(\nabla^2 f(x_k)\big)^{-1} \nabla f(x_k)$ (see Numerical Method 7.2). Let also the following conditions hold for some $\omega > 0$ and for all $p_k \in \mathbb{R}^n$:

$$\Big\| \big(\nabla^2 f(x_k)\big)^{-1} \big(\nabla^2 f(x_k + p_k) - \nabla^2 f(x_k)\big) \Big\|_{\text{op}} \leq \omega \, \|p_k\| \qquad \text{(“omega” condition)},$$

$$\|x_0 - x^\star\| < \frac{2}{\omega} \qquad \text{(initial guess condition)}.$$

Then, the iterates x_k converge to $x^\star$ with a Q-quadratic rate (with constant $\frac{\omega}{2}$).

Proof. This is also a direct consequence from Theorem 8.1 and Corollary 8.1 with $\kappa = 0$. □

Corollary 8.3: Local Convergence of Newton Type for (unconstrained) Optimization
Let $x^\star$ satisfy SOSC for $f \in C^2$, i.e. $\nabla f(x^\star) = 0$ and $\nabla^2 f(x^\star) \succ 0$, and regard the iterates given by a Newton type method, i.e., $x_{k+1} = x_k - B_k^{-1} \nabla f(x_k)$ with $B_k \succ 0$ (see Numerical Method 7.4). Let also the following conditions hold for some $\omega \in [0, \infty)$, $\kappa \in [0, 1)$, and for all $p_k \in \mathbb{R}^n$:

$$\big\| B_k^{-1} \big(\nabla^2 f(x_k + p_k) - \nabla^2 f(x_k)\big) \big\|_{\text{op}} \leq \omega \, \|p_k\| \qquad \text{(“omega” condition)},$$

$$\big\| \mathbb{I} - B_k^{-1} \nabla^2 f(x_k) \big\|_{\text{op}} \leq \kappa \qquad \text{(“kappa” condition)},$$

$$\|x_0 - x^\star\| < \frac{2(1-\kappa)}{\omega} \qquad \text{(initial guess condition)}.$$

Then, the iterates x_k converge to $x^\star$ with a Q-linear rate.
Furthermore, for Quasi-Newton methods (i.e., when $B_k \underset{k \to +\infty}{\longrightarrow} \nabla^2 f(x^\star)$), the rate is Q-superlinear.

Proof. This is also a direct consequence from Theorem 8.1 and Corollary 8.1. □

8.4 A Tight Condition for Local Convergence

The local contraction theorem of this chapter gives sufficient conditions for local convergence. The main restrictive condition is the compatibility condition (aka. the kappa condition). In the theorem below, we show that in some sense, this condition is tight, that is, in the case where M_k is a continuous function of w_k, the criterion $\rho\big(\mathbb{I} - M(w^\star)^{-1} \frac{\partial F}{\partial w}(w^\star)\big) < 1$ is a tight criterion for local convergence. This comes from *fixed-point theory*, from which we state the result that we need as a lemma below, whose proof is only sketched.

Lemma 8.1: Linear Stability Analysis
Let $G : \mathbb{R}^n \to \mathbb{R}^n$ be a continuously differentiable function in a neighborhood of a fixed point $w^\star$, i.e. $G(w^\star) = w^\star$. Then, the iterates given by $w_{k+1} = G(w_k)$ will:

- converge Q-linearly to $w^\star$ if $\rho\big(\frac{\partial G}{\partial w}(w^\star)\big) < 1$,
- diverge away[2]from $w^\star$ if $\rho\big(\frac{\partial G}{\partial w}(w^\star)\big) > 1$.

Sketch proof. Regard the first-order Taylor approximation of G at the fixed point $w^\star$:

$$\overbrace{G(w_k)}^{=w_{k+1}} \approx \overbrace{G(w^\star)}^{=w^\star} + \frac{\partial G}{\partial w}(w^\star)(w_k - w^\star) \implies w_{k+1} - w^\star \approx \frac{\partial G}{\partial w}(w^\star)(w_k - w^\star). \tag{8.8}$$

Applying this recursively, we obtain $w_k - w^\star \approx \big(\frac{\partial G}{\partial w}(w^\star)\big)^k (w_0 - w^\star)$. Finally, $\big(\frac{\partial G}{\partial w}(w^\star)\big)^k \underset{k\to+\infty}{\longrightarrow} 0$ if $\rho\big(\frac{\partial G}{\partial w}(w^\star)\big) < 1$, and $\big(\frac{\partial G}{\partial w}(w^\star)\big)^k$ diverges if $\rho\big(\frac{\partial G}{\partial w}(w^\star)\big) > 1$, which conclude the skectch of proof. □

Theorem 8.2: A tight condition for local convergence
Let $w^\star$ be a solution to the root-finding problem $F(w) = 0$ with $F(w)$ twice continuously differentiable in a neighborhood of $w^\star$. Regard the Newton type iteration of the form $w_{k+1} = w_k - M(w_k)^{-1}F(w_k)$, where $M(w)$ is continuously differentiable and invertible in a neighborhood of $w^\star$. Then:

- If $\rho\big(\mathbb{I} - M(w^\star)^{-1}\frac{\partial F}{\partial w}(w^\star)\big) < 1$, then $w_k \underset{k\to+\infty}{\longrightarrow} w^\star$ locally, with Q-linear convergence rate.
- If $\rho\big(\mathbb{I} - M(w^\star)^{-1}\frac{\partial F}{\partial w}(w^\star)\big) > 1$, then $w^\star$ is an unstable, meaning that the iterates will almost[3]never converge to $w^\star$.

Proof. Apply Lemma 8.1 to $G(w) \equiv w - M(w)^{-1}F(w)$, and remark that:

$$\frac{\partial G}{\partial w}(w^\star) = \mathbb{I} - \frac{\partial\big(M(\cdot)^{-1}\big)}{\partial w}(w^\star)\underbrace{F(w^\star)}_{=0} - M(w^\star)^{-1}\frac{\partial F}{\partial w}(w^\star) = \mathbb{I} - M(w^\star)^{-1}\frac{\partial F}{\partial w}(w^\star). \tag{8.9}$$

□

[2]Some very special cases (such as $w_0 = w^\star$) technically disproves this statement, but in practice, due to numerical errors, such an unstability implies divergence away from $w^\star$.
[3]Same remark as above.

Chapter 9

Globalization Strategies

The results that we saw in the previous chapter only guaranteed some local convergence, i.e. the method converges when the initial guess is already close to a solution. Recall for example the initial guess condition in Corollaries 8.2 and 8.3: $\|x_0 - x^\star\| \leq \frac{2(1-\kappa)}{\omega}$.
A natural question is how to ensure that Newton's method still converges when x_0 is not very close to some solution $x^\star$. The general idea is to ensure some descent at every iteration $f(x_{k+1}) < f(x_k)$, while making sure that the steps $x_{k+1} - x_k$ are not *too small*. This should result in $\nabla f(x_k) \to 0$. This descent is not ensured by the methods presented so far, where only *local information* about $f(\cdot)$ is used. The idea is to make the steps $x_{k+1} - x_k$ shorter, so that this local information becomes more meaningful.
In this chapter, two methods will be described for this: *Line-search* and *Trust-region*.

9.1 Backtracking Line-Search with Armijo Condition

9.1.1 Motivation for line-search

In this approach, we still compute the step p_k as before: $p_k = -B_k^{-1}\nabla f(x_k)$. However, instead of directly applying this step to x_k as we did before, this step will serve as a *search direction*, i.e. we will look for the next iterate along the line segment $\left\{x_k + t_k p_k \;\middle|\; t_k \in (0,1]\right\}$. Hence, the iteration now becomes:

$$x_{k+1} = x_k + t_k p_k, \tag{9.1}$$

with $t_k \in (0,1]$ a scalar called the *step-length* ($t_k = 1$ in case of a full step Newton type method). Note that by choosing $B_k \succ 0$, it is ensured that p_k is a *descent direction*[1]as long as $\nabla f(x_k) \neq 0$. Therefore, it is clear that by choosing t_k *small enough*, the descent condition $f(x_{k+1}) < f(x_k)$ will be satisfied. On the other hand, if t_k is *too small*, the algorithm may make insufficient progress. Therefore, it is a tradeoff to find. Here, t_k is chosen through a process called *line-search*.
Ideally, the line-search algorithm would solve the 1-D optimization problem:

$$\begin{array}{ll} \underset{t \in (0,1]}{\text{minimize}} & f(x_k + t p_k). \end{array} \tag{9.2}$$

This is called *exact line-search*. In practice, this is not necessary, and it would be too expensive to solve this problem at each iteration. Instead, we only ensure that a *decrease condition* holds. However, imposing only $f(x_{k+1}) < f(x_k)$ may be insufficient, as illustrated by the example below.

[1] p_k such that $\nabla f(x_k)^\top p_k < 0$, cf. Definition 7.3

Example 9.1: Insufficient decrease
Consider the very simple problem $\min_{x\in\mathbb{R}} x^2$, solved with the iterative method:

$$x_{k+1} = x_k + \underbrace{\left(1 - \frac{1}{4^k}\right)}_{t_k} \times \underbrace{(-2x_k)}_{p_k} \tag{9.3}$$

We have descent at every iteration[2]: $f(x_{k+1}) < f(x_k)$ but no convergence, because smaller and smaller progress is achieved at each step. This is illustrated in Figure 9.1 below.

[2] justification: $f(x_{k+1}) = x_{k+1}^2 = \left(1 - \frac{2}{4^k}\right)^2 x_k^2 < x_k^2 = f(x_k)$

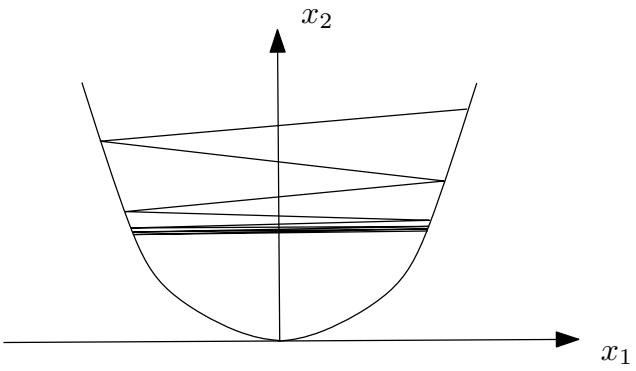


Figure 9.1: Visualization of Example 9.1

9.1.2 Armijo's sufficient decrease condition

This motivates the need for a criterion that ensures a *sufficient decrease*. The most common condition is the *Armijo condition*.

Definition 9.1: Armijo's condition
Armijo's sufficient decrease condition is the following:

$$f(x_k + t_k p_k) \leq f(x_k) + \gamma t_k \nabla f(x_k)^\top p_k, \tag{9.4}$$

with $\gamma \in (0, \frac{1}{2})$ the relaxation of the gradient.

The condition is illustrated in Figure 9.2.

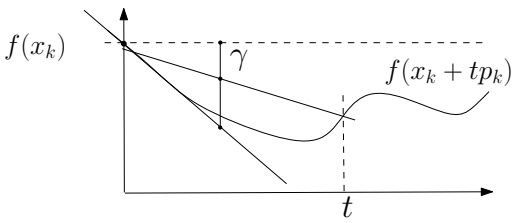


Figure 9.2: A visualization of the Armijo sufficient decrease condition.

Remark
In practice, γ is chosen quite small, say $\gamma = 0.1$ or even smaller. Note that with $\gamma = 1$, the right-hand side would be a first-order Taylor expansion (this condition would be too restrictive).
Actually, $\gamma \in [\frac{1}{2}, 1)$ would also guarantee global convergence, but this would unnecessarily restrict the step size, as we can see from the following property.
However, we would unnecessarily restrict the step size. More concretely, if the function $f(\cdot)$ is indeed quadratic, the Newton method should find the solution after one step only. But this step satisfies the Armijo condition only if $\gamma < \frac{1}{2}$.

Proposition 9.1: Armijo condition is satisfied for quadratic functions
If $f(\cdot)$ is (strictly) convex quadratic, then the Armijo condition is satisfied for the full exact Newton step if and only if $\gamma < \frac{1}{2}$.

Note that the full exact Newton step directly achieves the global minimum here.

Proof. This is a direct consequence of the following identity on strictly convex quadratic functions:

$$\forall x \in \mathbb{R}^n, \quad f(x) - f^\star = \frac{1}{2} \nabla f(x)^\top B^{-1} \nabla f(x) \tag{9.5}$$

with $f^\star$ being the minimum value of $f(\cdot)$, and B being its Hessian matrix. □

The following proposition guarantees that the Armijo condition is not too restrictive.

Proposition 9.2: Existence of Armijo point
Let f be twice continuously differentiable, $x_k \in \mathbb{R}^n$, and p_k a descent direction at x_k. Then the Armijo condition (9.4) is satisfied for all t_k sufficiently small.

Proof. Let $t_k \in (0, 1)$. Write the first-order Taylor expansion: $f(x_k + t_k p_k) = f(x_k) + t_k \nabla f(x_k)^\top p_k + r_k$, where r_k is bounded as follows:

$$|r_k| \leq \frac{L_{\text{g.}}}{2} \|t_k p_k\|^2 , \tag{9.6}$$

with $L_{\text{g.}} \coloneqq \sup_{t \in [0,1]} \left\| \nabla^2 f(x_k + t p_k) \right\|_2 < \infty$. Thus, we have:

$$f(x_k + t_k p_k) \leq f(x_k) + t_k \nabla f(x_k)^\top p_k + t_k^2 \frac{L_{\text{g.}}}{2} \|p_k\|^2 . \tag{9.7}$$

Since p_k is a descent direction (i.e., $\nabla f(x_k)^\top p_k < 0$), this implies that the Armijo condition is satisfied for all t_k such that:

$$t_k \leq \frac{2(1-\gamma)\left|\nabla f(x_k)^\top p_k\right|}{L_{\text{g.}}\left\|p_k\right\|^2}. \tag{9.8}$$

□

As we can see from this proposition, the Armijo condition is satisfied for t_k arbitrarily small, but we *do not* desire t_k to be arbitrarily small (both from the efficiency point-of-view, and for ensuring convergence). Fortunately, with *backtracking*, we can find an Armijo step-length that is guaranteed to be large enough to ensure convergence. Later, we will also mention some alternatives. Many ways exist to make sure that the steps do not get too short. Backtracking is one such method.

9.1.3 Backtracking

Numerical Method 9.1: Backtracking line-search (with Armijo condition)
Backtracking chooses the step-length by starting with $t_k = t_{\max}$, and decreasing t_k until the Armijo condition is satisfied. This reduction is done with a reduction factor $\beta \in (0, 1)$. This is described more precisely in Algorithm 1 below.

Algorithm 1 Backtracking with Armijo Condition

Parameters: γ, β, $t_{\max}$
Inputs: x_k, p_k
$t_k \leftarrow t_{\max}$
while $f(x_k + t_k p_k) \geq f(x_k) + \gamma t_k \nabla f(x_k)^\top p_k$ **do**
  $t_k \leftarrow \beta t_k$
end while
Return t_k

An important fact is that backtracking ensures to find *almost*[3]the biggest step-length that satisfies the Armijo condition. This is key to ensure global convergence, as we will see in the next part. Now we can summarize a Newton type method with backtracking line-search in Algorithm 2 below.

Remark
A remarkably good property is that, as long as $\gamma < \frac{1}{2}$, the *local convergence rate* (c.f. Chapter 8) is left unchanged by this globalization technique. This is because, given the conditions of Corollary 8.3, we have sufficient descent anyway when the iterates are close enough to the solution[4]. Therefore, backtracking is not performed: the algorithm behaves exactly in the same way as in Chapter 8

[4] we do not prove it here, but one should be easily convinced by taking a close look at Proposition 9.1 and Corollary 8.3.

It is still unclear for now whether this algorithm will ever stop. We will see below that under mild assumptions, this algorithm will indeed stop, i.e., $\nabla f(x_k) \to 0$.

[3] At least, we can ensure that $\beta\bar{t} < t_k \leq \bar{t}$, where $\bar{t}$ is the biggest Armijo step-length.

Algorithm 2 Newton type with Armijo Backtracking

Parameters: γ, β, $t_{\max}$, TOL > 0
Inputs: x_0
$k \leftarrow 0$
while $\|\nabla f(x_k)\| >$ TOL **do**
  obtain $B_k \succ 0$ (e.g. $B_k = \nabla^2 f(x_k)$ for exact Newton)
  $p_k \leftarrow -B_k^{-1} \nabla f(x_k)$
  $t_k \leftarrow t_{\max}$
  while $f(x_k + t_k p_k) \geq f(x_k) + \gamma t_k \nabla f(x_k)^\top p_k$ **do**
    $t_k \leftarrow \beta t_k$
  end while
  $x_{k+1} \leftarrow x_k + t_k p_k$
  $k \leftarrow k + 1$
end while
Output: x_k

Remark that we will need assumptions at least to prevent $f(x_k) \underset{k\to+\infty}{\longrightarrow} -\infty$. Otherwise, the algorithm takes a descent direction forever, like in the counterexample $\min_{x\in\mathbb{R}} x$.

9.2 Alternative Conditions for Globalization with Line-Search

We mainly focus on backtracking with Armijo condition, because of it is easy to implement, and ensures global convergence. However, we also briefly mention other popular line-search strategies. When backtracking is not used, one more condition is needed to ensure that the step-length t_k is not too small. This allows other line-search methods, such as bisection methods, or even scalar Newton methods. A couple of examples of such extra conditions are listed below.

- Wolfe condition:

$$\nabla f(x_k + t_k p_k)^\top p_k \geq \tilde{\gamma} \nabla f(x_k)^\top p_k, \qquad \text{with } 0 < \gamma < \tilde{\gamma} < 1. \tag{9.9}$$

- Strong Wolfe condition:

$$\left|\nabla f(x_k + t_k p_k)^\top p_k\right| \leq \tilde{\gamma} \left|\nabla f(x_k)^\top p_k\right|, \qquad \text{with } 0 < \gamma < \tilde{\gamma} < 1. \tag{9.10}$$

- Goldstein condition:

$$f(x_k + t_k p_k) \geq f(x_k) + (1-\gamma) t_k \nabla f(x_k)^\top p_k. \tag{9.11}$$

9.3 Analysis of Backtracking Line Search with Armijo Condition

Now we will prove that under mild assumptions, a Newton type method combined with Armijo backtracking converges globally. We first state a theorem about the termination of Algorithm 2. Then we will prove convergence of the iterates with the help of a lemma.

Theorem 9.1: Termination of Newton type methods with Armijo backtracking
Assume the following three assumptions (two of them are on the function $f(\cdot)$, and one on the method itself).

i. Bounded Hessian: $f \in C^2$ and $\nabla^2 f(\cdot)$ is bounded:

$$\sup_{x\in\mathbb{R}^n} \left\|\nabla^2 f(x)\right\| < \infty. \tag{9.12}$$

ii. $f(\cdot)$ is lower-bounded:

$$\text{for some } f_{\min} \in \mathbb{R}: \quad \forall x \in \mathbb{R}^n,\ f(x) \geq f_{\min}. \tag{9.13}$$

iii. Bounds on the Hessian approximation: $B_k \succ 0$ uniformly bounded:

$$\text{for some } c_1, c_2 > 0: \quad \forall k \geq 0:\ c_1\mathbb{I} \preccurlyeq B_k \preccurlyeq c_2\mathbb{I}, \tag{9.14}$$

Then, $\nabla f(x_k) \underset{k\to+\infty}{\longrightarrow} 0$ where $\{x_k\}_{k\in\mathbb{N}}$ are the iterates of the Newton type method with Armijo backtracking.
Notably, this proves that Algorithm 2 stops after a finite number of iterations.

Proof. Recall from Proposition 9.2 (and its proof) that the Armijo condition is satisfied whenever the following condition holds:

$$t_k \leq \frac{2(1-\gamma)\left|\nabla f(x_k)^\top p_k\right|}{L_{\text{g.}}\left\|p_k\right\|^2}. \tag{9.15}$$

Using the definition of the method, $p_k = -B_k^{-1}\nabla f(x_k)$, and the bounds (9.14) on B_k, we can lower bound the right-hand side by $\frac{2(1-\gamma)c_1^2}{L_{\text{g.}}c_2}$. On the other hand, according to the backtracking principle, either $t_k = t_{\max}$, or $\frac{t_k}{\beta}$ failed to satisfy the Armijo condition, which means that $\frac{t_k}{\beta} > \frac{2(1-\gamma)c_1^2}{L_{\text{g.}}c_2}$. Overall, we have:

$$t_k \geq \underbrace{\min\left\{t_{\max},\ \frac{2(1-\gamma)c_1^2\beta}{L_{\text{g.}}c_2}\right\}}_{=:t_{\min}}. \tag{9.16}$$

Since the Armijo condition is satisfied at each iteration, we also have $f(x_{k+1}) \leq f(x_k) + \gamma t_k \nabla f(x_k)^\top p_k$. Using once again the bounds on B_k, together with the definition of p_k and $t_k \geq t_{\min}$, we obtain:

$$f(x_{k+1}) \leq f(x_k) - \underbrace{\frac{\gamma t_{\min}}{c_2}}_{C}\left\|\nabla f(x_k)\right\|^2. \tag{9.17}$$

This proves, notably, that the sequence $\{f(x_k)\}_{k\in\mathbb{N}}$ is non-increasing. Thus, as it is lower-bounded by $f_{\min}$, it converges. Hence $f(x_{k+1}) - f(x_k) \underset{k\to+\infty}{\longrightarrow} 0$. Finally, plugging this into equation (9.17), we deduce that $\|\nabla f(x_k)\| \underset{k\to+\infty}{\longrightarrow} 0$. □

Theorem 9.1 can be viewed as the main result here. Indeed, it shows that the algorithm terminates and that the first-order optimality conditions are satisfied *asymptotically*. Furthermore, it is easy

to convince ourselves that, based on the present assumptions, we cannot really hope for a better result in some sense. Indeed, there exist counterexamples such as $\min_{x\in\mathbb{R}} e^{-x}$, where any decent algorithm would find $x_k \underset{k\to+\infty}{\longrightarrow} +\infty$, yet this problem satisfies the assumptions of Theorem 9.1. Another type of counterexample arises when the minimizers are not isolated, and the algorithm could *converge to a region of minimizers* instead of converging to a point.

*Global Convergence Proof (more in-depth analysis)

It might be frustrating to stop here because we can feel that we are close to proving convergence of the algorithm. This is why, in the following theorem, we add a couple of assumptions to prove global convergence. The proof is not immediate, so we will need first a lemma from topology. This lemma might sound difficult for someone not used to these topological arguments, but it is in reality very geometric.

Lemma 9.1: A lemma from topology
Let $\{x_k\}_{k\in\mathbb{N}}$ be a sequence that satisfies the following three properties:

i. the sequence is bounded;

ii. the limit points are isolated[5];

iii. the differences converge: $x_{k+1} - x_k \underset{k\to+\infty}{\longrightarrow} 0$;

then the sequence $\{x_k\}_{k\in\mathbb{N}}$ converges.

[5] i.e., for any limit point $\bar{x}$, there exists a neighborhood $N_{\bar{x}}$ of $\bar{x}$ that contains no other limit point.

Proof. From the Bolzano-Weierstrass theorem and the first assumption, there is at least one limit point $\bar{x} \in \mathbb{R}^n$. From the second assumption, we know that if we choose $\varepsilon > 0$ small enough, every other limit point $\bar{x}'$ will satisfy $\|\bar{x}' - \bar{x}\| > \varepsilon$. Therefore, the hollow sphere S defined as follows contains no limit point:

$$S \coloneqq \left\{ x \in \mathbb{R}^n \;\middle|\; \frac{\varepsilon}{2} \leq \|x - \bar{x}\| \leq \varepsilon \right\} \tag{9.18}$$

This set being compact and with no limit point of $\{x_k\}_{k\in\mathbb{N}}$, it contains at most finitely many points of the sequence (otherwise, the Bolzano-Weierstrass theorem would apply). Hence, for k large enough, $x_k \notin S$.
On the other hand, the third assumption indicates that $\|x_{k+1} - x_k\| < \varepsilon/2$ for k large enough. This implies that the iterates cannot "jump over" S, i.e., either they remain on one side of S or they remain on the other side. Mathematically, either $\|x_k - \bar{x}\| < \frac{\varepsilon}{2}$ *for all* k large enough, or $\|x_k - \bar{x}\| > \varepsilon$ *for all* k large enough. But the second possibility is impossible, because $\bar{x}$ is a limit point. Therefore, it is the first possibility that is valid: $\|x_k - \bar{x}\| < \frac{\varepsilon}{2}$ for all k large enough. This property holds for any $\varepsilon > 0$, which proves $x_k \underset{k\to+\infty}{\longrightarrow} \bar{x}$. □

Now, combining Theorem 9.1 with the lemma above, we can prove global convergence of the Newton type method with Armijo backtracking.

Theorem 9.2: Global convergence of Newton type with Armijo backtracking
Assume, in addition to the assumptions from Theorem 9.1, the two following:

i. the sublevel set $\left\{x \in \mathbb{R}^n \;\middle|\; f(x) \leq f(x_0)\right\}$ is bounded;

ii. the stationary points, i.e., the points $\bar{x} \in \mathbb{R}^n$ such that $\nabla f(\bar{x}) = 0$, are *isolated*.

Then, the Newton type method with Armijo backtracking converges globally.

Proof. We simply prove the three conditions of Lemma 9.1 for the sequence $\{x_k\}_{k\in\mathbb{N}}$ generated by Algorithm 2.

- *The sequence is bounded:* $f(x_k)$ is decreasing from Armijo's condition, hence the iterates remain in the sublevel set $\left\{x \in \mathbb{R}^n \;\middle|\; f(x) \leq f(x_0)\right\}$, which is itself bounded by assumption. Therefore, the sequence $\{x_k\}_{k\in\mathbb{N}}$ is bounded.
- *The limit points are isolated:* using Theorem 9.1: $\nabla f(x_k) \underset{k\to+\infty}{\longrightarrow} 0$, so by continuity of $\nabla f(\cdot)$, the limit points satisfy $\nabla f(\bar{x}) = 0$, which describes a set of isolated points by assumption.
- *The differences converge:* we simply have:

$$\|x_{k+1} - x_k\| = t_k \left\|B_k^{-1}\nabla f(x_k)\right\| \leq t_{\max} c_1^{-1} \|\nabla f(x_k)\| \underset{k\to+\infty}{\longrightarrow} 0, \tag{9.19}$$

Therefore, from Lemma 9.1, we conclude that the sequence $\{x_k\}_{k\in\mathbb{N}}$ converges. Its limit is a stationary point from the continuity of $\nabla f(\cdot)$ and the result from Theorem 9.1. □

9.4 Trust-Region Methods (TR)

9.4.1 Motivation for trust-region methods

Let us revisit the motivation for Newton methods discussed in the previous chapter. One of the motivations was that we minimize a quadratic approximation of the objective:

$$\text{if } x \approx x_k: \quad f(x) \approx m_k(x) = f(x_k) + \nabla f(x_k)^\top (x - x_k) + \frac{1}{2}(x - x_k)^\top B_k (x - x_k). \tag{9.20}$$

This approximation is only valid for $x \approx x_k$. Therefore, to ensure global convergence, we need to restrict the step size $x_{k+1} - x_k$ to be small enough.
In line-search methods, we used the model $m_k(\cdot)$ to compute a direction p_k and then backtracked along this direction to ensure that the step is not too large.
In trust-region methods, we restrict the step size *a priori*, i.e., before computing the search direction. This is somewhat similar to the idea behind the Levenberg-Marquardt method (cf. Numerical Method 6.3), except that instead of penalizing the step, we constrain it. This allows us to *control the step size* more directly.

9.4.2 The Trust-Region algorithm

As discussed above, the Trust-Region (TR) algorithm can be formulated as follows:

Numerical Method 9.2: Trust-Region (TR) algorithm
The TR algorithm is an iterative method where, at each step, the next iterate x_{k+1} is computed by solving the following TR subproblem:

$$\begin{aligned} &\underset{x \in \mathbb{R}^n}{\text{minimize}} && m_k(x) && \text{(9.21a)} \\ &\text{subject to} && \|x - x_k\| \leq \Delta_k, && \text{(9.21b)} \end{aligned}$$

where $m_k(\cdot)$ is the quadratic model defined in (9.20) and $\Delta_k > 0$ is the trust-region radius at iteration k.

Remark
The norm in (9.21b) is typically the L_1, the L_2, or the L_∞ norm. If L_1 or L_∞ is used, the problem becomes a QP. If L_2 is used, the problem is a Quadratically Constrained Quadratic Program (QCQP).

Remark
A major advantage of TR algorithms is that they do not require the Hessian to be positive definite. Indeed, the TR subproblem (9.21) is always well-defined because the trust-region constraint (9.21b) ensures that the feasible set is bounded. This is not the case for line-search methods, where the Hessian must be positive definite to ensure that the quadratic model is bounded below.

9.4.3 Choosing the radius Δ_k

For now, we have not discussed how the radius Δ_k is chosen. It should be small enough to ensure convergence but not too small to avoid excessively slow convergence. The best strategy is to adapt it based on the performance of each step. The idea is to shrink the trust-region when the step is not successful and to enlarge it when it is. This is why, instead of directly using the solution of (9.21) for x_{k+1}, we call it $\tilde{x}_{k+1}$, and we set the next iterate x_{k+1} according to the following rule:

$$x_{k+1} = \begin{cases} \tilde{x}_{k+1} & \text{if this step seems trustworthy,} \\ x_k & \text{otherwise.} \end{cases} \tag{9.22}$$

This motivates the need for a way to quantify the *trustworthiness* of the model of a step.

Definition 9.2: Trustworthiness
A measure for the *trustworthiness* of a model is the ratio of actual and predicted reduction:

$$\rho_k = \frac{f(x_k) - f(\tilde{x}_{k+1})}{\underbrace{m_k(x_k) - m_k(\tilde{x}_{k+1})}_{>0 \text{ if } \|\nabla f(x_k)\| \neq 0}} = \frac{\text{actual reduction}}{\text{predicted reduction}}. \tag{9.23}$$

Note that descent is guaranteed only for $\rho_k > 0$, and $\rho_k \approx 1$ means that the model is as good as expected. The idea is to check how large ρ_k is and, depending on that, decide whether to shrink or enlarge the trust-region, and also whether to accept the step or not. This discussion leads to

the design of Algorithm 3 below, with some positive constants ρ_{bad}, ρ_{good}, and ρ_{trust}. A general

Algorithm 3 Trust-Region

Parameters: $\Delta_{\max}$, $\rho_{\text{bad}}, \rho_{\text{good}}, \rho_{\text{trust}}$, Δ_0, x_0, TOL > 0

$k = 0$
while $\|\nabla f(x_k)\| >$ TOL **do**
  Solve the TR subproblem (9.21) (approximately) to get $\tilde{x}_{k+1}$
  Compute the trustworthiness measure ρ_k according to (9.23)
  if $\rho_k < \rho_{\text{bad}}$ **then**
    $\Delta_{k+1} \leftarrow \Delta_k \cdot \frac{1}{4}$ *(the model is bad)*
  else if $\rho_k > \rho_{\text{good}}$ and $\|\tilde{x}_{k+1} - x_k\| = \Delta_k$ **then**
    $\Delta_{k+1} \leftarrow \min(2 \cdot \Delta_k, \Delta_{\max})$ *(the model is good)*
  else
    $\Delta_{k+1} \leftarrow \Delta_k$ *(the model is good)*
  end if
  if $\rho_k > \rho_{\text{trust}}$ **then**
    $x_{k+1} \leftarrow \tilde{x}_{k+1}$ *(the step is good)*
  else
    $x_{k+1} \leftarrow x_k$ *(the step is bad)*
  end if
  $k \leftarrow k + 1$
end while
Return x_k

convergence proof of the TR algorithm can be found in Theorem 4.5 in [4].

9.5 Trust Region Approximation: the Cauchy Point

To guarantee that a trust-region method converges globally, one does not *need* to solve the TR subproblem (9.21) exactly. Instead, one can approximate the solution of the TR subproblem. One way of doing so is to solve the TR subproblem approximately in the direction of the gradient. This results in a new point called the Cauchy point.

Numerical Method 9.3: Cauchy point
The Cauchy point is the point that minimizes the TR model along the steepest descent direction. More precisely, it is the point $x_{k+1}^{\mathrm{C}} = x_k - t_k^{\mathrm{C}} \nabla f(x_k)$ where t_k^{C} is given by:

$$\underset{t \in \mathbb{R}}{\text{minimize}} \quad m_k(x_k - t_k \nabla f(x_k)) \tag{9.24a}$$

$$\text{subject to} \quad \|t_k \nabla f(x_k)\| \leq \Delta_k, \tag{9.24b}$$

This can be solved exactly with very low computational cost because it can be solved explicitly[6]:

$$t_k^{\mathrm{C}} = \min \left\{ \frac{\Delta_k}{\|\nabla f(x_k)\|}, \quad \frac{\|\nabla f(x_k)\|_2^2}{\nabla f(x_k)^\top B_k \nabla f(x_k)} \right\}. \tag{9.25}$$

[6] This formula is valid only for $B_k \succ 0$. We leave it to the reader to find the more general formula.

It is remarkable that using the Cauchy point is sufficient to ensure global convergence of Algorithm 3 (see [4]). However, in practice, this algorithm is quite slow (compared to Newton type methods).
Many algorithms are based on the idea of *improving the Cauchy point*. The motivation is to improve the *local convergence rate* while maintaining *global convergence guarantees*. For example, one popular algorithm is the *dogleg method*, described below.

Numerical Method 9.4: The dogleg method
In the dogleg method, we choose a combination of the Cauchy point and the full Newton type step:

$$x_{k+1} = (1 - \alpha_k) x_{k+1}^{\mathrm{NT}} + \alpha_k x_{k+1}^{\mathrm{C}}, \tag{9.26}$$

with $x_{k+1}^{\mathrm{NT}} = x_k - B_k^{-1} \nabla f(x_k)$ and x_{k+1}^{C} as defined above. For $\alpha_k \geq 0$, we choose it as small as possible, i.e., by solving the following simple problem:

$$\underset{\alpha_k \in [0,1]}{\text{minimize}} \quad \alpha_k \tag{9.27a}$$

$$\text{subject to} \quad \left\| x_k - \left((1 - \alpha_k) x_{k+1}^{\mathrm{NT}} + \alpha_k x_{k+1}^{\mathrm{C}} \right) \right\| \leq \Delta_k, \tag{9.27b}$$

Remark
The name "dogleg" comes from the fact that the iterates x_{k+1} are optimized within a bent line that connects the three points x_k, x_{k+1}^{C}, and x_{k+1}^{NT}.
This bent line looks a bit like the leg of a dog.

Remark
As for the backtracking linesearch approach, the dogleg method has the remarkable property of being both globally convergent, and locally as fast as the exact Newton method.

Chapter 10

Calculating Derivatives

In the previous chapters, we saw that we regularly need to calculate ∇f and $\nabla^2 f$. There are several methods for calculating these derivatives:

1. **By hand** Expensive and error-prone.
2. **Symbolic differentiation** Using Mathematica or Maple. The disadvantage is that the result is often a very long code and expensive to evaluate.
3. **Finite differences** *"Easy and fast, but inaccurate".* This method can always be applied, even if the function to be differentiated is only available as black-box code. To approximate the derivative, we use the fact that for any twice differentiable function

$$\frac{\partial f(x+tp) - f(x)}{\partial t} = \nabla f(x)^\top p + \mathcal{O}(t). \tag{10.1}$$

   Thus, we can take the left-hand side as an approximation of the directional derivative $\nabla f(x)^\top p$. But how should we choose t? If we take t too small, the approximation will suffer from numerical cancellation errors. On the other hand, if we take t too large, the linearization errors will be dominant. A good rule of thumb is to use $t = \sqrt{\varepsilon_{\text{mach}}}$, with $\varepsilon_{\text{mach}}$ the machine precision (or the precision of f, if it is lower than the machine precision). The accuracy of this method is $\sqrt{\varepsilon_{\text{mach}}}$, which means in practice that we lose half the valid digits compared to the function evaluation. Second-order derivatives are even more difficult to calculate accurately.
4. **Algorithmic Differentiation (AD)** This is the main topic of this chapter.

10.1 Algorithmic Differentiation (AD)

Algorithmic differentiation uses the fact that each differentiable function $F : \mathbb{R}^n \to \mathbb{R}^{n_F}$ is composed of several *elementary operations*, like multiplication, division, addition, subtraction, sine-functions, exp-functions, etc. If the function is written in a programming language like, e.g., C, C++, or FORTRAN, special AD-tools can have access to all these elementary operations. They can process the code in order to generate new code that does not only deliver the function value but also desired derivative information. Algorithmic differentiation was traditionally called *automatic differentiation*, but as this might lead to confusion with symbolic differentiation, most AD people

now prefer the term *algorithmic differentiation*, which fortunately has the same abbreviation. An authoritative textbook on AD is [3]. In order to see how AD works, let us regard a function $F : \mathbb{R}^n \to \mathbb{R}^{n_F}$ that is composed of a sequence of m elementary operations. While the inputs $x_1, \ldots, x_n$ are given before, each elementary operation ϕ_i, $i = 0, \ldots, m-1$, generates another intermediate variable, x_{n+i+1}. Some of these intermediate variables are used as output of the code, which we might call $y_1, \ldots, y_{n_F}$ here. The vector $y \in \mathbb{R}^{n_F}$ can be obtained from the vector of all previous variables $x \in \mathbb{R}^{n+m}$ by the expression $y = Cx$ with a selection matrix $C \in \mathbb{R}^{n_F \times (n+m)}$ that consists only of zeros and ones and has only one one in each row. This way to regard a function evaluation is stated in Algorithm 4 and illustrated in Example 10.1 below.

Algorithm 4 User Function Evaluation via Elementary Operations

Input: $x_1, \ldots, x_n$
Output: $y_1, \ldots, y_{n_F}$

for $i = 0$ to $m - 1$ **do**
  $x_{n+i+1} \leftarrow \phi_i(x_1, \ldots, x_{n+i})$
end for

for $j = 0$ to n_F **do**
  $y_j = \sum_{i=1}^{n+m} C_{ji} x_i$
end for
Remark 1: each ϕ_i depends on only one or two out of $\{x_1, \ldots, x_{n+i}\}$.
Remark 2: the selection of y_j from x_i creates no computational costs.

Example 10.1: Function Evaluation via Elementary Operations
Let us regard the simple scalar function

$$f(x_1, x_2, x_3) = \sin(x_1 x_2) + \exp(x_1 x_2 x_3) \tag{10.2}$$

with $n = 3$. We can decompose this function into $m = 5$ elementary operations, namely

$$x_4 = x_1 x_2 \tag{10.3a}$$
$$x_5 = \sin(x_4) \tag{10.3b}$$
$$x_6 = x_4 x_3 \tag{10.3c}$$
$$x_7 = \exp(x_6) \tag{10.3d}$$
$$x_8 = x_5 + x_7 \tag{10.3e}$$
$$y_1 = x_8 \tag{10.3f}$$

Thus, if the $n = 3$ inputs x_1, x_2, x_3 are given, the $m = 5$ elementary operations $\phi_0, \ldots, \phi_4$ compute the $m = 5$ intermediate quantities, $x_4, \ldots, x_8$. The last row defines that our desired output is x_8, i.e., the selection matrix C is in this example given by

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{10.4}$$

The idea of AD is to use the chain rule and differentiate each of the elementary operations ϕ_i separately. There are two modes of AD, on the one hand the "forward" mode of AD, and on the other hand the "backward", "reverse", or "adjoint" mode of AD. In order to present both of them in a consistent form, we first introduce an alternative formulation of the original user function, that uses augmented elementary functions, as follows[1]: we introduce new augmented states

$$\tilde{x}_0 = x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \tilde{x}_1 = \begin{bmatrix} x_1 \\ \vdots \\ x_{n+1} \end{bmatrix}, \quad \ldots, \quad \tilde{x}_m = \begin{bmatrix} x_1 \\ \vdots \\ x_{n+m} \end{bmatrix} \tag{10.5}$$

as well as new augmented elementary functions $\tilde{\phi}_i : \mathbb{R}^{n+i} \to \mathbb{R}^{n+i+1}$, $\tilde{x}_i \mapsto \tilde{x}_{i+1} = \tilde{\phi}_i(\tilde{x}_i)$ with

$$\tilde{\phi}_i(\tilde{x}_i) = \begin{bmatrix} x_1 \\ \vdots \\ x_{n+i} \\ \phi_i(x_1, \ldots, x_{n+i}) \end{bmatrix}, \quad i = 0, \ldots, m-1. \tag{10.6}$$

Thus, the whole evaluation tree of the function can be summarized as a concatenation of these augmented functions followed by a multiplication with the selection matrix C that selects from $\tilde{x}_m$ the final outputs of the computer code.

$$F(x) = C \cdot \tilde{\phi}_{m-1}(\tilde{\phi}_{m-2}(\cdots \tilde{\phi}_1(\tilde{\phi}_0(x)))). \tag{10.7}$$

The full Jacobian of F, that we denote by $J_F = \frac{\partial F}{\partial x}$, is given by the chain rule as the product of the Jacobians of the augmented elementary functions $\tilde{J}_i = \frac{\partial \tilde{\phi}_i}{\partial \tilde{x}_i}$, as follows:

$$J_F = C \cdot \tilde{J}_{m-1} \cdot \tilde{J}_{m-2} \cdots \tilde{J}_1 \cdot \tilde{J}_0. \tag{10.8}$$

Note that each elementary Jacobian is given as a unit matrix plus one extra row. Also note that the extra row that is here marked with stars $*$ has at maximum two non-zero entries.

$$\tilde{J}_i = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ * & * & * & * & \end{bmatrix}. \tag{10.9}$$

For the generation of first-order derivatives, algorithmic differentiation uses two alternative ways to evaluate the product of these Jacobians, the *forward* and the *backward mode* as described in the next sections.

10.2 The Forward Mode of AD

In forward AD, we first define a *seed vector* $p \in \mathbb{R}^n$ and then evaluate the directional derivative $J_F p$ in the following way:

$$J_F p = C \cdot (\tilde{J}_{m-1} \cdot (\tilde{J}_{m-2} \cdots (\tilde{J}_1 \cdot (\tilde{J}_0 p)))). \tag{10.10}$$

[1]MD thanks Carlo Savorgnan for having outlined to him this way of presenting forward and backward AD

In order to write down this long matrix product as an efficient algorithm where the multiplications of all the ones and zeros do not cause computational costs, it is customary in the field of AD to use a notation that uses "dot quantities" $\dot{x}_i$ that we might think of as the velocity with which a certain variable changes, given that the input x changes with speed $\dot{x} = p$. We can interpret them as

$$\dot{x}_i \equiv \frac{dx_i}{dx} p. \tag{10.11}$$

In the augmented formulation, we can introduce dot quantities $\dot{\tilde{x}}_i$ for the augmented vectors $\tilde{x}_i$, for $i = 0, \ldots, m-1$, and the recursion of these dot quantities is just given by the initialization with the seed vector, $\dot{\tilde{x}}_0 = p$, and then the recursion

$$\dot{\tilde{x}}_{i+1} = \tilde{J}_i(\tilde{x}_i)\dot{\tilde{x}}_i, \quad i = 0, 1, \ldots, m-1. \tag{10.12}$$

Given the special structure of the Jacobian matrices, most elements of $\dot{\tilde{x}}_i$ are only multiplied by one, and nothing needs to be done apart from the computation of the last component of the new vector $\dot{\tilde{x}}_{i+1}$. This last component is $\dot{x}_{n+i+1}$. Thus, in an efficient implementation, the forward AD algorithm works as the algorithm below. It first sets the seed $\dot{x} = p$ and then proceeds as follows.

Algorithm 5 Forward Automatic Differentiation

Input: $\dot{x}_1, \ldots, \dot{x}_n$ and all partial derivatives $\frac{\partial \phi_i}{\partial x_j}$
Output: $\dot{x}_1, \ldots, \dot{x}_{n+m}$

for $i = 0$ to $m-1$ **do**
  $\dot{x}_{n+i+1} \leftarrow \sum_{j=1}^{n+i} \frac{\partial \phi_i}{\partial x_j} \dot{x}_j$
end for

Note: each sum consists of only one or two non-zero entries.

In forward AD, the function evaluation and the derivative evaluation can be performed in parallel, which eliminates the need to store any internal information. This is best illustrated using an example.

Example 10.2: Forward Automatic Differentiation
We regard the same example as above, $f(x_1, x_2, x_3) = \sin(x_1 x_2) + \exp(x_1 x_2 x_3)$. First, each intermediate variable has to be computed, and then each line can be differentiated. For given x_1, x_2, x_3 and $\dot{x}_1, \dot{x}_2, \dot{x}_3$, the algorithm proceeds as follows:

$$x_4 = x_1 x_2 \qquad \dot{x}_4 = \dot{x}_1 x_2 + x_1 \dot{x}_2 \tag{10.13a}$$
$$x_5 = \sin(x_4) \qquad \dot{x}_5 = \cos(x_4)\dot{x}_4 \tag{10.13b}$$
$$x_6 = x_4 x_3 \qquad \dot{x}_6 = \dot{x}_4 x_3 + x_4 \dot{x}_3 \tag{10.13c}$$
$$x_7 = \exp(x_6) \qquad \dot{x}_7 = \exp(x_6)\dot{x}_6 \tag{10.13d}$$
$$x_8 = x_5 + x_7 \qquad \dot{x}_8 = \dot{x}_5 + \dot{x}_7 \tag{10.13e}$$

The result is $\dot{x}_8 = (\dot{x}_1, \dot{x}_2, \dot{x}_3)\nabla f(x_1, x_2, x_3)$.

It can be proven that the computational cost of Algorithm 5 is smaller than two times the cost of Algorithm 4, or short

$$\text{cost}(J_F p) \leq 2 \text{ cost}(F). \tag{10.14}$$

If we want to obtain the full Jacobian of F, we need to call Algorithm 5 several times, each time with the seed vector corresponding to one of the n unit vectors in $\mathbb{R}^n$, i.e.,

$$\dot{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}. \tag{10.15}$$

Thus, we have for the computation of the full Jacobian

$$\text{cost}(J_F) \leq 2\, n \text{ cost}(F). \tag{10.16}$$

AD in forward mode is slightly more expensive than numerical finite differences, but it is exact up to machine precision.

The "Imaginary Trick" in MATLAB

An easy way to obtain high-precision derivatives in MATLAB is closely related to AD in forward mode. It is based on the following observation: If $F : \mathbb{R}^n \to \mathbb{R}^{n_F}$ is analytic and can be extended to complex numbers as inputs and outputs, then for any $t > 0$ holds

$$J_F(x)p = \frac{\partial \text{imag}\left(F(x + itp)\right)}{\partial t} + \mathcal{O}(t^2). \tag{10.17}$$

Proof. We define a function g in a complex scalar variable $z \in \mathbb{C}$ as $g(z) = F(x + zp)$ and then look at its Taylor expansion:

$$g(z) = g(0) + g'(0)z + \frac{1}{2}g''(0)z^2 + \mathcal{O}(z^3) \tag{10.18a}$$

$$g(it) = g(0) + g'(0)it + \frac{1}{2}g''(0)i^2t^2 + \mathcal{O}(t^3) \tag{10.18b}$$

$$= g(0) - \frac{1}{2}g''(0)t^2 + g'(0)it + \mathcal{O}(t^3) \tag{10.18c}$$

$$\text{imag}\,(g(it)) = g'(0)t + \mathcal{O}(t^3) \tag{10.18d}$$

□

In contrast to finite differences, there is no subtraction in the numerator, so there is no danger of numerical cancellation errors, and t can be chosen extremely small, e.g., $t = 10^{-100}$, which means that we can compute the derivative up to machine precision. This "imaginary trick" can most easily be used in a programming language that does not declare the type of variables beforehand, so that real-valued variables can automatically be overloaded with complex-valued variables. This allows us to obtain high-precision derivatives of a given black-box code. We only need to be sure that the code is *analytic* (i.e., differentiable in the sense of complex numbers).

10.3 The Backward Mode of AD

In backward AD, we evaluate the product in Eq. (10.8) in the reverse order compared with forward AD. Backward AD does not evaluate forward directional derivatives. Instead, it evaluates *adjoint directional derivatives*. When we define a *seed vector* $\lambda \in \mathbb{R}^{n_F}$, backward AD is able to evaluate the product $\lambda^\top J_F$. It does so in the following way:

$$\lambda^\top J_F = ((((\lambda^\top C) \cdot \tilde{J}_{m-1}) \cdot \tilde{J}_{m-2}) \cdots \tilde{J}_1) \cdot \tilde{J}_0. \tag{10.19}$$

When writing this matrix product as an algorithm, we use "bar quantities" instead of the "dot quantities" that we used in the forward mode. These quantities can be interpreted as derivatives of the final output with respect to the respective intermediate quantity. We can interpret

$$\bar{x}_i \equiv \lambda^\top \frac{dF}{dx_i}. \tag{10.20}$$

Each intermediate variable has a bar variable, and at the start, we initialize all bar variables with the value that we obtain from $C^\top\lambda$. Note that most of these seeds will usually be zero, depending on the output selection matrix C. Then, the backward AD algorithm modifies all bar variables. Backward AD gets most transparent in the augmented formulation, where we have bar quantities $\bar{\tilde{x}}_i$ for the augmented states $\tilde{x}_i$. We can transpose the above Equation (10.19) in order to obtain

$$J_F^\top\lambda = \tilde{J}_0^\top \cdot (\underbrace{\tilde{J}_1^\top \cdots \tilde{J}_{m-1}^\top \underbrace{(C^\top\lambda)}_{=\bar{\tilde{x}}_m}}_{=\bar{\tilde{x}}_{m-1}}). \tag{10.21}$$

In this formulation, the initialization of the backward seed is nothing else than setting $\bar{\tilde{x}}_m = C^\top\lambda$ and then going in reverse order through the recursion

$$\bar{\tilde{x}}_i = \tilde{J}_i(\tilde{x}_i)^\top \bar{\tilde{x}}_{i+1}, \quad i = m-1, m-2, \ldots, 0. \tag{10.22}$$

Again, the multiplication with ones does not cause any computational cost, but an interesting feature of the reverse mode is that some of the bar quantities can get modified several times in very different stages of the algorithm. Note that the multiplication $\tilde{J}_i^\top \bar{\tilde{x}}_{i+1}$ with the transposed Jacobian

$$\tilde{J}_i^\top = \begin{bmatrix} 1 & & & * \\ & 1 & & * \\ & & \ddots & * \\ & & 1 & * \end{bmatrix}. \tag{10.23}$$

modifies at maximum two elements of the vector $\bar{\tilde{x}}_{i+1}$ by adding to them the partial derivative of the elementary operation multiplied with $\bar{x}_{n+i+1}$. In an efficient implementation, the backward AD algorithm looks as follows.

Algorithm 6 Reverse Automatic Differentiation

Input: seed vector $\bar{x}_1, \ldots, \bar{x}_{n+m}$ and all partial derivatives $\frac{\partial \phi_i}{\partial x_j}$
Output: $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$

for $i = m - 1$ down to 0 **do**
  for all $j = 1, \ldots, n + i$ **do**
    $\bar{x}_j \leftarrow \bar{x}_j + \bar{x}_{n+i+1} \frac{\partial \phi_i}{\partial x_j}$
  end for
end for

Note: each inner loop will only update one or two bar quantities.

Example 10.3: Reverse Automatic Differentiation
We regard the same example as before and want to compute the gradient $\nabla f(x) = (\bar{x}_1, \bar{x}_2, \bar{x}_3)^\top$ given (x_1, x_2, x_3). We set $\lambda = 1$. Because the selection matrix C selects only the last intermediate variable as output, i.e., $C = (0, \cdots 0, 1)$, we initialize the seed vector with zeros apart from the last component, which is one. In the reverse mode, the algorithm first has to evaluate the function with all intermediate quantities, and only then it can compute the bar quantities, which it does in reverse order. At the end, it obtains, among others, the desired quantities $(\bar{x}_1, \bar{x}_2, \bar{x}_3)$. The full algorithm is the following.

// *** forward evaluation of the function ***
$x_4 = x_1 x_2$
$x_5 = \sin(x_4)$
$x_6 = x_4 x_3$
$x_7 = \exp(x_6)$
$x_8 = x_5 + x_7$

// *** initialization of the seed vector ***
$\bar{x}_i = 0, \quad i = 1, \ldots, 7$
$\bar{x}_8 = 1$

// *** backwards sweep ***
// * differentiation of $x_8 = x_5 + x_7$
$\bar{x}_5 = \bar{x}_5 + 1\,\bar{x}_8$
$\bar{x}_7 = \bar{x}_7 + 1\,\bar{x}_8$
// * differentiation of $x_7 = \exp(x_6)$
$\bar{x}_6 = \bar{x}_6 + \exp(x_6)\bar{x}_7$
// * differentiation of $x_6 = x_4 x_3$
$\bar{x}_4 = \bar{x}_4 + x_3\bar{x}_6$
$\bar{x}_3 = \bar{x}_3 + x_4\bar{x}_6$
// * differentiation of $x_5 = \sin(x_4)$
$\bar{x}_4 = \bar{x}_4 + \cos(x_4)\bar{x}_5$
// differentiation of $x_4 = x_1 x_2$
$\bar{x}_1 = \bar{x}_1 + x_2\bar{x}_4$
$\bar{x}_2 = \bar{x}_2 + x_1\bar{x}_4$

The desired output of the algorithm is $(\bar{x}_1, \bar{x}_2, \bar{x}_3)$, equal to the three components of the gradient $\nabla f(x)$. Note that all three are returned in *only one* reverse sweep.

It can be shown that the cost of Algorithm 6 is less than 3 times the cost of Algorithm 4, i.e.,

$$\text{cost}(\lambda^\top J_F) \leq 3\ \text{cost}(F). \tag{10.24}$$

If we want to obtain the full Jacobian of F, we need to call Algorithm 6 several times with the n_F seed vectors corresponding to the unit vectors in $\mathbb{R}^{n_F}$, i.e., we have

$$\text{cost}(J_F) \leq 3\, n_F\ \text{cost}(F). \tag{10.25}$$

This is a remarkable fact: it means that the backward mode of AD can compute the full Jacobian at a cost that is independent of the state dimension n. This is particularly advantageous if $n_F \ll n$, e.g., if we compute the gradient of a scalar function like the objective or the Lagrangian. The reverse mode can be much faster than what we can obtain by finite differences, where we always need $(n + 1)$ function evaluations. To give an example, if we want to compute the gradient of a scalar function $f : \mathbb{R}^n \to \mathbb{R}$ with n =1,000,000 and each call of the function needs one second of CPU time, then the finite difference approximation of the gradient would take 1,000,001 seconds, while the computation of the same quantity with the backward mode of AD needs only 4 seconds (1 call of the function plus one backward sweep). Thus, besides being more accurate, backward AD can also be much faster than finite differences.
The only disadvantage of the backward mode of AD is that we have to store all intermediate variables and partial derivatives, in contrast to finite differences or forward AD. A partial remedy to this problem exists in the form of *checkpointing* that trades off computational speed and memory requirements. Instead of all intermediate variables, it only stores some "checkpoints" during the forward evaluation. During the backward sweep, starting at these checkpoints, it re-evaluates parts of the function to obtain those intermediate variables that have not been stored. The optimal number and location of checkpoints is a science of itself. Generally speaking, checkpointing reduces the memory requirements but comes at the expense of runtime.
From a user perspective, the details of implementation are not too relevant, but it is most important to just know that the reverse mode of AD exists and that it allows in many cases a much more efficient derivative generation than any other technique.

Efficient Computation of the Hessian

A particularly important quantity in Newton type optimization methods is the Hessian of the Lagrangian. It is the second derivative of the scalar function $\mathcal{L}(x, \lambda, \mu)$ with respect to x. As the multipliers are fixed for the purpose of differentiation, we can for notational simplicity just regard a function $f : \mathbb{R}^n \to \mathbb{R}$ of which we want to compute the Hessian $\nabla^2 f(x)$. With finite differences, we would at least need $n(n + 1)/2$ function evaluations in order to compute the Hessian, and due to round-off and truncation errors, the accuracy of a finite difference Hessian would be much lower than the accuracy of the function f. In contrast to this, algorithmic differentiation can without problems be applied recursively, yielding a code that computes the Hessian matrix at the same precision as the function f itself, i.e., typically at machine precision. Moreover, if we use the reverse mode of AD at least once, e.g., by first generating an efficient code for $\nabla f(x)$ (using backward AD) and then using forward AD to obtain the Jacobian of it, we can reduce the CPU time considerably compared to finite differences. Using the above procedure, we would obtain the Hessian $\nabla^2 f$ at a cost of $2\, n$ times the cost of a gradient ∇f, which is about four times the cost of evaluating f alone. This means that we have the following runtime bound:

$$\text{cost}(\nabla^2 f) \leq 8\, n\ \text{cost}(f). \tag{10.26}$$

A compromise between accuracy and ease of implementation that is equally fast in terms of CPU time is to use backward AD only for computing the first-order derivative $\nabla f(x)$, and then to use finite differences for the differentiation of $\nabla f(x)$.

10.4 Algorithmic Differentiation Software

Most algorithmic differentiation tools implement both forward and backward AD, and most are specific to one particular programming language. They come in two different variants: either they use *operator overloading* or *source-code transformation*.
The first class does not modify the code but changes the type of the variables and overloads the involved elementary operations. For the forward mode, each variable just gets an additional dot-quantity, i.e., the new variables are the pairs $(x_i, \dot{x_i})$, and elementary operations just operate on these pairs, like e.g.

$$(x, \dot{x}) \cdot (y, \dot{y}) = (xy, x\dot{y} + y\dot{x}). \tag{10.27}$$

An interesting remark is that operator overloading is also at the basis of the imaginary trick where we use the overloading of real numbers by complex numbers and use the small imaginary part as a dot quantity and exploit the fact that the extremely small higher-order terms disappear by numerical cancellation.
A prominent and widely used AD tool for generic user-supplied C++ code that uses operator overloading is ADOL-C. Though it is not the most efficient AD tool in terms of CPU time, it is well-documented and stable. Another popular tool in this class is CppAD.
The other class of AD tools is based on source-code transformation. They work like a text-processing tool that gets as input the user-supplied source code and produces as output a new and very differently looking source code that implements the derivative generation. Often, these codes can be made extremely fast. Tools that implement source code transformations are ADIC for ANSI C, and ADIFOR and TAPENADE for FORTRAN codes.
In the context of simulation of ordinary differential equations (ODE), there exist good numerical integrators with forward and backward differentiation capabilities that are more efficient and reliable than a naive procedure that would consist of taking an integrator and processing it with an AD tool. Examples for integrators that use the principle of forward and backward AD are the code DAESOL-II or the open-source codes from the ACADO Integrators Collection or from the SUNDIALS Suite.
Another interesting AD tool, CasADi, is an optimization modeling language that implements all variants of AD and provides several interfaces to ODE solvers with forward and derivative computations, as well as to optimization codes. It can conveniently be used from a Python front end.

Part III

Constrained Optimization: Optimality Conditions and Algorithms

Chapter 11

Optimality Conditions for Equality Constrained Problems

In this part, we regard an equality constrained minimization problem of the form

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \tag{11.1a}$$

$$\text{subject to} \quad g(x) = 0, \tag{11.1b}$$

in which $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are smooth. The feasible set for this problem is $\Omega = \{x \in \mathbb{R}^n \mid g(x) = 0\}$ and can be considered as a differentiable manifold. Differentiable manifolds can be complicated objects that are treated in much more detail than here in courses on differential geometry. However, we give a few relevant concepts from this field in order to formulate the optimality conditions for constrained optimization.

Definition 11.1: Tangent Vector
$p \in \mathbb{R}^n$ is called a "tangent vector" to Ω at $x^\star \in \Omega$ if there exists a smooth curve $\bar{x}(t) : [0, \varepsilon) \to \mathbb{R}^n$ with $\bar{x}(0) = x^\star$, $\bar{x}(t) \in \Omega \; \forall t \in [0, \varepsilon)$, and $\frac{d\bar{x}}{dt}(0) = p$.

Definition 11.2: Tangent Cone
The "tangent cone" $T_\Omega(x^\star)$ of Ω at $x^\star$ is the set of all tangent vectors at $x^\star$.

When we have only equality constraints and they are "well behaved" (as we define below), then the set of all tangent vectors at a point $x^\star \in \Omega$ forms a vector space, so the name "tangent space" would be appropriate. On the other hand, every space is also a cone, and when the equality constraints are not well behaved or we have inequality constraints, then the set of all tangent vectors forms only a cone. Thus, the name "tangent cone" is the appropriate name in the field of optimization.

Example 11.1: Tangent Cone
Regard $\Omega = \left\{x \in \mathbb{R}^2 \;\middle|\; x_1^2 + x_2^2 - 1 = 0\right\}$. In this example, we can generate the tangent cone by hand, making a sketch. Or we can use the fact that any feasible point $x^\star$ can be represented as $x^\star = [\cos(\alpha^\star), \sin(\alpha^\star)]^\top$. Then, feasible curves emanating from $x^\star$ have the form $\bar{x}(t) = [\cos(\alpha^\star + \omega t), \sin(\alpha^\star + \omega t)]^\top$, and their tangent vectors p are given by $p = \omega[-\sin(\alpha^\star), \cos(\alpha^\star)]^\top$. By choosing $\omega \in \mathbb{R}$, one can generate any vector in a one-dimensional vector space spanned by $[-\sin(\alpha^\star), \cos(\alpha^\star)]^\top$, and we have $T_\Omega(x^\star) = \left\{\omega[-\sin(\alpha^\star), \cos(\alpha^\star)]^\top \;\middle|\; \omega \in \mathbb{R}\right\}$. Note that for this example, the tangent vectors are orthogonal to the gradient of the constraint function $g(x) = x_1^2 + x_2^2 - 1$ at $x^\star$, because $\nabla g(x^\star) = 2[\cos(\alpha^\star), \sin(\alpha^\star)]^\top$.

We will see that the tangent cone can often directly be obtained by a linearization of the nonlinear inequalities. This is, however, only possible under some condition, which we will call "constraint qualification". Before we define it in more detail, let us see for what aim serves the tangent cone.

Theorem 11.1: First Order Necessary Conditions, Variant 1
If $x^\star$ is a local minimum of the NLP (11.1), then

1. $x^\star \in \Omega$
2. for all tangents $p \in T_\Omega(x^\star)$ holds: $\nabla f(x^\star)^\top p \geq 0$

Proof by contradiction. If $\exists p \in T_\Omega(x^\star)$ with $\nabla f(x^\star)^\top p < 0$, there would exist a feasible curve $\bar{x}(t)$ with $\frac{df(\bar{x}(t))}{dt}\big|_{t=0} = \nabla f(x^\star)^\top p < 0$. □

11.1 Constraint Qualification and Linearized Feasible Cone

How can we characterize $T_\Omega(x^\star)$?

Definition 11.3: LICQ
The "linear independence constraint qualification" (LICQ) holds at $x^\star \in \Omega$ if the vectors $\nabla g_i(x^\star)$ for $i \in \{1, \ldots, m\}$ are linearly independent.

Because the constraint Jacobian $\nabla g(x^\star)^\top$ collects all the above single gradients in its rows, LICQ is equivalent to stating that $\operatorname{rank}(\nabla g(x^\star)) = m$.

Definition 11.4: Linearized Feasible Cone for Equality Constraints
$\mathcal{F}(x^\star) = \left\{p \;\middle|\; \nabla g_i(x^\star)^\top p = 0,\ i = 1, \ldots, m\right\}$ is called the "linearized feasible cone" at $x^\star \in \Omega$.

Example 11.2: Linearized feasible cone for Example 11.1

$$g(x) = \begin{bmatrix} x_1^2 + x_2^2 - 1 \end{bmatrix} \tag{11.2a}$$

$$x^\star = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{11.2b}$$

$$\nabla g(x^\star) = \begin{bmatrix} 2x_1^\star \\ 2x_2^\star \end{bmatrix} \tag{11.2c}$$

$$= \begin{bmatrix} 0 \\ 2 \end{bmatrix} \tag{11.2d}$$

$$\mathcal{F}(x^\star) = \left\{ p \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} 0 \\ 2 \end{bmatrix}^\top p = 0 \right\}. \tag{11.2e}$$

It can be proven that the linearized feasible cone and the tangent cone coincide for this example.

Example 11.3: Linearized feasible cone can be larger than tangent cone

$$g(x) = \begin{bmatrix} x_1^3 - x_2 \\ \|x_1\|^3 - x_2 \end{bmatrix} \tag{11.3a}$$

$$x^\star = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{11.3b}$$

$$\nabla g(x^\star) = \begin{bmatrix} 3(x_1^\star)^2 & 3(x_1^\star)^2 \operatorname{sign}(x_2^\star) \\ -1 & -1 \end{bmatrix} \tag{11.3c}$$

$$= \begin{bmatrix} 0 & 0 \\ -1 & -1 \end{bmatrix} \tag{11.3d}$$

$$\mathcal{F}(x^\star) = \left\{ p \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} 0 & 0 \\ -1 & -1 \end{bmatrix}^\top p = 0 \right\} \tag{11.3e}$$

$$= \left\{ \begin{bmatrix} p_1 \\ 0 \end{bmatrix} \;\middle|\; p_1 \in \mathbb{R} \right\}, \quad \text{but} \tag{11.3f}$$

$$T_\Omega(x^\star) = \left\{ \begin{bmatrix} p_1 \\ 0 \end{bmatrix} \;\middle|\; p_1 \geq 0 \right\}. \tag{11.3g}$$

The feasible curves emanating from $x^\star$ for this example are given by $\bar{x}(t) = [t^3, t]^\top$ or time-scaled versions of it, but are only feasible for positive t. Note that LICQ does not hold for this example.

Theorem 11.2
At any $x^\star \in \Omega$ holds:

1. $T_\Omega(x^\star) \subset \mathcal{F}(x^\star)$.
2. If LICQ holds at $x^\star$, then $T_\Omega(x^\star) = \mathcal{F}(x^\star)$.

We prove the two parts of the theorem one after the other.

Proof of 1. We have to show that a vector p in the tangent cone is also in the linearized feasible cone.

$$p \in T_\Omega \implies \exists \bar{x}(t) \text{ with } p = \frac{d\bar{x}}{dt}\Big|_{t=0} \text{ and } \bar{x}(0) = x^\star \text{ and } \bar{x}(t) \in \Omega \tag{11.4}$$

$$\implies g(\bar{x}(t)) = 0 \quad \forall t \in [0, \varepsilon) \tag{11.5a}$$

$$\implies \frac{dg_i(\bar{x}(t))}{dt}\Big|_{t=0} = \nabla g_i(x^\star)^\top p = 0,\ i = 1, \dots, m, \tag{11.5b}$$

$$\implies p \in \mathcal{F}(x^\star). \tag{11.5c}$$

Proof of 2. In order to show equality if LICQ holds, we have to show that every vector p in the linearized feasible cone is also a tangent vector. The idea is to construct a curve $\bar{x}(t)$ which has the given vector $p \in \mathcal{F}(x^\star)$ as tangent by using the implicit function theorem. Let us first introduce a shorthand for the Jacobian $J \coloneqq \nabla g(x^\star)^\top$, and regard the singular value decomposition of it.

$$\nabla g(x^\star)^\top = J = USV^\top = U \begin{bmatrix} S_+ & 0 \end{bmatrix} \begin{bmatrix} Y^\top \\ Z^\top \end{bmatrix} = US_+Y^\top. \tag{11.6}$$

Here, $V = \begin{bmatrix} Y & Z \end{bmatrix}$ is an orthonormal matrix and the left block S_+ of $S = \begin{bmatrix} S_+ & 0 \end{bmatrix}$ is diagonal with strictly positive elements on its diagonal, due to the LICQ assumption. Now, Z is an orthonormal basis of the nullspace of J, which is equal to the linearized feasible cone. Thus, we have $\mathcal{F}(x^\star) = \left\{ Zv \;\middle|\; v \in \mathbb{R}^{(n-m)} \right\}$. We will soon need a little lemma that is easy to prove:

Lemma 11.1
$p \in \mathcal{F}(x^\star) \implies p = ZZ^\top p$.

Let us now construct a feasible curve $\bar{x}(t)$ with $\frac{\mathrm{d}\bar{x}}{\mathrm{d}t}(0) = p$ by using an implicit function representation $F(\bar{x}(t), t) = 0$. For this aim, let us define the following function:

$$F(x, t) = \begin{bmatrix} g(x) \\ Z^\top(x - (x^\star + tp)) \end{bmatrix}. \tag{11.7}$$

Check that it satisfies the necessary properties to apply the implicit function theorem. First, it is easy to check that $F(x^\star, 0) = 0$. Second, the Jacobian is given by:

$$\frac{\partial F}{\partial x}(x^\star, 0) = \begin{bmatrix} J \\ Z^\top \end{bmatrix} = \begin{bmatrix} US_+Y^\top \\ Z^\top \end{bmatrix} = \begin{bmatrix} US_+ & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} Y^\top \\ Z^\top \end{bmatrix}. \tag{11.8}$$

This matrix is invertible, as the product of two invertible matrices. Thus, we can apply the implicit function theorem. To compute the derivative at $t = 0$, we use:

$$\frac{\mathrm{d}\bar{x}}{\mathrm{d}t}(0) = -\frac{\partial F}{\partial x}(x^\star, 0)^{-1} \frac{\partial F}{\partial t}(x^\star, 0) = -\begin{bmatrix} Y & Z \end{bmatrix} \begin{bmatrix} S_+^{-1}U^\top & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 \\ -Z^\top p \end{bmatrix} = ZZ^\top p = p. \tag{11.9}$$

□

Based on the fact that LICQ implies equality of the linearized feasible cone and the tangent cone, we can state a second variant of the theorem. It uses $\mathcal{F}(x^\star)$, and due to the fact that it is a linear space that admits both p and $-p$ as elements, the gradient must be equal to zero on the nullspace of the constraint Jacobian.

Theorem 11.3: FONC, Variant 2
If LICQ holds at $x^\star$ and $x^\star$ is a local minimizer for the NLP (11.1), then:

1. $x^\star \in \Omega$.
2. $\forall p \in \mathcal{F}(x^\star) : \nabla f(x^\star)^\top p = 0$.

We can make the statements a bit more explicit by using the fact that each $p \in \mathcal{F}(x^\star)$ can be written as $p = Zv$ with some $v \in \mathbb{R}^{(n-m)}$, rephrasing the theorem as follows.

Theorem 11.4: FONC, Variant 3
If LICQ holds at $x^\star$ and $x^\star$ is a local minimizer for the NLP (11.1), then:

1. $g(x^\star) = 0$.
2. $Z^\top \nabla f(x^\star) = 0$.

How can we further simplify the second condition? Here helps a decomposition of the gradient $\nabla f(x^\star)$ into its components in the orthogonal subspaces spanned by Y and Z, as follows:

$$\nabla f(x^\star) = YY^\top \nabla f(x^\star) + ZZ^\top \nabla f(x^\star). \tag{11.10}$$

Now, $Z^\top \nabla f(x^\star) = 0$ is equivalent to saying that there exists some u (namely $u = Y^\top \nabla f(x^\star)$) such that $\nabla f(x^\star) = Yu$, or, equivalently, using the fact that $\nabla g(x^\star) = YS_+U^\top$ spans the same subspace as Y, that there exists some $\lambda^\star$ (namely $\lambda^\star = US_+^{-1}Y^\top \nabla f(x^\star) = \nabla g(x^\star)^+ \nabla f(x^\star)$) such that $\nabla f(x^\star) = \nabla g(x^\star)\lambda^\star$.

Theorem 11.5: FONC, Variant 4
If LICQ holds at $x^\star$ and $x^\star$ is a local minimizer for the NLP (11.1), then:

1. $g(x^\star) = 0$.
2. There exists $\lambda^\star \in \mathbb{R}^m$ such that $\nabla f(x^\star) = \nabla g(x^\star)\lambda^\star$.

This is a remarkable formulation because it allows us to search for a pair of $x^\star$ and $\lambda^\star$ together, e.g., via a Newton type root-finding method.

11.2 Second Order Conditions

Theorem 11.6: Second Order Necessary Conditions, SONC
If $x^\star$ satisfies LICQ and is a local minimizer of the NLP, then:

i. $\exists \lambda^\star$ such that the FONC hold;

ii. $\forall p \in \mathcal{F}(x^\star)$, it holds that $p^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star) p \geq 0$.

Theorem 11.7: Second Order Sufficient Conditions, SOSC
If $x^\star$ satisfies LICQ and:

i. $\exists \lambda^\star$ such that the FONC hold;

ii. $\forall p \in \mathcal{F}(x^\star),\ p \neq 0$, it holds that $p^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star) p > 0$,

then $x^\star$ is a strict local minimizer.

Sketch of proof of both theorems Let us regard points in the feasible set Ω. For fixed $\lambda^\star$, we have for all $x \in \Omega$:

$$\mathcal{L}(x, \lambda^\star) = f(x) - \sum \lambda_i^\star \underbrace{g_i(x)}_{=0} = f(x). \tag{11.11}$$

Also: $\nabla_x \mathcal{L}(x^\star, \lambda^\star) = 0$. So for all $x \in \Omega$, we have:

$$f(x) = \mathcal{L}(x, \lambda^\star) = \underbrace{\mathcal{L}(x^\star, \lambda^\star)}_{=f(x^\star)} + \underbrace{\nabla_x \mathcal{L}(x^\star, \lambda^\star)^\top}_{=0}(x - x^\star) + \frac{1}{2}(x - x^\star)^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star)(x - x^\star) + o(\|x - x^\star\|^2) = f(x^\star)$$

(11.12a)

□

11.3 Perturbation Analysis

Does the solution also exist for perturbed problem data? How do the minimum point $x^\star$ and the optimal value depend on perturbation parameters? For this aim, we regard the solution $x^\star(p)$ of the following parametric optimization problem, with twice continuously differentiable functions f and g, which we denote by NLP(p):

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x, p) && \text{(11.13a)} \\ \text{subject to} \quad & g(x, p) = 0. && \text{(11.13b)} \end{aligned}$$

Theorem 11.8: Stability under Perturbations
Regard a solution $\bar{x}$ of NLP($\bar{p}$) that satisfies (LICQ) and (SOSC), i.e., there exist multipliers $\bar{\lambda}$ such that the gradient of the Lagrangian is zero (FONC) and the Hessian is positive definite on the null space of the constraint Jacobian. Then the solution maps $x^\star(p)$ and $\lambda^\star(p)$ exist for all p in a neighborhood of $\bar{p}$.

Proof. Regard the joint variable vector $w = (x^\top, \lambda^\top)^\top$ and the function:

$$F(w,p) \coloneqq \begin{bmatrix} \nabla_x \mathcal{L}(x,\lambda,p) \\ g(x,p) \end{bmatrix}. \tag{11.14}$$

First, $F(\bar{w},\bar{p}) = 0$ due to (FONC), and second:

$$\frac{\partial F}{\partial w}(\bar{w},\bar{p}) = \begin{bmatrix} \nabla_x^2 \mathcal{L}(\bar{x},\bar{\lambda},\bar{p}) & -\nabla_x g(\bar{x},\bar{p}) \\ \nabla_x g(\bar{x},\bar{p})^\top & 0 \end{bmatrix} \tag{11.15}$$

is invertible due to the following lemma.

Lemma 11.2: KKT-Matrix-Lemma
Regard a matrix, which we call the "KKT-matrix":

$$\begin{bmatrix} B & A^\top \\ A & 0 \end{bmatrix} \tag{11.16}$$

with some given $B \in \mathbb{R}^{n\times n}$, $B = B^\top$, $A \in \mathbb{R}^{m\times n}$ with $m \leq n$. If $\operatorname{rank}(A) = m$ (A is of full rank, i.e., LICQ holds) and for all $p \neq 0$ in the nullspace of A, it holds that $p^\top B p > 0$ (SOSC), then the KKT-matrix is invertible.

We leave the proof of the lemma as an exercise (alternatively, we refer to [4], Section 16.1). Using the lemma with $B = \nabla_x^2 \mathcal{L}(\bar{x},\bar{\lambda},\bar{p})$ and $A = \nabla_x g(\bar{x},\bar{p})^\top$, and noting that the minus sign in the upper right block can be removed by a sign change of the corresponding unknown, we have indeed shown invertibility of $\frac{\partial F}{\partial w}(\bar{w},\bar{p})$, such that the implicit function theorem can be applied and the theorem is proven. We remark that the derivative of $w^\star(p)$ can also be computed using the inverse of the KKT matrix, by $\frac{\mathrm{d}w^\star}{\mathrm{d}p}(p) = -\frac{\partial F}{\partial w}(w,p)^{-1}\frac{\partial F}{\partial p}(w,p)$.

Theorem 11.9: Sensitivity of Optimal Value
Regard a solution $\bar{x}$ of NLP($\bar{p}$) that satisfies (LICQ) and (SOSC). Then the optimal value $f(x^\star(p),p)$ will be differentiable in a neighborhood of $\bar{p}$ and its derivative is given by the partial derivative of the Lagrangian w.r.t. p:

$$\frac{\mathrm{d}f(x^\star(p),p)}{\mathrm{d}p} = \frac{\partial \mathcal{L}}{\partial p}(x^\star(p),\lambda^\star(p),p). \tag{11.17}$$

Proof. Existence of the solution maps $x^\star(p)$ and $\lambda^\star(p)$ follows from Theorem 11.8. To obtain the derivative, we note that for all feasible points $x^\star(p)$, the values of f and of $\mathcal{L}$ coincide, i.e., we have $f(x^\star(p),p) = \mathcal{L}(x^\star(p),\lambda^\star(p),p)$. For the derivative, we obtain:

$$\frac{\mathrm{d}f(x^\star(p),p)}{\mathrm{d}p} = \frac{\mathrm{d}\mathcal{L}(x^\star(p),\lambda^\star(p),p)}{\mathrm{d}p} = \underbrace{\frac{\partial \mathcal{L}}{\partial x}}_{=0}\frac{\mathrm{d}x^\star}{\mathrm{d}p} + \underbrace{\frac{\partial \mathcal{L}}{\partial \lambda}}_{=0}\frac{\mathrm{d}\lambda^\star}{\mathrm{d}p} + \frac{\partial \mathcal{L}}{\partial p} = \frac{\partial \mathcal{L}}{\partial p}(x^\star(p),\lambda^\star(p),p). \tag{11.18a}$$

Corollary 11.1: Multipliers as Shadow Prices
Regard the following optimization problem with perturbed equality constraints:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(11.19a)} \\ \text{subject to} \quad & g(x) - p = 0, && \text{(11.19b)} \end{aligned}$$

and a solution $\bar{x}$ with multipliers $\bar{\lambda}$ that satisfies (LICQ) and (SOSC) for $p = \bar{p}$. Then the optimal value $f(x^\star(p))$ will be differentiable in a neighborhood of $\bar{p}$ and its derivative is given by $\lambda^\star(p)$:

$$\frac{\mathrm{d}f(x^\star(p))}{\mathrm{d}p} = \lambda^\star(p)^\top. \tag{11.20}$$

Proof. We apply Theorem 11.9, but the Lagrangian is now given by $\mathcal{L}(x, \lambda) = f(x) - \lambda^\top g(x) + \lambda^\top p$, such that its partial derivative w.r.t. p is given by $\frac{\partial \mathcal{L}}{\partial p} = \lambda^\top$. □

Chapter 12

Equality Constrained Optimization Algorithms

In this chapter, the problem to

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \tag{12.1a}$$

$$\text{subject to} \quad g(x) = 0, \tag{12.1b}$$

with $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, where f and g are both smooth functions, will be further treated in detail.

12.1 Optimality Conditions

KKT Conditions

The necessary KKT optimality condition for

$$\mathcal{L}(x, \lambda) = f(x) - \lambda^\top g(x) \tag{12.2}$$

leads to the expression

$$\nabla \mathcal{L}(x^\star, \lambda^\star) = 0 \tag{12.3a}$$

$$g(x^\star) = 0 \tag{12.3b}$$

Keep in mind that this expression is only valid if we have LICQ, or equivalently stated, if the vectors $\nabla g_i(x^\star)$ are linearly independent. Recall the definition of the gradient

$$\nabla g(x) = \left(\frac{\partial g}{\partial x}(x) \right)^\top = (\nabla g_1(x), \nabla g_2(x), \ldots, \nabla g_m(x)) \tag{12.4}$$

The rank of the matrix $\nabla g(x^\star)$ must be m to obtain LICQ. The tangent space is defined as

$$T_\Omega(x^\star) = \left\{ p \mid \nabla g(x^\star)^\top p = 0 \right\} =: \text{Ker} \left(\nabla g(x^\star)^\top \right) \tag{12.5}$$

An explicit form of $\text{Ker}\left(\nabla g(x)^\top\right)$ can be obtained by a basis for this space $Z \in \mathbb{R}^{n \times (n-m)}$ such that the $\text{kernel}\left(\nabla g(x)^\top\right) = \text{image}(Z)$, i.e., $\nabla g(x)^\top Z = 0$ and $\text{rank}(Z) = n - m$. This basis $(Z_1 Z_2 \ldots Z_{n-m})$ can be obtained by using a QR-factorization of the matrix $\nabla g(x)$.

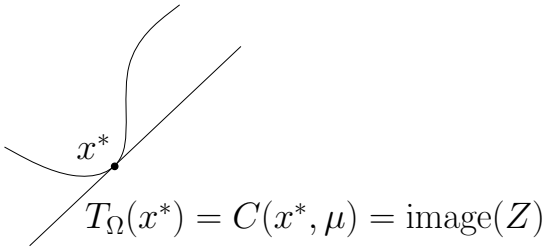


Figure 12.1: The critical cone equals the tangent cone when there are no inequality constraints.

SONC and SOSC

For equality constrained problems, SONC looks like

$$Z^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star) Z \succcurlyeq 0 \tag{12.6}$$

The SOSC points out that if

$$Z^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star) Z \succ 0 \tag{12.7}$$

and the LICQ and KKT conditions are satisfied, then $x^\star$ is a minimizer. The crucial role is played by the "reduced Hessian" $Z^\top \nabla_x^2 \mathcal{L} Z$.

12.2 Equality Constrained QP

Regard the optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} x^\top B x + g^\top x \tag{12.8a}$$

$$\text{subject to} \quad b + Ax = 0, \tag{12.8b}$$

with $B \in \mathbb{R}^{n \times n}$, $A \in \mathbb{R}^{m \times n}$, $B = B^\top$. The KKT condition leads to the equation

$$Bx + g - A^\top \lambda = 0 \tag{12.9a}$$

$$b + Ax = 0. \tag{12.9b}$$

In matrix notation:

$$\begin{bmatrix} B & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ -\lambda \end{bmatrix} = - \begin{bmatrix} g \\ b \end{bmatrix}. \tag{12.10}$$

Note that we made the second variable $-\lambda$ to get a symmetric KKT matrix.

Lemma 12.1: KKT-Matrix-Lemma
Let $B \in \mathbb{R}^{n \times n}$ be symmetric, and $A \in \mathbb{R}^{m \times n}$ with $m \leq n$. Assume $\text{rank}(A) = m$ (LICQ) and $p^\top B p > 0$ for all $p \in \text{Ker}(A) \setminus \{0\}$ (SOSC). Then, the KKT matrix $\begin{bmatrix} B & A^\top \\ A & 0 \end{bmatrix}$ is invertible.

For the proof, we refer to [4, Section 16.1].
Remark that for a QP, $B = \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star)$ and $A = \nabla g(x)^\top$, so that the above invertibility condition is equivalent to SOSC. Note also that the QP is convex under these conditions.

Solving the KKT System

Solving KKT systems is an important research topic. There exist many ways to solve the system (12.9). Some methods are:

i. Brute Force: obtain a dense LU-factorization of the KKT-matrix.

ii. As the KKT-matrix is not definite, a standard Cholesky decomposition does not work.Use an indefinite Cholesky decomposition.

iii. Schur complement method or so-called "Range Space method": first eliminate x, by equation

$$x = B^{-1}(A^\top \lambda - g) \tag{12.11}$$

and plug it into the second equation (12.9b). Get λ from

$$b + A(B^{-1}(A^\top \lambda - g)) = 0. \tag{12.12}$$

This method requires that B is invertible, which is not always true.

iv. Null Space Method: First find basis $Z \in \mathbb{R}^{n\times(n-m)}$ of $\mathrm{Ker}(A)$, set $x = Zv + y$ with $b + Ay = 0$ (a special solution). Every $x = Zv + y$ satisfies $b + Ax = 0$, so we have to regard only (12.9a). This is an unconstrained problem

$$\underset{v\,\in\,\mathbb{R}^{n-m}}{\text{minimize}} \quad g^\top(Zv + y) + \frac{1}{2}(Zv + y)^\top B(Zv + y). \tag{12.13}$$

One can solve this as follows:

$$\iff Z^\top BZv + Z^\top g + Z^\top By = 0 \tag{12.14a}$$
$$\iff v = -(Z^\top BZ)^{-1}(Z^\top g + Z^\top By). \tag{12.14b}$$

The matrix $Z^\top BZ$ is called "Reduced Hessian". This method is always possible if SOSC holds.

v. Sparse direct methods like sparse LU decomposition.

vi. Iterative methods of linear algebra.

12.3 Newton Lagrange Method

Regard again the optimization problem (12.1) as stated at the beginning of the chapter. The idea now is to apply Newton's method to solve the nonlinear KKT conditions:

$$\nabla_x \mathcal{L}(x, \lambda) = 0, \quad g(x) = 0. \tag{12.15a}$$

Define:

$$w \coloneqq \begin{bmatrix} x \\ \lambda \end{bmatrix}, \quad F(w) \coloneqq \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{bmatrix}, \tag{12.16}$$

with $w \in \mathbb{R}^{n+m}$, $F : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$, so that the optimization is just a nonlinear root-finding problem:

$$F(w) = 0, \tag{12.17}$$

which we solve again by Newton's method:

$$F(w_k) + \frac{\partial F}{\partial w_k}(w_k)(w - w_k) = 0. \tag{12.18}$$

Written in terms of gradients:

$$\nabla_x \mathcal{L}(x_k, \lambda_k) + \nabla_x^2 \mathcal{L}(x, \lambda)(x - x_k) - \nabla g(x_k)(\lambda - \lambda_k) = 0. \tag{12.19}$$

Here, $\nabla_x^2 \mathcal{L}(x, \lambda)(x - x_k)$ is the linearization with respect to x, and $\nabla g(x_k)(\lambda - \lambda_k)$ is the linearization with respect to λ. Recall that $\nabla \mathcal{L} = \nabla f - \nabla g \lambda$.

$$g(x_k) + \nabla g(x_k)^\top (x - x_k) = 0. \tag{12.20}$$

Written in matrix form, an interesting result is obtained:

$$\begin{bmatrix} \nabla_x \mathcal{L} \\ g \end{bmatrix} + \underbrace{\begin{bmatrix} \nabla_x^2 \mathcal{L} & \nabla g \\ \nabla g^\top & 0 \end{bmatrix}}_{\text{KKT-matrix}} \begin{bmatrix} x - x_k \\ -(\lambda - \lambda_k) \end{bmatrix} = 0. \tag{12.21}$$

The KKT-matrix is invertible if the KKT-matrix lemma holds. From this point, it is clear that at a given solution $(x^\star, \lambda^\star)$ with LICQ and SOSC, the KKT-matrix would be invertible. This also holds in the neighborhood of $(x^\star, \lambda^\star)$. Thus, if $(x^\star, \lambda^\star)$ satisfies LICQ and SOSC, then the Newton method is well-defined for all (x_0, λ_0) in the neighborhood of $(x^\star, \lambda^\star)$ and converges Q-quadratically. The method is stated as an algorithm in Algorithm 7. Using the definition:

Algorithm 7 Equality constrained Newton Lagrange method

Choose: $x_0, \lambda_0, \varepsilon$
Set: $k = 0$
while $\left\| \begin{bmatrix} \nabla \mathcal{L}(x_k, \lambda_k) \\ g(x_k) \end{bmatrix} \right\| \geq \varepsilon$ **do**
  Get Δx_k and $\Delta \lambda_k$ from (12.23)
  $x_{k+1} = x_k + \Delta x_k$
  $\lambda_{k+1} = \lambda_k + \Delta \lambda_k$
  $k = k + 1$
end while

$$\lambda_{k+1} = \lambda_k + \Delta \lambda_k, \quad \nabla \mathcal{L}(x_k, \lambda_k) = \nabla f(x_k) - \nabla g(x_k) \lambda_k, \tag{12.22a}$$

the system (12.21) needed for calculating the new dual value λ_{k+1} and the primal step Δx_k together is equivalent to:

$$\begin{bmatrix} \nabla f(x_k) \\ g(x_k) \end{bmatrix} + \begin{bmatrix} \nabla_x^2 \mathcal{L}(x_k, \lambda_k) & \nabla g(x_k) \\ \nabla g(x_k)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ -\lambda_{k+1} \end{bmatrix} = 0. \tag{12.23}$$

Note that due to the trick to use the value of the new multiplier rather than the step as the variable of the linear system, we were able to replace the Lagrange gradient by the objective gradient. This formulation shows that the new iterate does not depend strongly on the old multiplier guess λ_k, which only affects the (exact) Hessian matrix. We will later see that we can approximate the Hessian with different methods, some of which are completely independent of the multipliers.

12.4 Quadratic Model Interpretation

The Newton Lagrange method from the previous section can be interpreted as a method that solves a quadratic program (QP) in each iteration.

Theorem 12.1
x_{k+1} and λ_{k+1} are obtained from the solution of a QP:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x_k)^\top (x - x_k) + \frac{1}{2}(x - x_k)^\top \nabla_x^2 \mathcal{L}(x_k, \lambda_k)(x - x_k) && \text{(12.24a)} \\ \text{subject to} \quad & g(x_k) + \nabla g(x_k)^\top (x - x_k) = 0. && \text{(12.24b)} \end{aligned}$$

So we can get a QP solution x^{QP} and λ^{QP} and take it as the next NLP solution guess x_{k+1} and λ_{k+1}.

Proof. KKT of QP:

$$\nabla f(x_k) + \nabla^2 \mathcal{L}(x_k, \lambda_k)(x^{\text{QP}} - x_k) - \nabla g(x_k)\lambda^{\text{QP}} = 0, \quad g + \nabla g^\top (x^{\text{QP}} - x_k) = 0. \tag{12.25}$$

□

More generally, one can replace $\nabla_x^2 \mathcal{L}(x_k, \lambda_k)$ by some approximation B_k, ($B_k = B_k^\top$, often $B_k \succcurlyeq 0$) by Quasi-Newton updates or other.

12.5 Constrained Gauss-Newton

Regard:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & \frac{1}{2} \|F(x)\|_2^2 && \text{(12.26a)} \\ \text{subject to} \quad & g(x) = 0. && \text{(12.26b)} \end{aligned}$$

As in the unconstrained case, linearize both F and g. Get approximation by:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & \frac{1}{2} \|F(x_k) + J(x_k)(x - x_k)\|_2^2 && \text{(12.27a)} \\ \text{subject to} \quad & g(x_k) + \nabla g(x_k)^\top (x - x_k) = 0. && \text{(12.27b)} \end{aligned}$$

This is a LS-QP which is convex. We call this the constrained Gauss-Newton method. This approach gets a new iterate x_{k+1} by solving (12.27) in each iteration. Note that no multipliers λ_{k+1} are needed. The KKT conditions of LS-QP:

$$\nabla_x \frac{1}{2} \|F + J(x - x_k)\|_2^2 = J^\top J(x - x_k) + J^\top F, \tag{12.28}$$

equals:

$$J^\top J(x - x_k) + J^\top F - \nabla g \lambda = 0, \quad g + \nabla g^\top (x - x_k) = 0. \tag{12.29a}$$

Recall that $J^\top J$ is the same as by Newton iteration, but we replace the Hessian. The constrained Gauss-Newton gives a Newton type iteration with $B_k = J^\top J$. For LS:

$$\nabla_x^2 \mathcal{L}(x, \lambda) = J(x)^\top J(x) + \sum F_i(x) \nabla^2 F_i(x) - \sum \lambda_i \nabla^2 g_i(x). \tag{12.30}$$

One can show that $\|\lambda\|$ gets small if $\|F\|$ is small.

12.6 BFGS Method (for equality constrained optimization)

Regard the equality constrained BFGS method, as stated in Algorithm 8.

Algorithm 8 Equality constrained BFGS method

Choose x_0, B_0, tolerance
$k = 0$
Evaluate $\nabla f(x_0)$, $g(x_0)$, $\frac{\partial g}{\partial x}(x_0)$
while $\|g(x_k)\| >$ tolerance or $\left\|\nabla \mathcal{L}(x_k, \widetilde{\lambda}_k)\right\| >$ tolerance **do**
Solve KKT-system:
$$\begin{bmatrix} \nabla f \\ g \end{bmatrix} + \begin{bmatrix} B_k & \frac{\partial g}{\partial x}^\top \\ \frac{\partial g}{\partial x} & 0 \end{bmatrix} \begin{bmatrix} p_k \\ -\widetilde{\lambda}_k \end{bmatrix} = 0$$
Set $\Delta\lambda_k = \widetilde{\lambda}_k - \lambda_k$
Choose step length $t_k \in (0, 1]$ (details 11.7)
$x_{k+1} = x_k + t_k p_k$
$\lambda_{k+1} = \lambda_k + t_k \Delta\lambda_k$
Compute old Lagrange gradient:
$\nabla_x \mathcal{L}(x_k, \lambda_{k+1}) = \nabla f(x_k) - \frac{\partial g}{\partial x}(x_k)^\top \lambda_{k+1}$
Evaluate $\nabla f(x_{k+1})$, $g(x_{k+1})$, $\frac{\partial g}{\partial x}(x_{k+1})$
Compute new Lagrange gradient $\nabla_x \mathcal{L}(x_{k+1}, \lambda_{k+1})$
Set $s_k = x_{k+1} - x_k$
Set $y_k = \nabla_x \mathcal{L}(x_{k+1}, \lambda_{k+1}) - \nabla_x \mathcal{L}(x_k, \lambda_{k+1})$
Calculate B_{k+1} (e.g., with a BFGS update) using s_k and y_k
$k = k + 1$
end while

Remark: B_{k+1} can alternatively be obtained by either calculating the exact Hessian $\nabla^2 \mathcal{L}(x_{k+1}, \lambda_{k+1})$ or by calculating the Gauss-Newton Hessian $(J(x_{k+1})^\top J(x_{k+1})$ for a LS objective function).

12.7 Local Convergence

Theorem 12.2: Newton type convergence
Regard the root-finding problem

$$F(x) = 0, \quad F : \mathbb{R}^n \to \mathbb{R}^n \tag{12.31}$$

with $x^\star$ satisfying $F(x^\star) = 0$ as a local solution, $J(x) = \frac{\partial F}{\partial x}(x)$, and iteration $x_{k+1} = x_k - M_k^{-1}F(x_k)$ with $\forall k : M_k \in \mathbb{R}^{n\times n}$ invertible, and a Lipschitz condition

$$\left\|M_k^{-1}(J(x_k) - J(x^\star))\right\| \leq \omega \left\|x_k - x^\star\right\| \tag{12.32}$$

and a compatibility condition with $\kappa < 1$:

$$\left\|M_k^{-1}(J(x_k) - M_k)\right\| \leq \kappa_k < \kappa, \quad \|x_0 - x^\star\| \leq \frac{2}{\omega}(1-\kappa), \tag{12.33}$$

then $x_k \to x^\star$ with linear rate or even quadratic rate if $\kappa = 0$, or superlinear rate if $\kappa_k \to 0$ (proof as before).

Corollary 12.1: Convergence rates for constrained optimization
Newton type methods for constrained optimization locally converge:

- quadratically if $B_k = \nabla^2\mathcal{L}(x_k, \lambda_k)$ (i.e., exact Hessian),
- superlinearly if $B_k \to \nabla^2\mathcal{L}(x_k, \lambda_k)$ (e.g., BFGS),
- linearly if $\left\|B_k - \nabla^2\mathcal{L}(x_k, \lambda_k)\right\|$ is not too big (e.g., Gauss-Newton).

Proof.

$$J_k = \begin{bmatrix} \nabla^2\mathcal{L}(x_k, \lambda_k) & -\frac{\partial g}{\partial x}(x_k)^\top \\ \frac{\partial g}{\partial x}(x_k) & 0 \end{bmatrix}, \quad M_k = \begin{bmatrix} B_k & -\frac{\partial g}{\partial x}(x_k)^\top \\ \frac{\partial g}{\partial x}(x_k) & 0 \end{bmatrix}, \tag{12.34a}$$

$$J_k - M_k = \begin{bmatrix} \nabla^2\mathcal{L}(x_k, \lambda_k) - B_k & 0 \\ 0 & 0 \end{bmatrix}. \tag{12.34b}$$

□

Note that we could still ensure convergence even if the Jacobians $\frac{\partial g}{\partial x}$ were approximated. This could lead to potentially cheaper iterations as building and factoring the KKT matrix is the main cost per iteration. As in all Newton type methods, we only need to ensure that the residual $F(x)$ is exactly evaluated. The Lagrange gradient can be obtained by reverse automatic differentiation without ever evaluating $\frac{\partial g}{\partial x}$.

12.8 Globalization by Line Search

Idea: Use a "merit function" to measure progress in both *objective* and *constraints*.

Definition 12.1: L_1-merit function
The "L_1-merit function" is defined to be $T_1(x) = f(x) + \sigma \|g(x)\|_1$ with $\sigma > 0$.

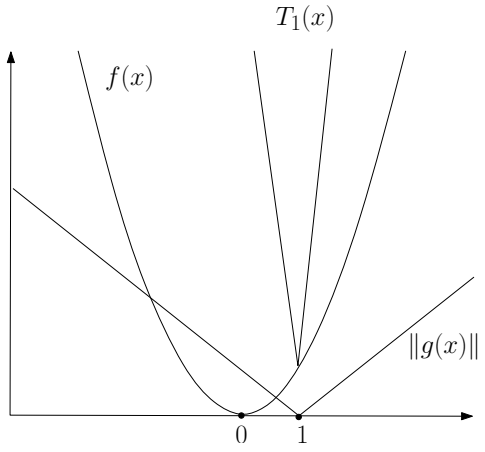


Figure 12.2: An example of an L_1 merit function with $f(x) = x^2$, $g(x) = x - 1$, and $\sigma = 10$.

Definition 12.2: Directional derivative
The "directional derivative of F at x in direction p" is $DF(x)[p] = \lim_{t\to 0, t>0} \frac{F(x+tp)-F(x)}{t}$.

Example 12.1: Directional derivative
Consider the the function $F(x) = |x - 1|$, which is not differentiable at $x = 1$. Then the directional derivatives at $x = 1$ in the directions 2 and -3 are:

$$DF(1)[2] = \lim_{t\to 0, t>0} \frac{|1 + t \cdot 2 - 1| - |1 - 1|}{t} = 2, \tag{12.35a}$$

$$DF(1)[-3] = \lim_{t\to 0, t>0} \frac{|1 + t \cdot (-3) - 1| - |1 - 1|}{t} = 3. \tag{12.35b}$$

Lemma 12.2
If p and $\widetilde{\lambda}$ solve

$$\begin{bmatrix} \nabla f \\ g \end{bmatrix} + \begin{bmatrix} B & \frac{\partial g}{\partial x}^\top \\ \frac{\partial g}{\partial x} & 0 \end{bmatrix} \begin{bmatrix} p \\ -\widetilde{\lambda} \end{bmatrix} = 0, \tag{12.36}$$

then

$$DT_1(x)[p] = \nabla f(x)^\top p - \sigma \|g(x)\|_1 \leq -p^\top B p - (\sigma - \left\|\widetilde{\lambda}\right\|_\infty) \|g(x)\|_1 . \tag{12.37}$$

Proof.

$$T_1(x+tp) = f(x+tp) + \sigma \left\| g(x+tp) \right\|_1 \tag{12.38a}$$
$$= f(x) + t\nabla f(x)^\top p + \sigma \left\| g(x) + \frac{\partial g}{\partial x}(x) pt \right\|_1 + \mathcal{O}(t^2) \tag{12.38b}$$
$$= f(x) + t\nabla f(x)^\top p + \sigma(1-t) \left\| g(x) \right\|_1 + \mathcal{O}(t^2) \tag{12.38c}$$
$$= T_1(x) + t(\nabla f(x)^\top p - \sigma \left\| g(x) \right\|_1) + \mathcal{O}(t^2), \tag{12.38d}$$

which implies the first inequality.

$$\nabla f(x) + Bp - \frac{\partial g}{\partial x}(x)^\top \widetilde{\lambda} = 0, \tag{12.39a}$$
$$\implies \nabla f(x)^\top p = \widetilde{\lambda}^\top \frac{\partial g}{\partial x}(x) p - p^\top B p = -\widetilde{\lambda}^\top g(x) - p^\top B p, \tag{12.39b}$$
$$\implies \left| \nabla f(x)^\top p \right| \leq \left\| \widetilde{\lambda} \right\|_\infty \left\| g(x) \right\|_1 - p^\top B p, \tag{12.39c}$$

which implies the second inequality. □

Corollary 12.2
If $B \succ 0$ and $\sigma \geq \left\| \widetilde{\lambda} \right\|_\infty$, then p is a descent direction of T_1.

In Algorithm 8, use Armijo backtracking with the L_1-merit function. One have to ensure that $\sigma \geq \left\| \widetilde{\lambda} \right\|_\infty$. If this is not the case, then one should increase σ.

12.9 Careful BFGS Updating

In this section, we recall the careful BFGS update method, which was already introduced earlier, in Method 7.6 (see Section 7.3 from Chapter 7). This method relies on Proposition 7.1, which answers the question: How can we make sure that B_k remains positive definite? We repeat this proposition here.

Proposition 12.1: Positive BFGS update
If $B_{k-1} \succ 0$ and $y_k^\top s_k > 0$, then B_k from the BFGS update is positive definite.

As noted before, in Section 7.3, this condition is tight, for any choice of B_k that satisfies the secant condition $B_k s_k = y_k$, if $y_k^\top s_k < 0$, then B_k is not positive semidefinite. This property is visualized in Figure 12.3.

Numerical Method 12.1: Careful BFGS updating with "Powell's trick"
If $y_k^\top s_k < 0.2 s_k^\top B s_k$, then do the update with a $\widetilde{y_k}$ instead of y_k with $\widetilde{y_k} = y_k + \theta(B_k s_k - y_k)$ so that $\widetilde{y_k}^\top s_k = 0.2 s_k^\top B_k s_k > 0$. The explicit formula for θ is easily seen to be

$$\theta = \begin{cases} \frac{0.2 s_k^\top B_k s_k - s_k^\top y_k}{s_k^\top B_k s_k - s_k^\top y_k} & \text{if } y_k^\top s_k < 0.2 s_k^\top B_k s_k, \\ 0 & \text{else.} \end{cases} \tag{12.40}$$

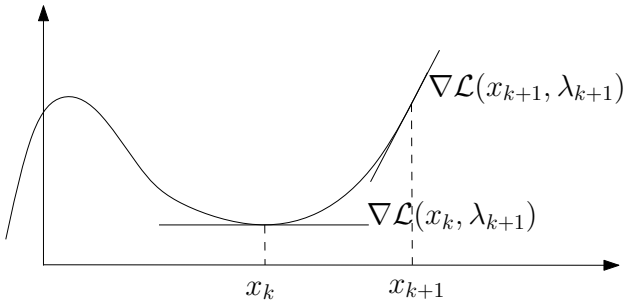


Figure 12.3: Visualization of Proposition 12.1. Remark that $y_k = \nabla\mathcal{L}(x_{k+1}, \lambda_{k+1}) - \nabla\mathcal{L}(x_k, \lambda_{k+1})$ and $s_k = x_{k+1} - x_k$.

Remark
If $\theta = 1$, then $\widetilde{y_k} = B_k s_k$ and $B_{k+1} = B_k$. Thus, the choice of θ between 0 and 1 damps the BFGS update.

Remark
Note that the new Hessian B_{k+1} will satisfy the modified secant condition $B_{k+1}s_k = \tilde{y}_k$, so we will have $s_k^\top B_{k+1} s_k = s_k^\top \tilde{y}_k > 0.2 s_k^\top B_k s_k$. The damping thus ensures that the positive curvature of the Hessian in direction s_k, which is expressed in the term $s_k^\top B_k s_k$, will never decrease by more than a factor of 5.

Chapter 13

Optimality Conditions for Constrained Optimization

From now on, we regard the general equality and inequality constrained minimization problem in standard form:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(13.1a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(13.1b)} \\ & h(x) \geq 0, && \text{(13.1c)} \end{aligned}$$

in which $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, and $h : \mathbb{R}^n \to \mathbb{R}^q$ are smooth. Recall that the feasible set for this problem is given by $\Omega = \big\{x \in \mathbb{R}^n \;\big|\; g(x) = 0,\ h(x) \geq 0\big\}$.

The tangent cone

The definition of tangent vector and tangent cone are still valid from before, as in Defs. 11.1 and 11.2.

Definition 13.1: Tangent
A direction $p \in \mathbb{R}^n$ is called a "tangent" to Ω at $x^\star \in \Omega$ if there exists a smooth curve $\bar{x}(t) : [0, \varepsilon) \to \mathbb{R}^n$ with $\bar{x}(0) = x^\star, \bar{x}(t) \in \Omega\ \forall t \in [0, \varepsilon)$ and $\frac{d\bar{x}}{dt}(0) = p$.

Definition 13.2: Tangent Cone
The "tangent cone" $T_\Omega(x^\star)$ of Ω at $x^\star$ is the set of all tangent vectors at $x^\star$.

Example 13.1: Tangent Cone
Regard $\Omega = \big\{x \in \mathbb{R}^2 \;\big|\; h(x) \geq 0\big\}$ with $h(x) = \begin{bmatrix} (x_1 - 1)^2 + x_2^2 - 1, \\ -(x_2 - 2)^2 - x_1^2 + 4 \end{bmatrix}$.

For $x^\star = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$, we have $T_\Omega(x^\star) = \left\{p \;\middle|\; p^\top \begin{bmatrix} 0 \\ -1 \end{bmatrix} \geq 0\right\} = \mathbb{R} \times \mathbb{R}_{--}$.

For $x^\star = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, we have $T_\Omega(x^\star) = \left\{p \;\middle|\; p^\top \begin{bmatrix} -1 \\ 0 \end{bmatrix} \geq 0,\ p^\top \begin{bmatrix} 0 \\ 1 \end{bmatrix} \geq 0\right\} = \big\{p \;\big|\; p_1 \leq 0,\ p_2 \geq 0\big\}$.

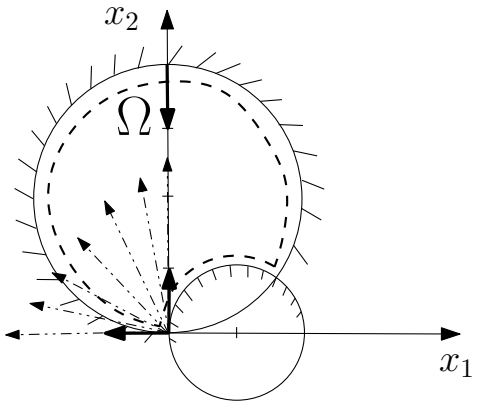


Figure 13.1: Visualization of Example 13.1.

In this example, we can generate the tangent cone by hand, making a sketch and afterwards defining the sets accordingly using suitable inequalities. But we will soon see a much more powerful way to generate the tangent cone directly by a linearization of the nonlinear inequalities. This is, however, only possible under some condition, which we will call "constraint qualification". Before, let us see for what aim serves the tangent cone.

13.1 First Order Necessary Condition for Constrained Problems

Theorem 13.1: First Order Necessary Condition, Variant 0
If $x^\star$ is a local minimizer of the NLP (13.1), then:

1. $x^\star \in \Omega$,
2. for all tangents $p \in T_\Omega(x^\star)$, it holds: $\nabla f(x^\star)^\top p \geq 0$.

Proof. By contradiction: ff $\exists p \in T_\Omega(x^\star)$ with $\nabla f(x^\star)^\top p < 0$, there would exist a feasible curve $\bar{x}(t)$ with $\frac{df(\bar{x}(t))}{dt}\big|_{t=0} = \nabla f(x^\star)^\top p < 0$, which would contradict the local optimality of $x^\star$. □

13.2 Active Constraints and Constraint Qualification

First, we introduce a couple of definitions, which we call "Constraint Qualifications". These are essentially regularity assumptions about the constraints. Broadly speaking, we want to assume that there are no redundant active constraints.

Vocabulary for constraint qualifications

Definition 13.3: Active/Inactive Constraint
An inequality constraint $h_i(x) \geq 0$ is called "active" at $x^\star \in \Omega$ iff $h_i(x^\star) = 0$, and otherwise "inactive".

Definition 13.4: Active Set
The index set $\mathcal{A}(x^\star) \subset \{1, \dots, q\}$ of active constraints is called the "active set".

Remark
Inactive constraints do not influence $T_\Omega(x^\star)$.

Definition 13.5: LICQ
The "linear independence constraint qualification" (LICQ) holds at $x^\star \in \Omega$ iff all vectors $\nabla g_i(x^\star)$ for $i \in \{1, \dots, m\}$ and $\nabla h_i(x^\star)$ for $i \in \mathcal{A}(x^\star)$ are linearly independent.

Remark
This is a technical condition and is usually satisfied.

Definition 13.6: Linearized Feasible Cone
$\mathcal{F}(x^\star) = \left\{p \mid \nabla g_i(x^\star)^\top p = 0,\ i = 1, \dots, m, \text{ and } \nabla h_i(x^\star)^\top p \geq 0,\ i \in \mathcal{A}(x^\star)\right\}$ is called the "linearized feasible cone" at $x^\star \in \Omega$.

Example 13.2: Linearized Feasible Cone
Consider the constraint $h(x) \geq 0$, with $h(\cdot)$ as in Example 13.1.
Then, for $x^\star = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$, we have $\mathcal{A}(x^\star) = \{2\}$ and $\nabla h_2(x^\star) = \begin{bmatrix} -2x_1^\star \\ -2(x_2^\star - 2) \end{bmatrix} = \begin{bmatrix} 0 \\ -4 \end{bmatrix}$. Therefore, $\mathcal{F}(x^\star) = \left\{p \mid \begin{bmatrix} 0 \\ -4 \end{bmatrix}^\top p \geq 0\right\} = \left\{p \mid p_2 \leq 0\right\}$, which is the same as $T_\Omega(x^\star)$.

$$h(x) =, \tag{13.2a}$$

$$x^\star = \begin{bmatrix} 0 \\ 4 \end{bmatrix}, \quad \mathcal{A}(x^\star) = \{2\}, \tag{13.2b}$$

$$\nabla h_2(x) = \begin{bmatrix} -2x_1 \\ -2(x_2 - 2) \end{bmatrix}, \tag{13.2c}$$

$$= \begin{bmatrix} 0 \\ -4 \end{bmatrix}, \tag{13.2d}$$

$$\mathcal{F}(x^\star) = \left\{p \mid \begin{bmatrix} 0 \\ -4 \end{bmatrix}^\top p \geq 0\right\}. \tag{13.2e}$$

Characterization of the tangent cone

In this part, we use the assumptions mentioned above to characterize the tangent cone $T_\Omega(x^\star)$.

Theorem 13.2
At any $x^\star \in \Omega$, it holds:

1. $T_\Omega(x^\star) \subset \mathcal{F}(x^\star)$,
2. If LICQ holds at $x^\star$, then $T_\Omega(x^\star) = \mathcal{F}(x^\star)$.

Sketch of proof.

$$p \in T_\Omega \implies \exists \bar{x}(t) \text{ with } p = \left.\frac{d\bar{x}}{dt}\right|_{t=0}, \text{and } \bar{x}(0) = x^\star, \text{and } \bar{x}(t) \in \Omega, \tag{13.3a}$$

$$\implies g(\bar{x}(t)) = 0, \quad \text{and} \quad h(\bar{x}(t)) \geq 0, \quad \forall t \in [0, \varepsilon), \tag{13.3b}$$

$$\implies \begin{cases} \left.\frac{dg_i(\bar{x}(t))}{dt}\right|_{t=0} = \nabla g_i(x^\star)^\top p = 0, \ i = 1, \ldots, m, \quad \text{and} \\ \left.\frac{dh_i(\bar{x}(t))}{dt}\right|_{t=0} = \lim_{t \to 0+} \frac{h_i(\bar{x}(t)) - h_i(x^\star)}{t} \geq 0, \quad \text{for } i \in \mathcal{A}(x^\star), \end{cases} \tag{13.3c}$$

$$\implies \left.\frac{dh_i(\bar{x}(t))}{dt}\right|_{t=0} = \nabla h_i(x^\star)^\top p \geq 0, \tag{13.3d}$$

$$\implies p \in \mathcal{F}(x^\star). \tag{13.3e}$$

□

For the full proof, see [Noc2006]. The idea is to use the implicit function theorem to construct a curve $\bar{x}(t)$ which has a given vector $p \in \mathcal{F}(x^\star)$ as tangent.

The Karush-Kuhn-Tucker (KKT) conditions

Theorem 13.3: FONC, Variant 1
If LICQ holds at $x^\star$ and $x^\star$ is a local minimizer for the NLP (13.1), then:

1. $x^\star \in \Omega$,
2. $\forall p \in \mathcal{F}(x^\star)$, it holds: $\nabla f(x^\star)^\top p \geq 0$.

To simplify the second condition, the following lemma is helpful. To interpret it, remember that $\mathcal{F}(x^\star) = \left\{p \mid Gp = 0, \ Hp \geq 0\right\}$, with $G = \frac{dg}{dx}(x^\star)$ and $H = \begin{bmatrix} \nabla h_i(x^\star)^\top \\ \vdots \end{bmatrix}$ for $i \in \mathcal{A}(x^\star)$.

Lemma 13.1: Farkas' Lemma
For any matrices $G \in \mathbb{R}^{m \times n}$, $H \in \mathbb{R}^{q \times n}$, and vector $c \in \mathbb{R}^n$, the following holds:

$$\text{either } \exists \lambda \in \mathbb{R}^m, \mu \in \mathbb{R}^q, \text{ with } \mu \geq 0, \text{ and } c = G^\top \lambda + H^\top \mu, \tag{13.4a}$$

$$\text{or } \exists p \in \mathbb{R}^n, \text{ with } Gp = 0, \text{ and } Hp \geq 0, \text{ and } c^\top p < 0, \tag{13.4b}$$

but never both ("theorem of alternatives").

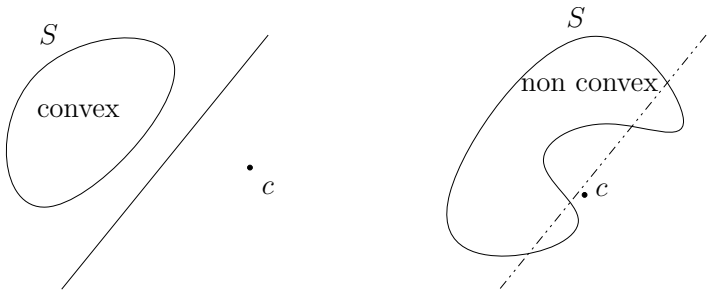
Proof. In the proof, we use the "separating hyperplane theorem" with respect to the point $c \in \mathbb{R}^n$ and the set $S = \left\{G^\top \lambda + H^\top \mu \;\middle|\; \lambda \in \mathbb{R}^m,\ \mu \in \mathbb{R}^q,\ \mu \geq 0\right\}$. S is a convex cone. The separating hyperplane theorem states that two disjoint convex sets—in our case, the set S and the point c—can always be separated by a hyperplane. In our case, the hyperplane touches the set S at the origin and is described by a normal vector p. Separation of S and c means that for all $y \in S$, $y^\top p \geq 0$, and on the other hand, $c^\top p < 0$.

$$\text{Either } c \in S \iff \text{(13.4a)}, \tag{13.5a}$$
$$\text{or } c \notin S, \tag{13.5b}$$
$$\iff \exists p \in \mathbb{R}^n : \forall y \in S : p^\top y \geq 0, \text{ and } p^\top c < 0, \tag{13.5c}$$
$$\iff \exists p \in \mathbb{R}^n : \forall \lambda, \mu, \text{ with } \mu \geq 0 : p^\top (G^\top \lambda + H^\top \mu) \geq 0, \text{ and } p^\top c < 0, \tag{13.5d}$$
$$\iff \exists p \in \mathbb{R}^n : Gp = 0, \text{ and } Hp \geq 0, \text{ and } p^\top c < 0 \iff \text{(13.4b)}. \tag{13.5e}$$

The last line follows because:

$$\forall \lambda, \mu, \text{ with } \mu \geq 0 : p^\top (G^\top \lambda + H^\top \mu) \geq 0, \tag{13.6a}$$
$$\iff \forall \lambda : \lambda^\top Gp \geq 0, \text{ and } \forall \mu \geq 0 : \mu^\top Hp \geq 0, \tag{13.6b}$$
$$\iff Gp = 0, \text{ and } Hp \geq 0. \tag{13.6c}$$

□



(a) convex (b) non-convex

Figure 13.2: Visualization of the separating hyperplane theorem, used in the proof of Lemma 13.1. For the non-convex case, no hyperplane can be found.

From Farkas' lemma follows the desired simplification of the previous theorem:

Theorem 13.4: FONC, Variant 2: KKT Conditions
If $x^\star$ is a local minimizer of the NLP (13.1) and LICQ holds at $x^\star$, then there exists a $\lambda^\star \in \mathbb{R}^m$ and $\mu^\star \in \mathbb{R}^q$ such that:

$$\nabla f(x^\star) - \nabla g(x^\star)\lambda^\star - \nabla h(x^\star)\mu^\star = 0, \tag{13.7a}$$
$$g(x^\star) = 0, \tag{13.7b}$$
$$h(x^\star) \geq 0, \tag{13.7c}$$
$$\mu^\star \geq 0, \tag{13.7d}$$
$$\mu_i^\star h_i(x^\star) = 0, \quad i = 1, \ldots, q. \tag{13.7e}$$

Note: The KKT conditions are the first-order necessary conditions for optimality (FONC) for constrained optimization and are thus equivalent to $\nabla f(x^\star) = 0$ in unconstrained optimization.

Proof. We already know that (13.7b), (13.7c) $\iff$ $x^\star \in \Omega$. We have to show that (13.7a), (13.7d), (13.7e) $\iff$ $\forall p \in \mathcal{F}(x^\star) : p^\top \nabla f(x^\star) \geq 0$. Using Farkas' lemma, we have:

$$\forall p \in \mathcal{F}(x^\star) : p^\top \nabla f(x^\star) \geq 0 \iff \text{It is not true that } \exists p \in \mathcal{F}(x^\star) : p^\top \nabla f(x^\star) < 0, \tag{13.8a}$$
$$\iff \exists \lambda^\star, \mu_i^\star \geq 0 : \nabla f(x^\star) = \sum \nabla g_i(x^\star)\lambda_i^\star + \sum_{i \in \mathcal{A}(x^\star)} \nabla h_i(x^\star)\mu_i^\star. \tag{13.8b}$$

Now we set all components of μ that are not elements of $\mathcal{A}(x^\star)$ to zero, i.e., $\mu_i = 0$ if $h_i(x^\star) > 0$, and conditions (13.7d) and (13.7e) are trivially satisfied, as well as (13.7a) due to $\sum_{i \in \mathcal{A}(x^\star)} \nabla h_i(x^\star)\mu_i^\star = \sum_{i=\{1,\ldots,q\}} \nabla h_i(x^\star)\mu_i$ if $\mu_i^\star = 0$ for $i \notin \mathcal{A}(x^\star)$. □

Though it is not necessary for the proof of the *necessity* of the optimality conditions of the above theorem (variant 2), we point out that the theorem is completely equivalent to variant 1 but has the computational advantage that its conditions can be checked easily: if someone gives you a triple $(x^\star, \lambda^\star, \mu^\star)$, you can check if it is a KKT point or not.
Note: Using the definition of the Lagrangian, we have (13.7a) $\iff$ $\nabla_x \mathcal{L}(x^\star, \lambda^\star, \mu^\star) = 0$. In the absence of inequalities, the KKT conditions simplify to $\nabla_x \mathcal{L}(x, \lambda) = 0$, $g(x) = 0$, a formulation that is due to Lagrange and was much earlier known than the KKT conditions.

Example 13.3: KKT Condition

$$\begin{aligned} \underset{x \in \mathbb{R}^2}{\text{minimize}} \quad & -x_2 && \text{(13.9a)} \\ \text{subject to} \quad & x_1^2 + x_2^2 - 1 \geq 0, && \text{(13.9b)} \\ & -(x_2 - 2)^2 - x_1^2 + 4 \geq 0. && \text{(13.9c)} \end{aligned}$$

Does the local minimizer $x^\star = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$ satisfy the KKT conditions?

First:

$$\mathcal{A}(x^\star) = \{2\}, \quad \nabla f(x^\star) = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad \nabla h_2(x^\star) = \begin{bmatrix} 0 \\ -4 \end{bmatrix}. \tag{13.10}$$

Then we write down the KKT conditions, which are for the specific dimensions of this example equivalent to the right-hand side terms:

$$\begin{aligned} &\text{(13.7a)} && \iff \nabla f(x^\star) - \nabla h_1(x^\star)\mu_1^\star - \nabla h_2(x^\star)\mu_2^\star = 0, && \text{(13.11a)} \\ &\text{(13.7b)} && \iff - && \text{(13.11b)} \\ &\text{(13.7c)} && \iff h_1(x^\star) \geq 0 \text{ and } h_2(x^\star) \geq 0, && \text{(13.11c)} \\ &\text{(13.7d)} && \iff \mu_1 \geq 0 \text{ and } \mu_2 \geq 0, && \text{(13.11d)} \\ &\text{(13.7e)} && \iff \mu_1 h_1(x^\star) = 0 \text{ and } \mu_2 h_2(x^\star) = 0. && \text{(13.11e)} \end{aligned}$$

Finally, we check that indeed all five conditions are satisfied if we choose $\mu_1^\star$ and $\mu_2^\star$ suitably:

$$\begin{aligned} &\text{(13.7a)} && \impliedby \begin{bmatrix} 0 \\ -1 \end{bmatrix} - \begin{bmatrix} * \\ * \end{bmatrix} \mu_1 - \begin{bmatrix} 0 \\ -4 \end{bmatrix} \mu_2 = 0, \quad (\mu_1 \text{ is inactive, use } \mu_1^\star = 0,\ \mu_2^\star = \frac{1}{4}), && \text{(13.12a)} \\ &\text{(13.7b)} && \impliedby -, && \text{(13.12b)} \\ &\text{(13.7c)} && \impliedby h_1(x^\star) > 0 \text{ and } h_2(x^\star) = 0, && \text{(13.12c)} \\ &\text{(13.7d)} && \impliedby \mu_1 = 0 \text{ and } \mu_2 = \frac{1}{4} \geq 0, && \text{(13.12d)} \\ &\text{(13.7e)} && \impliedby \mu_1 h_1(x^\star) = 0 \text{ and } \mu_2 h_2(x^\star) = \mu_2 \cdot 0 = 0. && \text{(13.12e)} \end{aligned}$$

13.3 KKT Conditions are Sufficient for Convex Problems

Now we present a theorem that states, similarly to the unconstrained case, that the first-order conditions are sufficient for convex problems. Note that here, an additional constraint qualification assumption is needed, i.e., the LICQ condition.

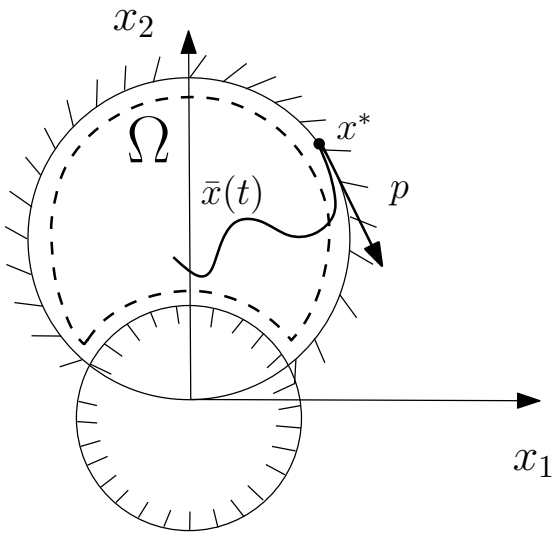


Figure 13.3: Visualization of Example 13.3.

Theorem 13.5: For convex problems, the KKT conditions are also necessary (if LICQP holds)
Regard a convex NLP and a point $x^\star$ at which LICQ holds. Then:

$$x^\star \text{ is a global minimizer } \iff \exists \lambda, \mu \text{ so that KKT conditions hold.}$$

Proof. We only need the "$\Leftarrow$"-direction.
Recall that a convex NLP takes the form:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(13.13a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(13.13b)} \\ & -h(x) \leq 0, && \text{(13.13c)} \end{aligned}$$

with f and all $-h_i$ convex, and g is affine, i.e., $g(x) = Gx + a$.

- Assume $(x^\star, \lambda^\star, \mu^\star)$ satisfies the KKT conditions.
- $\mathcal{L}(x, \lambda, \mu) = f(x) - \sum g_i(x)\lambda_i - \sum h_i(x)\mu_i$.
- $\mathcal{L}$ is a convex function of x, and for fixed $\mu^\star, \lambda^\star$, its gradient is zero, $\nabla\mathcal{L}(x^\star, \lambda^\star, \mu^\star) = 0$. Therefore, $x^\star$ is a global minimizer of the unconstrained convex minimization problem $\min_x \mathcal{L}(x, \lambda^\star, \mu^\star)$ with value $\mathcal{L}(x^\star, \lambda^\star, \mu^\star)$.
- We know, due to feasibility and complementarity, that:

$$\mathcal{L}(x^\star, \lambda^\star, \mu^\star) = f(x^\star) - \underbrace{\sum g_i(x^\star)\lambda_i^\star}_{=0} - \underbrace{\sum h_i(x^\star)\mu_i^\star}_{=0} = f(x^\star). \tag{13.14}$$

- We also know that the objective at the feasible point $x^\star$ can only be worse than the optimal primal objective value, i.e., $f(x^\star) \geq p^\star \coloneqq \min_{x\in\Omega} f(x)$, where Ω is the feasible set: $\Omega \coloneqq \left\{x \in \mathbb{R}^n \;\middle|\; g(x) = 0,\, h(x) \geq 0\right\}$. Furthermore:

$$\mathcal{L}(x^\star, \lambda^\star, \mu^\star) = q(\lambda^\star, \mu^\star) \leq \max_{\mu\geq 0, \lambda} q(\lambda, \mu) =: d^\star. \tag{13.15}$$

- This implies: $p^\star \leq f(x^\star) = \mathcal{L}(x^\star, \lambda^\star, \mu^\star) \leq d^\star$.
- From weak duality, we know that $d^\star \leq p^\star$, which implies with the above that the inequality goes does ways, so $p^\star = d^\star$. It als follows that $x^\star$ is a global minimizer.

□

13.4 Complementarity

Remark
The last KKT condition (13.7e) is called the *complementarity* condition. The situation for $h_i(x)$ and μ_i that satisfy the three conditions $h_i \geq 0$, $\mu_i \geq 0$, and $h_i\mu_i = 0$ is visualized in Figure 13.4.

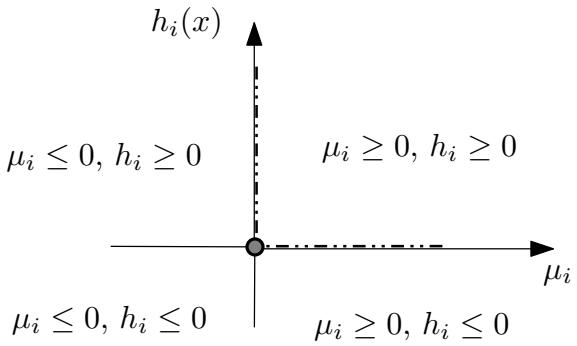


Figure 13.4: The complementarity condition. The origin, $h_i = 0$, and $\mu_i = 0$ make the complementarity condition non-smooth. Note that strict complementarity makes many theorems easier because it avoids the origin.

Definition 13.7: Weakly/Strictly active constraints
Regard a KKT point $(x^\star, \lambda, \mu)$. For $i \in \mathcal{A}(x^\star)$, we say h_i is *weakly active* if $\mu_i = 0$, otherwise, if $\mu_i > 0$, we call it *strictly active*. We say that *strict complementarity* holds at this KKT point iff all active constraints are strictly active. We define the set of weakly active constraints to be $\mathcal{A}_0(x^\star, \mu)$ and the set of strictly active constraints $\mathcal{A}_+(x^\star, \mu)$. The sets are disjoint, and $\mathcal{A}(x^\star) = \mathcal{A}_0(x^\star, \mu) \cup \mathcal{A}_+(x^\star, \mu)$.

13.5 Second Order Conditions

Definition 13.8
Regard the KKT point $(x^\star, \lambda, \mu)$. The critical cone $C(x^\star, \mu)$ is the following set:

$$C(x^\star, \mu) = \left\{ p \in \mathbb{R}^n \;\middle|\; \nabla g_i(x^\star)^\top p = 0, \; \begin{cases} \nabla h_i(x^\star)^\top p = 0 \text{ if } i \in \mathcal{A}_+(x^\star, \mu), \\ \nabla h_i(x^\star)^\top p \geq 0 \text{ if } i \in \mathcal{A}_0(x^\star, \mu) \end{cases} \right\}. \tag{13.16}$$

Note: $C(x^\star, \mu) \subset \mathcal{F}(x^\star)$. In case that LICQ holds, even $C(x^\star, \mu) \subset T_\Omega(x^\star)$. Thus, the critical cone is a subset of all feasible directions. In fact, it contains all feasible directions which are from first-order information neither uphill nor downhill directions, as the following theorem shows.

Theorem 13.6: Criticality of Critical Cone
Regard the KKT point $(x^\star, \lambda, \mu)$ with LICQ, then $\forall p \in T_\Omega(x^\star)$ holds:

$$p \in C(x^\star, \mu) \iff \nabla f(x^\star)^\top p = 0. \tag{13.17}$$

Proof. Use $\nabla_x \mathcal{L}(x^\star, \lambda, \mu) = 0$ to get for any $p \in C(x^\star, \mu)$:

$$\nabla f(x^\star)^\top p = \lambda^\top \underbrace{\nabla g^\top p}_{=0} + \sum_{i \text{ s.t. } \mu_i > 0} \mu_i \underbrace{\nabla h_i(x^\star)^\top p}_{=0} + \sum_{i \text{ s.t. } \mu_i = 0} \mu_i \nabla h_i(x^\star)^\top p = 0. \tag{13.18}$$

Conversely, if $p \in T_\Omega(x^\star)$, then all terms on the right-hand side must be non-negative, so that $\nabla f(x^\star)^\top p = 0$ implies in particular $\sum_{i,\, \mu_i > 0} \mu_i \nabla h_i(x^\star)^\top p = 0$, which implies $\nabla h_i(x^\star)^\top p = 0$ for all $i \in \mathcal{A}_+(x^\star, \mu)$, i.e., $p \in C(x^\star, \mu)$. □

Example 13.4

$$\begin{aligned} \underset{x \in \mathbb{R}^2}{\text{minimize}} \quad & x_2 && \text{(13.19a)} \\ \text{subject to} \quad & 1 - x_1^2 - x_2^2 \geq 0. && \text{(13.19b)} \end{aligned}$$

$$x^\star = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad \nabla h(x) = \begin{bmatrix} -2x_1 \\ -2x_2 \end{bmatrix}, \quad \nabla f(x) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{13.20a}$$

$\mu = ?$

$$\nabla f(x^\star) - \nabla h(x^\star)\mu = 0, \quad \implies \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 2 \end{bmatrix} \mu = 0 \implies \mu = \frac{1}{2}. \tag{13.21a}$$

Thererofore, $x^\star = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $\mu^\star = \frac{1}{2}$ is a KKT point.

$$T_\Omega(x^\star) = \mathcal{F}(x^\star) = \left\{ p \mid \nabla h^\top p \geq 0 \right\} = \left\{ p \mid \begin{bmatrix} 0 \\ 2 \end{bmatrix}^\top p \geq 0 \right\}, \tag{13.22a}$$

$$C(x^\star, \nabla) = \left\{ p \mid \nabla h^\top p = 0 \text{ if } \mu > 0 \right\} = \left\{ p \mid \begin{bmatrix} 0 \\ 2 \end{bmatrix}^\top p = 0 \right\}. \tag{13.22b}$$

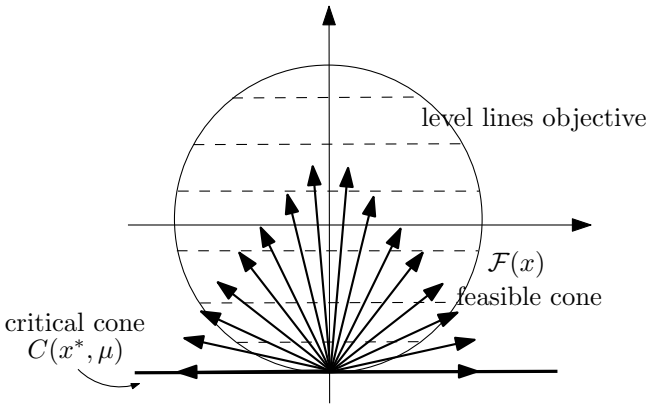


Figure 13.5: Conceptual visualization of Example 13.4.

Theorem 13.7: SONC
Regard $x^\star$ with LICQ. If $x^\star$ is a local minimizer of the NLP, then:

i. $\exists \lambda^\star, \mu^\star$ so that KKT conditions hold;

ii. $\forall p \in C(x^\star, \mu^\star)$ holds that $p^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star, \mu^\star) p \geq 0$.

Theorem 13.8: SOSC
If $x^\star$ satisfies LICQ and:

i. $\exists \lambda^\star, \mu^\star$ so that KKT conditions hold;

ii. $\forall p \in C(x^\star, \mu^\star),\ p \neq 0$, holds that $p^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star, \mu^\star) p > 0$,

then $x^\star$ is a strict local minimizer.

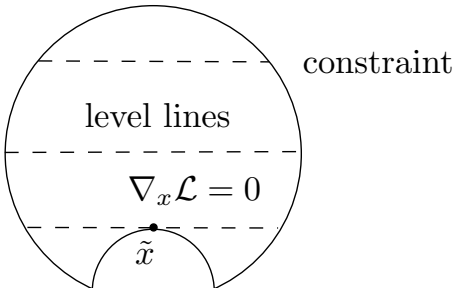


Figure 13.6: Motivation for Theorem 13.8: the point $\tilde{x}$ is not a local minimizer.

Note: $\nabla_x^2 \mathcal{L}(x^\star, \lambda^\star, \mu^\star) = \nabla^2 f(x^\star) - \sum \lambda_i^\star \nabla^2 g_i(x^\star) - \sum \mu_i^\star \nabla^2 h_i(x^\star)$, i.e., $\nabla_x^2 \mathcal{L}$ contains the curvature of the constraints.

Sketch of proof of both theorems. Regard the following restriction of the feasible set ($\bar{\Omega} \subset \Omega$):

$$\bar{\Omega} = \left\{ x \;\middle|\; g(x) = 0,\ h_i(x) = 0 \text{ if } i \in \mathcal{A}_+(x^\star, \mu),\ h_i(x) \geq 0 \text{ if } i \in \mathcal{A}_0(x^\star, \mu) \right\}. \tag{13.23}$$

The critical cone is the tangent cone of this set $\bar{\Omega}$. First, for any feasible direction $p \in T_\Omega(x^\star) \setminus C(x^\star, \mu)$ we have $\nabla f(x^\star)^\top p > 0$. Thus, the difficult directions are those in the critical cone only. So let us regard points in the set $\bar{\Omega}$. For fixed λ, μ we have for all $x \in \bar{\Omega}$:

$$\begin{aligned} \mathcal{L}(x, \lambda, \mu) &= f(x) - \sum \lambda_i \underbrace{g_i(x)}_{=0} - \sum_{i \text{ s.t. } \mu_i > 0} \mu_i \underbrace{h_i(x)}_{=0} - \underbrace{\sum_{i \text{ s.t. } \mu_i = 0} \mu_i h_i(x)}_{=0} \\ &= f(x). \end{aligned} \tag{13.24}$$

Also: $\nabla_x \mathcal{L}(x^\star, \lambda^\star, \mu^\star) = 0$. So for all $x \in \bar{\Omega}$ we have:

$$\begin{aligned} f(x) &= \mathcal{L}(x, \lambda, \mu) && \text{(13.25a)} \\ &= \underbrace{\mathcal{L}(x^\star, \lambda^\star, \mu^\star)}_{=f(x^\star)} + \underbrace{\nabla_x \mathcal{L}(x^\star, \lambda^\star, \mu^\star)^\top}_{=0}(x - x^\star) && \text{(13.25b)} \\ &\qquad + \frac{1}{2}(x - x^\star)^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star, \mu^\star)(x - x^\star) + o(\|x - x^\star\|^2) \\ &= f(x^\star) + \frac{1}{2}(x - x^\star)^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star, \mu^\star)(x - x^\star) + o(\|x - x^\star\|^2). && \text{(13.25c)} \end{aligned}$$

□

Example 13.5
Regard the example from before:

$$\mathcal{L}(x, \mu) = x_2 - \mu(1 - x_1^2 - x_2^2), \nabla_x \mathcal{L} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \mu \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}, \nabla_x^2 \mathcal{L} = 0 + \mu \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}. \quad \text{(13.26a)}$$

For $\mu = \frac{1}{2}$ and $x^\star = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, we have:

$$C(x^\star, \mu) = \left\{ p \mid \nabla h^\top p = 0 \right\} = \left\{ p \mid \begin{bmatrix} 0 \\ 2 \end{bmatrix}^\top p = 0 \right\} = \left\{ \begin{bmatrix} p_1 \\ 0 \end{bmatrix} \mid p_1 \in \mathbb{R} \right\}, \quad \text{(13.27a)}$$

$$\nabla_x^2 \mathcal{L}(x^\star, \lambda, \mu) = \frac{1}{2} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad \text{(13.27b)}$$

$$\text{SONC:} \quad \underbrace{\begin{bmatrix} p_1 \\ 0 \end{bmatrix}^\top \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ 0 \end{bmatrix}}_{=p_1^2} \geq 0. \quad \text{(13.28)}$$

$$\text{SOSC:} \quad \begin{cases} \text{if } p \neq 0,\ p \in C: & p^\top \nabla_x^2 \mathcal{L} p > 0, \\ \text{if } p_1 \neq 0: & p_1^2 > 0. \end{cases} \quad \text{(13.29)}$$

Example 13.6

$$\begin{aligned} &\underset{x \in \mathbb{R}^2}{\text{minimize}} && x_2 && \text{(13.30a)} \\ &\text{subject to} && 2x_2 \geq x_1^2 - 1 - (x_2 + 1)^2. && \text{(13.30b)} \end{aligned}$$

Here $x^\star = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $\mu = \frac{1}{2}$ is still a KKT point: $\nabla_x \mathcal{L}(x^\star, \mu) = 0$. However, $\nabla_x^2 \mathcal{L}(x^\star, \mu) = \mu \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$.

Chapter 14

Inequality Constrained Optimization Algorithms

For simplicity, drop equalities and regard:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(14.1a)} \\ \text{subject to} \quad & h(x) \geq 0 && \text{(14.1b)} \end{aligned}$$

In the KKT conditions we had (for $i = 1, \ldots, q$):

1. $\nabla f(x) - \sum_{i=1}^{q} \nabla h_i(x) \mu_i = 0$
2. $h_i(x) \geq 0$
3. $\mu_i \geq 0$
4. $\mu_i h_i(x) = 0$

Conditions 2, 3 and 4 are non-smooth, which implies that Newton's method will not work here.

14.1 Quadratic Programming via Active Set Method

Regard the QP problem to be solved:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & g^\top x + \frac{1}{2} x^\top B x && \text{(14.2a)} \\ \text{subject to} \quad & Ax + b \geq 0. && \text{(14.2b)} \end{aligned}$$

Assume a convex QP ($B \succcurlyeq 0$). The KKT conditions are necessary and sufficient for global optimality (this is the basis for the algorithm):

$$\begin{aligned} Bx^\star + g - A^\top \mu^\star &= 0 && \text{(14.3a)} \\ Ax^\star + b &\geq 0 && \text{(14.3b)} \\ \mu^\star &\geq 0 && \text{(14.3c)} \\ \mu_i^\star (Ax^\star + b)_i &= 0 && \text{(14.3d)} \end{aligned}$$

for $i = 1, \ldots, q$. How do we find $x^\star$, $\mu^\star$ and the corresponding active set $\mathcal{A}(x^\star) \subset \{1, \ldots, q\}$ so that KKT holds?

Definition 14.1: Index Set

$$\mathbb{A} \subset \{1, \ldots, q\} \text{ "Active"} \tag{14.4a}$$
$$\mathbb{I} = \{1, \ldots, q\} \backslash \mathbb{A} \text{ "Inactive"} \tag{14.4b}$$

Vector division

$$b = \begin{bmatrix} b_{\mathbb{A}} \\ b_{\mathbb{I}} \end{bmatrix} \quad b \in \mathbb{R}^q \tag{14.5}$$

Matrix division

$$A = \begin{bmatrix} A_{\mathbb{A}} \\ A_{\mathbb{I}} \end{bmatrix} \tag{14.6}$$

ie

$$Ax + b \geq 0 \iff A_{\mathbb{A}}x + b_{\mathbb{A}} \geq 0 \text{ AND } A_{\mathbb{I}}x + b_{\mathbb{I}} \geq 0 \tag{14.7}$$

Lemma 14.1
$x^\star$ is a global minimizer of the QP iff there exist an index set $\mathbb{A}$ and $\mathbb{I}$ and a vector $\mu^\star_{\mathbb{A}}$ so that:

$$Bx^\star + g - A_{\mathbb{A}}^\top \mu^\star_{\mathbb{A}} = 0 \tag{14.8a}$$
$$A_{\mathbb{A}}x^\star + b_{\mathbb{A}} = 0 \tag{14.8b}$$
$$A_{\mathbb{I}}x^\star + b_{\mathbb{I}} \geq 0 \tag{14.8c}$$
$$\mu^\star_{\mathbb{A}} \geq 0 \tag{14.8d}$$

and

$$\mu^\star = \begin{bmatrix} \mu^\star_{\mathbb{A}} \\ \mu^\star_{\mathbb{I}} \end{bmatrix} \text{ with } \quad \mu^\star_{\mathbb{I}} = 0 \tag{14.9}$$

The *active set method* idea and the *primal active set method* idea are shown in algorithm 9 and 10.

Algorithm 9 Active set method idea

Choose a set $\mathbb{A}$
Solve (14.8a) and (14.8b) to get $x^\star$ and $\mu^\star$

if (14.8c) and (14.8d) are satisfied **then**
  Solution found
else
  Change set $\mathbb{A}$ by adding or removing constraint indices
end if

For the last step many variants exists: primal, dual, primal-dual, online... E.g., QPSOL, quadprog (Matlab) and qpOASES.

Algorithm 10 Primal active set method in detail

Choose a feasible starting point x_0 with corresponding active set $\mathbb{A}_0$
$k \leftarrow 0$

while *no solution found* **do**
  Solve $B\tilde{x}_k + g - A_{\mathbb{A}_k}^\top \tilde{\mu}_k = 0$ and $A_{\mathbb{A}_k}\tilde{x}_k + b_{\mathbb{A}_k} = 0$
  Go on a line from x_k to $\tilde{x}_k$: $x_{k+1} = x_k + t_k(\tilde{x}_k - x_k)$ with some $t_k \in [0, 1]$ so that x_{k+1} is feasible

  if $t_k < 1$ **then**

    $\mathbb{A}_{k+1} \leftarrow \mathbb{A}_k \cup \{i^\star\}$ *(Add a blocking constraint $i^\star$ to $\mathbb{A}$)*
    $k \leftarrow k + 1$

  else if $t_k = 1$ **then**
    ($\tilde{x}_k$ is feasible)

    if $\tilde{\mu}_k \geq 0$ **then**
      Solution found
    else
      Drop index i^{**} in $\mathbb{A}_k$ with $\tilde{\mu}_{k,i^{**}} < 0$ and $\mathbb{A}_{k+1} = \mathbb{A}_k \backslash \{i^{**}\}$
      $k \leftarrow k + 1$
    end if

  end if
end while

Remark: we can prove that $f(x_{k+1}) \leq f(x_k)$ (with f the quadratic performance index).

Example 14.1: Active set method
Consider the problem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & \|x\|_2^2 && \text{(14.10a)} \\ \text{subject to} \quad & x_1 \geq 1, && \text{(14.10b)} \\ & x_2 + 1 \geq 0, && \text{(14.10c)} \\ & 1 - x_2 \geq 0. && \text{(14.10d)} \end{aligned}$$

We choose x_0 as a feasible starting point with corresponding active set $\mathbb{A}_0 = \{3\}$. At the first iteration, the infeasible point $\tilde{x}_0$ is obtained by solving the two equations. This point will be avoided by adding second constraint (1 on Figure 14.1) as a blocking constraint because $t_0 < 1$. The new iterate is x_1 with active set $\mathbb{A}_1 = \{1, 3\}$. For the second iteration, by solving the equations we get $\tilde{x}_1 = x_1$ and $t_k = 1$. Regarding the negative multiplier $\tilde{\mu}_{k,3}$ we drop index 3 in $\mathbb{A}_1$ and get $\mathbb{A}_2 = \{1\}$. The next iteration has as conclusion $\tilde{x}_2 = x^\star$ and is the last iteration.

It can be proven that $f(x_{k+1}) < f(x_k)$ in each iteration.

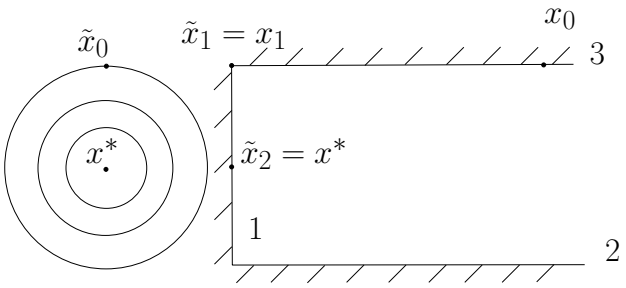


Figure 14.1: Visualization of Example 14.1.

14.2 Sequential Quadratic Programming (SQP)

Regard the NLP:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(14.11a)} \\ \text{subject to} \quad & h(x) \geq 0. && \text{(14.11b)} \end{aligned}$$

The SQP idea is to solve in each iteration the QP:

$$\begin{aligned} \underset{p \in \mathbb{R}^n}{\text{minimize}} \quad & \nabla f(x_k)^\top p + \frac{1}{2} p^\top B p && \text{(14.12a)} \\ \text{subject to} \quad & h(x_k) + \frac{\partial h}{\partial x}(x_k) p \geq 0. && \text{(14.12b)} \end{aligned}$$

Local convergence would follow from equality constrained optimization if the active set of the QP is the same as the active set of the NLP, at least in the last iterations.

Theorem 14.1: Robinson
If $x^\star$ is a local minimizer of the NLP with LICQ and strict complementarity and if x_k is close enough to $x^\star$ and $B \succcurlyeq 0$ and B is positive definite on the nullspace of the linearization of the active constraints, then the solution of the QP has the same active set as the NLP.

Proof. Define $\mathbb{A} = \mathcal{A}(x^\star)$ and regard:

$$\nabla f(x) + Bp - \frac{\partial h_\mathbb{A}}{\partial x}(x)^\top \mu_\mathbb{A}^{\mathrm{QP}} = 0 \tag{14.13a}$$

$$h_\mathbb{A}(x) + \frac{\partial h_\mathbb{A}}{\partial x}(x)p = 0 \tag{14.13b}$$

this defines an implicit function

$$\begin{bmatrix} p(x,B) \\ \mu_\mathbb{A}^{\mathrm{QP}}(x,B) \end{bmatrix} \tag{14.14}$$

with

$$p(x^\star, B) = 0 \text{ and } \mu_\mathbb{A}^{\mathrm{QP}}(x^\star, B) = \mu_\mathbb{A}^\star) \tag{14.15}$$

This follows from

$$\nabla f(x^\star) + Bp - \frac{\partial h_\mathbb{A}}{\partial x}(x^\star)^\top \mu_\mathbb{A}^\star = 0 \iff \nabla_x \mathcal{L}(x^\star, \mu^\star) = 0 \tag{14.16a}$$

$$h_\mathbb{A}(x^\star) + \frac{\partial h_\mathbb{A}}{\partial x}(x^\star)0 = 0 \tag{14.16b}$$

which hold because of

$$h_\mathbb{A}(x^\star) = 0 \tag{14.17a}$$

$$h_\mathbb{I}(x^\star) > 0 \tag{14.17b}$$

$$\mu_\mathbb{I}^\star = 0 \tag{14.17c}$$

Note that $\mu_\mathbb{A}^\star > 0$ because of strict complementarity.
For x close to $x^\star$, due to continuity of $p(x,B)$ and $\mu_\mathbb{A}^{\mathrm{QP}}(x,B)$ we still have $h_\mathbb{I}(x) > 0$ and

$$\mu_\mathbb{A}^{\mathrm{QP}}(x,B) > 0 \tag{14.18}$$

and even more:

$$h_\mathbb{I}(x) + \frac{\partial h_\mathbb{I}}{\partial x}(x)p(x,B) > 0 \tag{14.19}$$

Therefore a solution of the QP has the same same active set as the NLP and also satisfies strict complementarity. □

Remark
We can generalise his Theorem to the case where the jacobian $\frac{\partial h}{\partial x}(x_h)$ is only approximated.

14.3 Powell's Classical SQP Algorithm

For an equality and inequality constrained NLP, we can use the BFGS algorithm as before but:

1. We solve an *inequality constrained* QP instead of a linear system
2. We use $T_1(x) = f(x) + \sigma \|g(x)\|_1 + \sigma \sum_{i=1}^{q} |\min(0, h_i(x))|$
3. Use full Lagrange gradient $\nabla_x \mathcal{L}(x, \lambda, \mu)$ in the BFGS formula

(eg "fmincon" in Matlab).

14.4 Interior Point Methods

The IP method is an alternative for the active set method for QPs or LPs or for the SQP method. The previous methods had problems with the non-smoothness in the KTT-conditions (2), (3) and (4) (for $i = 1, \ldots, q$):

1. $\nabla f(x) - \sum_{i=1}^{q} \mu_i \nabla h_i(x) = 0$
2. $h_i(x) \geq 0$
3. $\mu_i \geq 0$
4. $\mu_i h_i(x) = 0$.

The IP-idea is to replace 2,3 and 4 by a smooth condition (which is an approximation): $h_i(x)\mu_i = \tau$ with $\tau > 0$ becoming smaller at each iteration. This makes this equation smooth in some sense, as illustrated in Figure 14.2. One can show that the solution $(\bar{x}(\tau), \bar{\mu}(\tau))$ of the modified KKT-

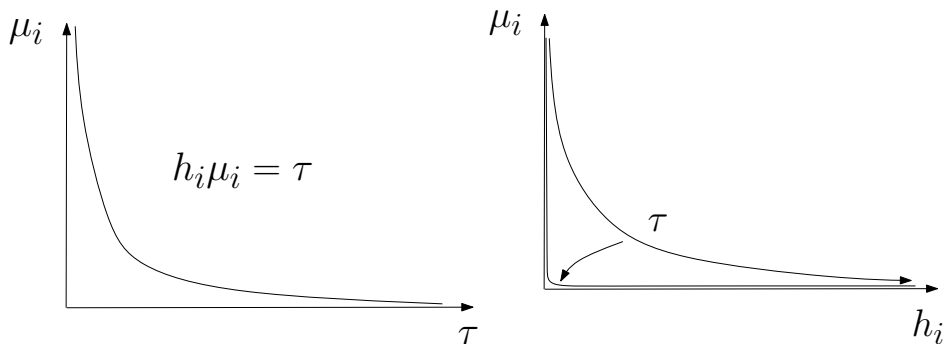


Figure 14.2: The interior point method idea: make the KKT-conditions a smooth root finding problem.

conditions will be close to the solution of the original KKT-conditions if τ is small enough:

$$\bar{x}(\tau) \underset{\tau \to 0}{\longrightarrow} x^\star \tag{14.20a}$$

$$\bar{\mu}(\tau) \underset{\tau \to 0}{\longrightarrow} \mu^\star \tag{14.20b}$$

Solving the modified KKT-conditions with Newton's method produces the following algorithm.

Numerical Method 14.1: Primal-Dual Interior Point method

1. Start with a big $\tau \gg 0$, choose $\beta \in (0,1)$
2. Solve the following modified KKT conditions to get $\bar{x}(\tau)$ and $\bar{\mu}(\tau)$:

$$\nabla f(x) - \sum_{i=1}^{q} \mu_i \nabla h_i(x) = 0 \tag{14.21a}$$

$$h_i(x)\mu_i - \tau = 0 \quad i = 1, \dots, q \tag{14.21b}$$

3. Replace $\tau \leftarrow \beta\tau$ and go to 2 (warm-start Newton iterations with current solution).

Remark
The set of solutions $\left\{ \begin{bmatrix} \bar{x}(\tau) \\ \bar{\mu}(\tau) \end{bmatrix} \mid \tau \in (0, \infty) \right\}$ is called the *central path.*

Remark
The system of equations (14.21) is equivalent to the FONC of the *Barrier Problem* (BP):

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) - \tau \sum_{i=1}^{q} \log h_i(x). \tag{14.22}$$

Indeed, the FONC of (14.22) is

$$\nabla f(x) - \sum_{i=1}^{q} \frac{\tau}{h_i(x)} \nabla h_i(x) = 0 \iff \begin{cases} \nabla f(x) - \sum_{i=1}^{q} \mu_i \nabla h_i(x) = 0 \\ \mu_i = \frac{\tau}{h_i(x)} \quad i = 1, \dots, q \end{cases} \tag{14.23}$$

Example 14.2: Barrier Problem
The problem

$$\underset{x \in \mathbb{R}}{\text{minimize}} \quad x \tag{14.24a}$$

$$\text{subject to} \quad x \geq 0, \tag{14.24b}$$

could be approximated by the Barrier problem:

$$\underset{x > 0}{\text{minimize}} \quad x - \tau \log(x), \tag{14.25}$$

which is visualized in Figure 14.3.

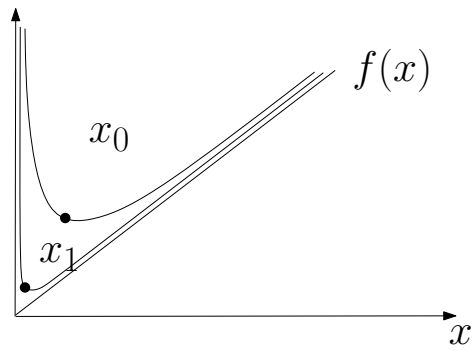


Figure 14.3: Visualization of Example 14.2.

Optimization software based on interior point methods: For convex problems, IP methods are well understood with strong complexity results. For LPs and QPs and other convex problems, the IP method is successfully implemented e.g. in OOQP, CPLEX, SeDuMi, SDPT3, CVX, or CVXGEN. But IP methods also exist for general nonlinear programs where they still work very reliable. A very powerful and widely used IP method for sparse NLP is the open-source code IPOPT.

Chapter 15

Optimal Control Problems

We regard a dynamical system with dynamics

$$x_{k+1} = f(x_k, u_k) \tag{15.1}$$

with u_k the "controls" or "inputs" and x_k the "states". Let $x_k \in \mathbb{R}^{n_x}$ and let $u_k \in \mathbb{R}^{n_u}$ with $k = 0, \ldots, N-1$. If we know the initial state x_0 and the controls $u_0, \ldots, u_{N-1}$ we could simulate the system to obtain all other states. In optimization, we might have different requirements than just a fixed initial state. We might, for example, have both a fixed initial state and a fixed terminal state that we want to reach. Or we might just look for periodic sequences with $x_0 = x_N$. All these desires on the initial and the terminal state can be expressed by a boundary constraint function

$$r(x_0, x_N) = 0. \tag{15.2}$$

For the case of fixed initial value, this function would just be

$$r(x_0, x_N) = x_0 - \bar{x}_0 \tag{15.3}$$

where $\bar{x}_0$ is the fixed initial value and not an optimization variable. Another example would be to have both ends fixed, resulting in a function r of double the state dimension, namely:

$$r(x_0, x_N) = \begin{bmatrix} x_0 - \bar{x}_0 \\ x_N - \bar{x}_N \end{bmatrix}. \tag{15.4}$$

Finally, periodic boundary conditions can be imposed by setting

$$r(x_0, x_N) = (x_0 - x_N). \tag{15.5}$$

An illustration of inputs and states is given in Figure 15.1.

15.1 Optimal Control Problem (OCP) Formulation

The simplified optimal control problem in discrete time that we regard in this chapter is the following equality constrained NLP.

$$\underset{x_0, u_0, x_1, \ldots, u_{N-1}, x_N}{\text{minimize}} \quad \sum_{k=0}^{N-1} L(x_k, u_k) + E(x_N) \tag{15.6a}$$

$$\text{subject to} \quad x_{k+1} - f(x_k, u_k) = 0, \quad \text{for} \quad k = 0, \ldots, N-1, \tag{15.6b}$$

$$r(x_0, x_N) = 0. \tag{15.6c}$$

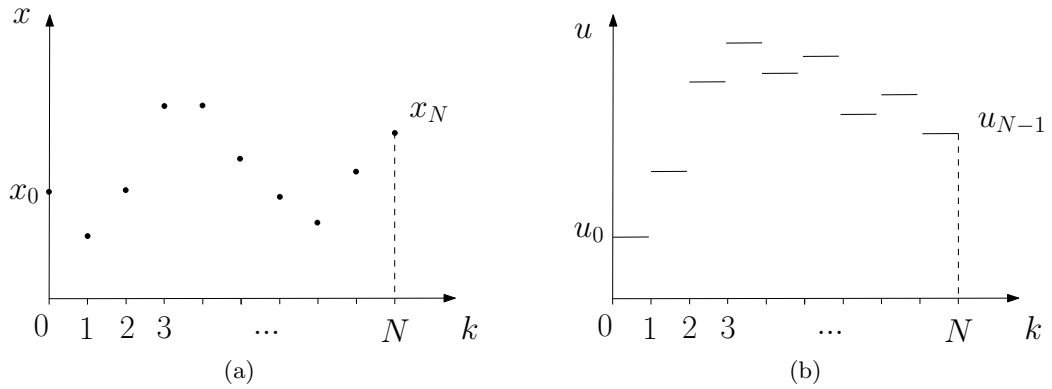


Figure 15.1: A conceptual example of an optimal control problem with states (a) and controls (b).

Remark
One could also add inequality constraints on the form: $h(x_k, u_k) \geq 0$. In this lecture, we only present the case without these constraints for simplicity.

Remark
We can also add some optimization variables p that are not time-dependant. An example would be to optimize over the size of a chemical reactor, with some constraints on the evolution of the reaction.
To keep the same optimization pattern, one can add an extra state $x_k \leftarrow \begin{bmatrix} x_k \\ p_k \end{bmatrix}$ with the constraints $p_{k+1} = p_k$.

We also remark that any free parameter p could be added to the optimisation formulation above, e.g. the constant size of a vessel in a chemical reactor. For this we could define an extra dummy state for $k = 0, \ldots, N-1$

$$p_{k+1} = p_k. \tag{15.7}$$

15.2 KKT Conditions of Optimal Control Problems

First summarize the variables $w = \{x_0, u_0, x_1, u_1, \ldots, u_{N-1}, x_N\}$ and summarize the multipliers $\lambda = \{\lambda_1, \ldots, \lambda_N, \lambda_r\}$. The optimal control problem has the form

$$\underset{w}{\text{minimize}} \quad F(w) \tag{15.8a}$$

$$\text{subject to} \quad G(w) = 0, \tag{15.8b}$$

where we have defined:

$$G(w) = \begin{bmatrix} x_1 - f(x_0, u_0) \\ x_2 - f(x_1, u_1) \\ \vdots \\ x_N - f(x_{N-1}, u_{N-1}) \\ r(x_0, x_N) \end{bmatrix} \tag{15.9a}$$

The Lagrangian function has the form

$$\mathcal{L}(w, \lambda) = F(w) - \lambda^\top G(w) \tag{15.10a}$$

$$= \sum_{k=0}^{N-1} L(x_k, u_k) + E(x_N) - \sum_{k=0}^{N-1} \lambda_{k+1}^\top (x_{k+1} - f(x_k, u_k)) \tag{15.10b}$$

$$- \lambda_r^\top r(x_0, x_N) \tag{15.10c}$$

The KKT-conditions of the problem are

$$\nabla_w \mathcal{L}(w, \lambda) = 0 \tag{15.11a}$$

$$G(w) = 0 \tag{15.11b}$$

In more detail, the derivative of $\mathcal{L}$ with respect to x_k, where $k = 0$ and $k = N$ are considered as special cases. First $k = 0$ is treated

$$\nabla_{x_0} \mathcal{L}(w, \lambda) = \nabla_{x_0} L(x_0, u_0) + \frac{\partial f}{\partial x_0}(x_0, u_0)^\top \lambda_1 - \frac{\partial r}{\partial x_0}(x_0, x_N)^\top \lambda_r = 0. \tag{15.12}$$

Then the case for $k = 1, \ldots, N-1$ is treated

$$\nabla_{x_k} \mathcal{L}(w, \lambda) = \nabla_{x_k} L(x_k, u_k) - \lambda_k + \frac{\partial f}{\partial x_k}(x_k, u_k)^\top \lambda_{k+1} = 0. \tag{15.13}$$

Now the special case $k = N$

$$\nabla_{x_N} \mathcal{L}(w, \lambda) = \nabla_{x_N} E(x_N) - \lambda_N - \frac{\partial r}{\partial x_N}(x_0, x_N)^\top \lambda_r = 0. \tag{15.14}$$

The Lagrangian with respect to u is calculated, for $k = 0, \ldots, N-1$

$$\nabla_{u_k} \mathcal{L}(w, \lambda) = \nabla_{u_k} L(x_k, u_k) + \frac{\partial f}{\partial u_k}(x_k, u_k)^\top \lambda_{k+1} = 0. \tag{15.15}$$

The last two conditions are

$$x_{k+1} - f(x_k, u_k) = 0 \quad k = 0, \ldots, N-1 \tag{15.16a}$$

$$r(x_0, x_N) = 0 \tag{15.16b}$$

The equations (15.12) till (15.16b) are the KKT-system of the OCP. There exist different approaches to solve this system. One method is to solve equations (15.12) to (15.16b) directly, this is called the simultaneous approach. The other approach is to calculate all the states in (15.16a) by forward elemination. This is called the sequential approach and treated first.

15.3 Sequential Approach to Optimal Control

This method is also called "single shooting" or "reduced approach". The idea is to keep only x_0 and $U = [u_0^\top, \dots, u_{N-1}^\top]^\top$ as variables. The states $x_1, \dots, x_N$ are eleminated recursively by

$$\bar{x}_0(x_0, U) = x_0 \tag{15.17a}$$
$$\bar{x}_{k+1}(x_0, U) = f(\bar{x}_k(x_0, U), u_k) \tag{15.17b}$$

Then the optimal control problem is equivalent to a problem with less variables

$$\underset{x_0, U}{\text{minimize}} \quad \sum_{k=0}^{N-1} L(\bar{x}_k(x_0, U), u_k) + E(\bar{x}_N(x_0, U)) \tag{15.18a}$$
$$\text{subject to} \quad r(x_0, \bar{x}_N(x_0, U)) = 0. \tag{15.18b}$$

Note that equation (15.16a) is implicitly satisfied. This is called the reduced optimal control problem. It can be solved by e.g. Newton type method (SQP if inequalities are present). If $r(x_0, x_N) = x_0 - \bar{x}_0$ one can also eliminate $x_0 \equiv \bar{x}_0$. The optimality conditions for this problem are found in the next subsection.

15.4 Backward Differentiation of Sequential Lagrangian

The Lagrangian function is given by

$$\bar{\mathcal{L}}(x_0, U, \lambda_r) = \sum_{k=0}^{N-1} L(\bar{x}_k(x_0, U), u_k) + E(\bar{x}_N(x_0, U)) - \lambda_r^\top r(x_0, \bar{x}_N(x_0, U)) \tag{15.19}$$

so the KKT conditions for the reduced optimal control problem are

$$\nabla_{x_0} \bar{\mathcal{L}}(x_0, U, \lambda_r) = 0 \tag{15.20a}$$
$$\nabla_{u_k} \bar{\mathcal{L}}(x_0, U, \lambda_r) = 0 \quad k = 0, \dots, N-1 \tag{15.20b}$$
$$r(x_0, \bar{x}_N(x_0, U)) = 0 \tag{15.20c}$$

Usually derivatives are computed by finite differences, the I-Trick or forward automatic differention (AD). But here, backward automatic differentiation (AD) is more efficient. The result for backward AD to the equations (15.20a) to (15.20c) to get $\nabla_{x_0} \bar{\mathcal{L}}$ and $\nabla_{u_k} \bar{\mathcal{L}}$ is stated in Algorithm 11. Compare this algorithm with equations (15.12) to (15.15) where $\bar{\lambda}_k \equiv \lambda_k$.
We get a second interpretation to the sequential approach with backward AD: when solving (15.12) to (15.16b) we eliminate all equations that can be eliminated by (15.16a), (15.14) and (15.13). Only the equations (15.16b), (15.12) and (15.15) remain. Backward automatic differentiation (AD) gives the gradient at a cost scaling linearly with N and forward differences with respect to $u_0, \dots, u_{N-1}$, would grow with N^2.
The sequential approach and backward automatic differentiation (AD) leads to a small *dense* (Jacobians are dense matrices) nonlinear system in variables $(x_0, u_0, \dots, u_{N-1}, \lambda_r)$. The next section tries to avoid the dense Jacobians.

Algorithm 11 Result of backward AD to KKT-ROCP

Inputs
x_0, u_0,...,u_{N-1}, λ_r
Outputs
r, $\nabla_{u_0}\mathcal{L}, \ldots, \nabla_{u_{N-1}}\mathcal{L}$ and $\nabla_{x_0}\mathcal{L}$

Set $k = 0$, execute forward sweep:
repeat
$x_{k+1} = f(x_k, u_k)$
$k = k + 1$
until $k = N - 1$
Get $r(x_0, x_N)$

Set $\lambda_N = \nabla E(x_N) - \frac{\partial r}{\partial x_N}(x_0, x_N)^\top \lambda_r$
Set $k = N - 1$, execute backward sweep:
repeat
$\lambda_k = \nabla_{x_k} L(x_k, u_k) + \frac{\partial f}{\partial x_k}(x_k, u_k)^\top \lambda_{k+1}$
$\nabla_{u_k}\mathcal{L} = \nabla_{u_k} L(x_k, u_k) + \frac{\partial f}{\partial u_k}(x_k, u_k)^\top \lambda_{k+1}$
$k = k - 1$
until $k = 0$

Compute $\nabla_{x_0}\mathcal{L} = \lambda_0 - \frac{\partial r}{\partial x_0}(x_0, x_N)^\top \lambda_r$

15.5 Simultaneous Optimal Control

This method is also called "multiple shooting" or "one shot optimization". The idea is to solve (15.12) to (15.16b) directly by a sparsity exploiting Newton type method. If we regard the original OCP, it is an NLP in variables $w = (x_0, u_0, x_1, u_1, \ldots, u_{N-1}, x_N)$ with multipliers $(\lambda_1, \ldots, \lambda_N, \lambda_r) = \lambda$. In the SQP method we get

$$w_{k+1} = w_k + \Delta w_k \tag{15.21a}$$
$$\lambda_{k+1} = \lambda_k^{QP} \tag{15.21b}$$

by solving

$$\underset{\Delta w}{\text{minimize}} \quad \nabla_w F(w_k)^\top \Delta w + \frac{1}{2}\Delta w^\top B_k \Delta w \tag{15.22a}$$
$$\text{subject to} \quad G(w) + \frac{\partial G}{\partial w}(w)\Delta w = 0. \tag{15.22b}$$

If we use $B_k = \nabla_w^2 \mathcal{L}(w_k, \lambda_k)$, this QP is very structured and equivalent to

$$\underset{\Delta x_0, \Delta u_0, \ldots, \Delta x_N}{\text{minimize}} \quad \sum_{k=0}^{N} \frac{1}{2} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix}^\top Q_k \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix}^\top g_k \tag{15.23a}$$
$$\text{subject to} \quad r(x_0, x_N) + \frac{\partial r(x_0, x_N)}{\partial x_0}\Delta x_0 + \frac{\partial r(x_0, x_N)}{\partial x_N}\Delta x_N = 0, \tag{15.23b}$$
$$x_{k+1} + \Delta x_{k+1} = f(x_k, u_k) + A_k \Delta x_k + B_k \Delta u_k, \quad \text{for} \quad k = 0, \ldots, N-1, \tag{15.23c}$$

with $u_N = 0$ and, for $k = 0, \ldots, N$:

$$Q_k = \nabla^2_{(x_k, u_k)} \mathcal{L}, \qquad g_k = \nabla_{(x_k, u_k)} L(x_k, u_k), \tag{15.24a}$$
$$A_k = \frac{\partial f}{\partial x_k}(x_k, u_k), \qquad B_k = \frac{\partial f}{\partial u_k}(x_k, u_k). \tag{15.24b}$$

Note that for $k \neq m$

$$\frac{\partial}{\partial x_k}\frac{\partial}{\partial x_m}\mathcal{L} = 0, \qquad \frac{\partial}{\partial x_k}\frac{\partial}{\partial u_m}\mathcal{L} = 0, \qquad \frac{\partial^2}{\partial u_k \partial u_m}\mathcal{L} = 0. \tag{15.25}$$

This QP leads to a very sparse linear system and can be solved at a cost linear with N. Simultaneous approaches can deal better with unstable systems $x_{k+1} = f(x_k, u_k)$.

Appendix A

Example Report on Student Optimization Projects

In this section a project report written by students of a previous year at the end of the exercise sessions is presented. It comes in the original form without corrections (which would still be applicable), but might serve as an example of how such a report might look like.

Optimal Trajectory Design for a Servo Pneumatic Traction System

by Thijs Dewilde and Dries Van Overbeke

A.0.1 Introduction

Servo Pneumatic Positioning The system consists of an electromechanical actuator, the valve, and a pneumatic actuator or cylinder. (Figure A.1)
An electrically controlled valve with 5 ports and 3 switch positions drives a double-acting pneumatic cylinder. A linear unit is formed by combining the cylinder, piston and slider. The presented valve blocks the mass flows in its center switch position. Also, the valve is proportional which means it can switch continuously between positions. For a desired direction of movement, the piston is preset by controlling the valve accordingly. The valve is able to regulate the air mass flow rate and thus controls the movement of the piston.

Model Equations and States We assume the mass flows $\dot{m}$ are proportional to the valve control input u: $\dot{m}_1 \sim u$ and $\dot{m}_2 \sim u$. The time-dependant (t) model states are the cylinder chamber pressures P_1 and P_2, the velocity v and the position s. We define the model state vector $x = \begin{bmatrix} P_1(t) & P_2(t) & v(t) & s(t) \end{bmatrix}^\top$.
The volumes of the chambers can be computed by the following equations:

$$V_1 = V_t + A_c \cdot (s_{\text{ref}} + s), \tag{A.1a}$$
$$V_2 = V_t + A_c \cdot (L - (s_{\text{ref}} + s)), \tag{A.1b}$$

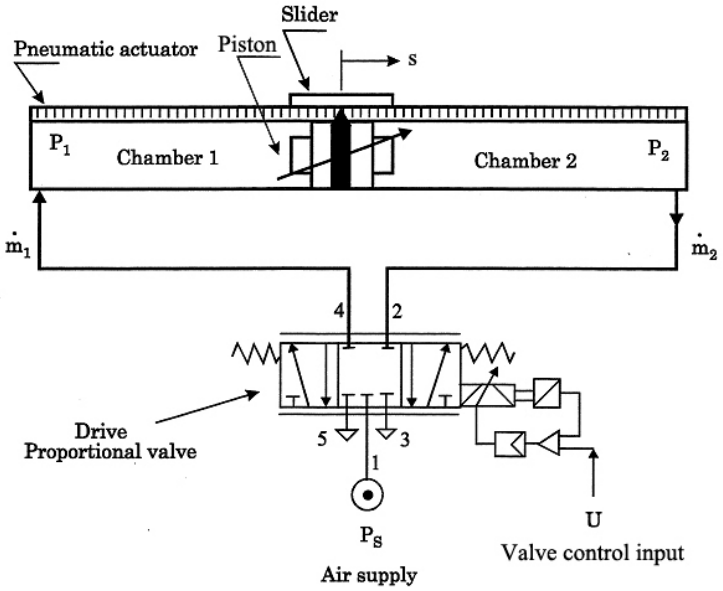


Figure A.1: Servo Pneumatic Traction System

with	s_{ref}	:	Reference position,
	s	:	Relative position of the piston,
	L	:	Cilinder stroke,
	D_c	:	Cilinder diameter,
	A_c	:	Cilinder chamber area,
	L_t	:	Tube length,
	D_t	:	Tube diameter,
	V_t	:	Tube volume.

Now consider an isothermic filling and venting process and differentiate the ideal gas law

$$p_i \cdot V_i = m_i \cdot R \cdot T_i, \tag{A.2}$$

with	P_i	:	absolute pressure in chamber i,
	V_i	:	volume of chamber i, see equations $(A.1)$,
	m_i	:	mass of air in chamber i,
	R	:	specific gas constant for air $(287\frac{J}{kg \cdot K})$,
	T_i	:	absolute temperature of the air in chamber i.

The following expressions for the pressure in the two chambers of the cylinder can be found:

$$\dot{p}_1 = \frac{R \cdot T \cdot \dot{m}_1 - Ac \cdot p_1 \cdot v}{V_1}, \tag{A.3a}$$

$$\dot{p}_2 = \frac{R \cdot T \cdot \dot{m}_2 + Ac \cdot p_2 \cdot v}{V_2}. \tag{A.3b}$$

with	$\dot{m}_1$	:	mass flow of air to chamber 1,
	$\dot{m}_2$	:	mass flow of air to chamber 2,
	v	:	velocity of the piston.

After evaluating the differential pressure $p = p1 - p2$, the traction force on the piston can be calculated $p \cdot Ac$. Newton's second law of motion $F = m \cdot a$ and considering a viscous friction force $b \cdot v$, yields:

$$a = \frac{p \cdot Ac - b \cdot v}{m}, \tag{A.4}$$

with m : mass of the piston and slider,
b : viscous friction coefficient.

Hereby the motion of the slider is entirely modelled, the velocity and position can be derivated by kinematics laws. So we have a dynamic system of the form:

$$\dot{x} = f(x, u, \tau). \tag{A.5}$$

A.0.2 Optimization Problem

Optimal Trajectory The optimal trajectory in this case is the fastest way to reach a position setpoint or in other words the appropriate control input for the proportional valve.

Parameters The control input signal is divided into m intervals, with $m \in \mathbb{N}$. The optimization parameters are the length of a control time interval τ and valve control value for each interval $u(m)$.

Formulation The total elapsed time for reaching a position setpoint or time horizon is minimized, The time horizon T can be regarded as a parameter in the differential equation by a time-transformation $T = m \cdot \tau$, so the objective function looks as follows:

$$J(T). \tag{A.6}$$

The equality constraint function ensures we hold a fixed position setpoint s_{end} at the end, by requiring $v_{end} = 0\frac{m}{s}$ and $a_{end} = 0\frac{m}{s^2}$. We summarize these equality constraints in a function g: $\mathbb{R}^3 \times R^m \to \mathbb{R}^2$:

$$g(x(T)) \tag{A.7}$$

Finally, the inequality constraint function limits the control value $-5 \leq u \leq 5$ and guarantees a positive time interval $T \geq 0$:

$$h(u(m), T) \leq 0. \tag{A.8}$$

We can formulate the problem in standard form:

$$\begin{aligned}
&\underset{x \in \mathbb{R}^4}{\text{minimize}} && J(J) && \text{(A.9a)}\\
&\text{subject to} && \dot{x} = f(x, u(m), \tau), && \text{(A.9b)}\\
&&& s_{\text{start}} = 0, && \text{(A.9c)}\\
&&& v_{\text{start}} = 0, && \text{(A.9d)}\\
&&& p_{1,\text{start}} = 0, && \text{(A.9e)}\\
&&& p_{2,\text{start}} = 0, && \text{(A.9f)}\\
&&& g(x(T)) = 0, && \text{(A.9g)}\\
&&& h(u(m), T) \leq 0. && \text{(A.9h)}
\end{aligned}$$

The model states are updated by a discrete function: $\dot{x} = f(x(k), u(m), \tau)$, To achieve better state updates the time interval is divided into a number h^{-1} of discretization steps.

$$f(x(k), u(m), \tau) = \begin{cases} p_{1,k+1} = p_{1,k} + h \cdot \tau \cdot \frac{R \cdot T \cdot \dot{m}_1 - Ac \cdot p_{1,k} \cdot v}{V_1} \\ p_{2,k+1} = p_{2,k} + h \cdot \tau \cdot \frac{R \cdot T \cdot \dot{m}_2 + Ac \cdot p_{2,k} \cdot v}{V_2} \\ v_{k+1} = v_k + h \cdot \tau \cdot a \\ s_{k+1} = s_k + h \cdot \tau \cdot v \end{cases} \tag{A.10}$$

for $k \in \{1, \ldots, n\}$. The following initial states are chosen:

$$\begin{aligned} p_{1,\text{start}} &= \text{atmosphere pressure } (101325Pa), \\ p_{2,\text{start}} &= \text{atmosphere pressure}, \\ v_{\text{start}} &= 0\tfrac{m}{s}, \\ s_{\text{start}} &= 0m. \end{aligned}$$

Numerical Solution We present a solution obtained by a sequential quadratic programming method (Figure A.2). The relative position setpoint is $200mm$ and we take 10 degrees of freedom (m) for the controls values, so the time horizon T is $10 \cdot \tau$.

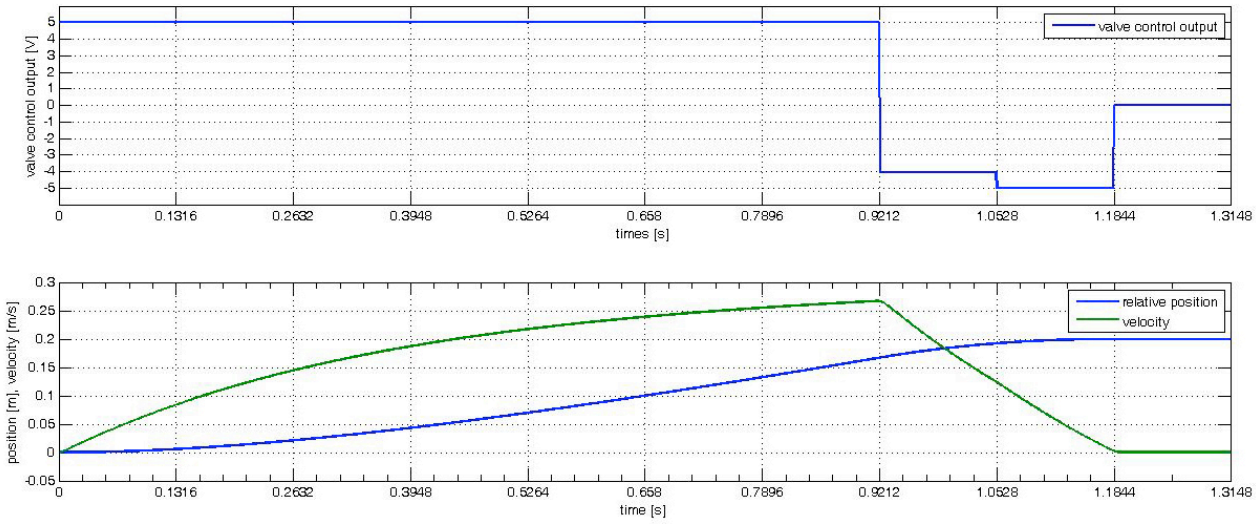


Figure A.2: Numerical Solution

In the plots we can see that the end condition constraints are satisfied and the presented control input is a "bang-bang" solution, with 2 degrees of freedom, namely the control value in between the limits and the control value for the last time interval. We can explain the first degree of freedom for reaching the setpoint position and the last control value to decrease the velocity and acceleration.

Appendix B

Exam Preparation

B.1 Study Guide

Important Chapters and Sections from the Book of Nocedal and Wright

Most, but not all, of the topics of the course are covered in the book by Nocedal and Wright. Particularly useful chapters and sections that are good reading to course participants are

- Appendix A.1 and A.2: all
- Chapter 1: all
- Chapter 2: all
- Chapter 3: Section 3.1, Algorithm 3.1, Section 3.3, Theorem 3.5
- Chapter 4: Algorithm 4.1
- Chapter 6: Formula (6.19)
- Chapter 8: all
- Chapter 10: Sections 10.1, 10.2, 10.3
- Chapter 12: all
- Chapter 16: Sections 16.1, 10.2
- Chapter 18: all
- Chapter 19: Section 19.1, 19.2

Important topics from the course that are not covered well in the book are the Constrained Gauss-Newton method and convex optimization.

Important Chapters and Sections from the Book of Boyd and Vandenberghe

Regarding convex optimization, all topics of the course are covered in the book by Boyd and Vandenberghe. Particularly useful chapters and sections that are good reading to course participants are

- Chapter 1: all
- Chapter 2: Sections 2.1, 2.2, 2.3
- Chapter 3: Sections 3.1, 3.2
- Chapter 4: Sections 4.1, 4.2, 4.3, 4.4, 4.6
- Chapter 5: Sections 5.1, 5.2

B.2 Rehearsal Questions

The following questions might help in rehearsing the contents of the course:

1. What is an optimization problem? Objective, degrees of freedom, constraints. Feasible set? Standard form of NLP.
2. Definition of global and local minimum.
3. State a condition under which there exists at least one minimizers.
4. Types of optimization problems: Linear / Quadratic programming (LP/QP), convex, smooth, mixed-integer, ...
5. When is a function convex? Gives the definition and a characterization when it is twice differentiable. Definition. If it is twice differentiable?
6. When is a set convex? Definition.
7. What is a "stationary" point in the the case of unconstrained smooth optimization?
8. How are gradient and Hessian of a function $f : \mathbb{R}^n \to \mathbb{R}$ defined?
9. What are the first order necessary conditions for optimality (FONC) in the case of unconstrained optimization problems?
10. What are the second order necessary conditions for optimality (SONC) in the case of unconstrained optimization problems?
11. What are the second order sufficient conditions for optimality (SOSC) in the case of unconstrained optimization problems?
12. Basic idea of iterative descent methods?
13. Definition of local convergence rates: q/r-linear, superlinear, quadratic?
14. What is a locally convergent algorithm? What is a globally convergent algorithm? What does the term "globalization" usually mean for optimizers?
15. What is the Armijo condition? Why is it used in line-search algorithms?
16. Why is satisfaction of Armijo condition alone not sufficient to guarantee convergence towards stationary points? Give a counterexample.

17. What is backtracking?
18. Define the *the steepest descent method*. What is the local convergence rate?
19. What is Newton's method for solution of nonlinear equations $F(x) = 0$? How does it iterate, what is the motivation for it. How does it converge locally?
20. How works Newton's method for unconstrained optimization?
21. What are *Newton type / approximate Newton* methods?
22. What is the idea behind Quasi-Newton methods?
23. What is the secant condition? How is it motivated?
24. What is the BFGS formula? Under which condition does it preserve positive definiteness?
25. Prove that the latter condition is necessary for any update formula satisfying the secant condition to stay positive definite.
26. What is a linear least squares problem (unconstrained)? What is a a nonlinear least squares problem (unconstrained)?
27. How does the Gauss-Newton method iterate? When is it applicable?
28. When does the Gauss-Newton method perform well? What local convergence rate does it have?
29. Statistical motivation of least squares terms in estimation problems?
30. What are the differences and similarities between line search and trust region methods?
31. List two ways to compute derivatives with help of computers.
32. What errors occur when computing derivatives with finite differences? Do you know a rule of thumb of how large to choose the perturbation?
33. What is the idea behind Automatic Differentiation (AD)? What is its main advantage?
34. Can AD be applied to compute second order derivatives?
35. There are two ways of AD. Describe briefly. What are the advantages / disadvantages of the two?
36. Write a nonlinear program (NLP) in its standard form. How is the lagrangian function defined?
37. What is the constraint qualification (CQ)? What is the linear independence constraint qualification (LICQ) at some point $\bar{x}$?
38. What are the Karush-Kuhn-Tucker (KKT) conditions for optimality? Why is it useful?
39. What are the first order necessary conditions for optimality (FONC) (constrained)?
40. What are the second order necessary conditions for optimality (SONC) (constrained)?

41. What are the second order sufficient conditions for optimality (SOSC) (constrained)?
42. What is the "active set of constraints"?
43. Give a standardform of a QP.
44. When is a QP convex?
45. What is the main idea of an active set strategy?
46. What is the main idea behind an SQP method (for inequality constrained problems)?
47. What is the L_1-penalty method for equality-constrained problems? Under which condition is it "exact", i.e. has the same local minima as the original NLP?
48. How works Newton's method for equality constrained optimization?
49. What local convergence rate does an SQP method with exact Hessian usually have?
50. What is the basic idea of interior point methods? Gives two views this class of method.
51. What is the Lagrangian dual function of a general NLP?
52. What is the dual problem of a general NLP?
53. What is weak duality? To which problems does it apply?
54. What is strong duality? Under which sufficient conditions does it apply?
55. What is a semidefinite program (SDP)? Give a standardform.
56. How would you reformulate the following eigenvalue optimization problem into an SDP for A_0, A_1, A_2 three symmetric matrices?

$$\underset{x \in \mathbb{R}^2}{\text{minimize}} \quad \lambda_{\max}(A_0 + x_1 A_1 + x_2 A_2). \tag{B.1}$$

B.3 Answers to Rehearsal Questions

The following questions might help in rehearsing the contents of the course:

1. What is an optimization problem? Objective, degrees of freedom, constraints. Feasible set? Standard form of NLP.

An optimization problem consists of the following three ingredients:

- An objective function $x \mapsto f(x) \in \mathbb{R}$ that shall be maximized or minimized.
- Some decision variables, typically lying in a vector space $\mathbb{R}^n$: $x \in \mathbb{R}^n$
- A feasible set $\Omega \subset \mathbb{R}^n$, typically defined by equality and inequality constraints:

$$\Omega = \left\{ x \in \mathbb{R}^n \;\middle|\; g(x) = 0 \text{ and } h(x) \geq 0 \right\} \tag{B.2}$$

The problem is then written as follows:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(B.3a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(B.3b)} \\ & h(x) \geq 0. && \text{(B.3c)} \end{aligned}$$

2. Definition of global and local minimum.

We say that $x^\star$ is a *global minimizer* when it is feasible, i.e. $x^\star \in \Omega$ and optimal, i.e. $\forall x \in \Omega : f(x) \geq f(x^\star)$.
We say that it is the *strict global minimizer* when $x^\star \in \Omega$ and $\forall x \in \Omega \setminus \{x^\star\} : f(x) > f(x^\star)$.
We say that it is a *local minimizer* when there exists a neighborhood $\mathcal{N}$ of $x^\star$ (e.g. an open ball around $x^\star$) such that it is optimal within that neighborhood, i.e. $\forall x \in \Omega \cap \mathcal{N} : f(x) \geq f(x^\star)$.

3. State a condition under which there exists at least one minimizers.

If the feasible set is *compact*, i.e. bounded and closed, and the objective function is continuous, then there exists at least one *global* minimizer (this is called the Theorem of Weierstrass)

4. Types of optimization problems: Linear / Quadratic programming (LP/QP), convex, smooth, mixed-integer, ...

- LP:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^\top x && \text{(B.4a)} \\ \text{subject to} \quad & Ax - b = 0, && \text{(B.4b)} \\ & Cx - d \geq 0. && \text{(B.4c)} \end{aligned}$$

- QP:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^\top x + \frac{1}{2} x^\top B x && \text{(B.5a)} \\ \text{subject to} \quad & Ax - b = 0, && \text{(B.5b)} \\ & Cx - d \geq 0. && \text{(B.5c)} \end{aligned}$$

- **Convex problem:** in the following form: $f(\cdot)$ a convex function, each component $h_j(\cdot)$ has to be concave, and $g(\cdot)$ has to be affine

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(B.6a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(B.6b)} \\ & h(x) \geq 0. && \text{(B.6c)} \end{aligned}$$

- **Smooth problem:** the functions $f(\cdot)$, $g(\cdot)$ and $h(\cdot)$ are continuously differentiable in the following problem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(B.7a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(B.7b)} \\ & h(x) \geq 0. && \text{(B.7c)} \end{aligned}$$

- **Mixed-integer program:** some of the variables are restricted to integers:

$$\begin{aligned} \underset{x \in \mathbb{R}^n,\, z \in \mathbb{Z}^{n'}}{\text{minimize}} \quad & f(x, z) && \text{(B.8a)} \\ \text{subject to} \quad & g(x, z) = 0, && \text{(B.8b)} \\ & h(x, z) \geq 0. && \text{(B.8c)} \end{aligned}$$

5. When is a function convex? Gives the definition and a characterization when it is twice differentiable. Definition. If it is twice differentiable?

A function $f : \Omega \to \mathbb{R}$ is *convex*,if Ω is convex and if:

$$\forall x, y \in \Omega, t \in [0,1]: \quad f(x + t(y - x)) \leq f(x) + t(f(y) - f(x)) \tag{B.9}$$

i.e. all secants are above graph.
When $f(\cdot)$ is *twice continuously differentiable* and Ω convex, then it is convex *if and only if* the Hessian is positive semi-definite at all point, i.e.

$$\forall x \in \Omega: \qquad \nabla^2 f(x) \succcurlyeq 0 \tag{B.10}$$

6. When is a set convex? Definition.

A set $\Omega \subset \mathbb{R}^n$ is convex,if:

$$\forall x, y \in \Omega, t \in [0,1]: \quad x + t(y - x) \in \Omega \tag{B.11}$$

i.e. all connecting lines lie inside set.

7. What is a "stationary" point in the the case of unconstrained smooth optimization?

In the case of unconstrained optimization a stationary point is a point $\bar{x}$ where the gradient is zero: $\nabla f(\bar{x}) = 0$.

8. How are gradient and Hessian of a function $f : \mathbb{R}^n \to \mathbb{R}$ defined?

The *gradient* of $f(\cdot)$ is defined to be the *vector field* whose components are the *partial derivatives* of f, i.e.:

$$\nabla f(x) \coloneqq \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \tag{B.12}$$

The *Hessian* of $f(\cdot)$ is the matrix containing the second derivatives of f:

$$\nabla^2 f(x) \coloneqq \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdot & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdot & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdot & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{B.13}$$

9. What are the first order necessary conditions for optimality (FONC) in the case of unconstrained optimization problems?

The theorem states that if $f(\cdot)$ is continuously differentiable, and $x^\star$ is a solution of the problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x). \tag{B.14}$$

Then $x^\star$ is a stationary point, i.e.

$$\nabla f(x^\star) = 0 \tag{B.15}$$

10. What are the second order necessary conditions for optimality (SONC) in the case of unconstrained optimization problems?

The theorem states that if $f(\cdot)$ is twice continuously differentiable, and $x^\star$ is a solution of the problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x). \tag{B.16}$$

Then $x^\star$ is a stationary point and the Hessian is positive semi-definite at this point, i.e.

$$\nabla f(x^\star) = 0, \nabla^2 f(x^\star) \succcurlyeq 0. \tag{B.17}$$

11. What are the second order sufficient conditions for optimality (SOSC) in the case of unconstrained optimization problems?

The theorem states that if $f(\cdot)$ is twice continuously differentiable, and $x^\star$ is a stationary point and the Hessian is positive definite at this point, i.e.

$$\nabla f(x^\star) = 0, \nabla^2 f(x^\star) \succ 0, \tag{B.18}$$

then $x^\star$ is a local minimizer to the problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x). \tag{B.19}$$

12. Basic idea of iterative descent methods?

It is an iterative algorithm where at iteration k, the iterate $x_k \in \mathbb{R}^n$ is the current best guess to the solution, and we look for the next iteration x_{k+1} to improve it. In optimization, we typically look for x_{k+1} such that a sufficient decrease of the value will be produced, which takes the form of $f(x_{k+1}) \leq f(x_k) - \ldots$.

13. Definition of local convergence rates: q/r-linear, superlinear, quadratic?

Let $(x_k)_{k\in\mathbb{N}}$ be a sequence in $\mathbb{R}^n$ that converges to $x^\star$. We say that the convergence is:

- **Q-linear:** if there is a constant $r \in (0,1)$ such that for k sufficiently large:

$$\|x_{k+1} - x^\star\| \leq r \|x_k - x^\star\|, \tag{B.20}$$

  e.g. $x^k = \frac{1}{2^k}$

- **R-linearly:** if it is upperbounded by another sequence itself convering Q-linearly:

$$y_{k+1} \leq r y_k, \tag{B.21a}$$
$$\|x_k - x^\star\| \leq y_k. \tag{B.21b}$$

- **Q-superlinear:** if there is a sequence r_k such that $r_k \to 0$ and:

$$\|x_{k+1} - x^\star\| \leq r_k \|x_k - x^\star\|, \tag{B.22}$$

  e.g. $x^k = \frac{1}{k!}$

- **Q-quadratic:** if there is a constant $M > 0$ such that:

$$\|x_{k+1} - x^\star\| \leq M \|x_k - x^\star\|^2. \tag{B.23}$$

14. What is a locally convergent algorithm? What is a globally convergent algorithm? What does the term "globalization" usually mean for optimizers?

An algorithm is locally convergent when it converges to a solution under the condition that it is initialized closed enough to a solution.
An algorithm is globally convergent when it converges to a solution no matter how it is initialized.
The term "globalization" is a component of an algorithm to ensure convergence even for bad initialization of the algotithm.

15. What is the Armijo condition? Why is it used in line-search algorithms?

The Armijo condition is a condition on the new iterate x_{k+1}, making sure that it induces a *sufficient decrease* of the objective function:

$$f(x_{k+1}) \leq f(x_k) + \gamma \nabla f(x_k)^\top (x_{k+1} - x_k). \tag{B.24}$$

Using this condition in a backtracking line-search algorithm often ensures global convergence of the optimization algorithm (given some condition on the chosen descent direction).

16. Why is satisfaction of Armijo condition alone not sufficient to guarantee convergence towards stationary points? Give a counterexample.

If one choses $x_{k+1} = x_k + t_k p_k$ with p_k being a descent direction, and the t_k a sequence that converges very quickly to zero, the algorithm might get stuck, even though the Armijo condition will be satisfied.
Example: $f(x) = x^2$ and $x_{k+1} = x_k - \frac{x_k}{(k+2)^2}$ satisfies the Armijo condition for k large enough, and yet it is does not converge to the solution:

$$x_k = \prod_{i=2}^{k} \left(1 - \frac{1}{i^2}\right) x_0, \tag{B.25a}$$

$$= \frac{1}{2}\frac{k+1}{k-1} x_0 \underset{k\to+\infty}{\longrightarrow} \frac{x_0}{2} \neq 0. \tag{B.25b}$$

17. What is backtracking?

In backtracking, the candidates for the next iterate is $x_{k+1} = x_k + t_k p_k$, and we shrink the step size $t_k \leftarrow \beta t_k$ with $\beta \in (0, 1)$ until a condition is satisfied.
Usually, the condition is the Armijo condition, we start at $t_k = 1$ and we choose $\beta = 0.8$.

18. Define the *the steepest descent method*. What is the local convergence rate?

The steepest descent method is the following iterative algorithm:

$$x_{k+1} = x_k - \alpha \nabla f(x_k), \tag{B.26}$$

with $\alpha > 0$ being the *learning rate*.
If the learning rate α is small enough to ensure convergence, the steepest descent method converges *Q-linearly* to a stationary point, i.e.:

$$\|x_{k+1} - x^\star\| \leq r \|x_k - x^\star\| \leq \cdots \leq r^{k+1} \|x_0 - x^\star\|, \tag{B.27}$$

where r is a constant in $(0, 1)$.

19. What is Newton's method for solution of nonlinear equations $F(x) = 0$? How does it iterate, what is the motivation for it. How does it converge locally?

It is a numerical method for finding a solution to some nonlinear equation $F(x) = 0$. It is an iterative method where at each step, we linearize the function $F(\cdot)$ at the current iterate x_k and solve the linearized equation. The iteration is given by:

$$x_{k+1} = x_k - \nabla F(x_k)^{-1} F(x_k). \tag{B.28}$$

This locally converges very fast for smooth functions, i.e. the convergence rate is *quadratic*, i.e.:

$$\|x_{k+1} - x^\star\| \leq M \|x_k - x^\star\|^2, \tag{B.29}$$

where M is a constant that depends on the function $F(\cdot)$ and the point $x^\star$.
A limitation is that the method is guranteed to converge *only if* the initial guess x_0 is close enough to some solution $x^\star$.

20. How works Newton's method for unconstrained optimization?

In Newton's method for unconstrained optimization, we apply Newton's method to the nonlinear root-finding equation:

$$F(x) \coloneqq \nabla f(x) = 0. \tag{B.30}$$

The resulting iteration is:

$$x_{k+1} = x_k - \nabla^2 f(x_k)^{-1} \nabla f(x_k). \tag{B.31}$$

21. What are *Newton type / approximate Newton* methods?

In Newton type methods, we apply tyhe same idea, but replace the Hessian $\nabla^2 f(x_k)$ by some approximation B_k of the Hessian, i.e.:

$$x_{k+1} = x_k - B_k^{-1} \nabla f(x_k), \tag{B.32}$$

where B_k replaces the Hessian $\nabla^2 f(x_k)$.

22. What is the idea behind Quasi-Newton methods?

In Quasi-Newton methods, the Hessian is approximated in such a way that the approximation and the exact Hessian match eventually:

$$B_k \underset{k \to +\infty}{\longrightarrow} \nabla^2 f(x_k). \tag{B.33}$$

This is useful because it ensures *superlinear* convergence, often almost as good as the exact Newton method.

23. What is the secant condition? How is it motivated?

The secant condition is a condition on the Hessian approximation that would justify the latest change in the gradient. More precisely, it is the following condition:

$$B_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k). \tag{B.34}$$

This formula can be retrieved by linearizing the gradient.

24. What is the BFGS formula? Under which condition does it preserve positive definiteness?

Write $s_k \coloneqq x_{k+1} - x_k$ the latest change in the solution point, and $y_k \coloneqq \nabla f(x_{k+1}) - \nabla f(x_k)$ the latest change in the gradient.
The BFGS update formula is given by:

$$B_{k+1} = B_k + \frac{y_k y_k^\top}{y_k^\top s_k} - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k}. \tag{B.35}$$

This preserves positive definiteness of the Hessian approximation B_k when $y_k^\top s_k > 0$.

25. Prove that the latter condition is necessary for any update formula satisfying the secant condition to stay positive definite.

Multiplying on both side of the secant condition by $s_k^\top$ gives:

$$0 < s_k^\top B_{k+1} s_k = s_k^\top y_k, \tag{B.36}$$

which proves that the condition was necessary.

26. What is a linear least squares problem (unconstrained)? What is a a nonlinear least squares problem (unconstrained)?

- **Linear least squares problem:**

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \|Ax - b\|^2 . \tag{B.37}$$

- **Nonlinear least squares problem:**

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \|F(x)\|^2 . \tag{B.38}$$

  where $F : \mathbb{R}^n \to \mathbb{R}^m$ is a nonlinear function.

27. How does the Gauss-Newton method iterate? When is it applicable?

Gauss-Newton method is an iterative method for solving nonlinear least squares problems. It iterates by solving the linear least squares problem induced by a linearization of the inner function $F(\cdot)$ at each iteration:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \|F(x_k) + \nabla F(x_k)(x - x_k)\|^2 . \tag{B.39}$$

28. When does the Gauss-Newton method perform well? What local convergence rate does it have?

In general, the Gauss-Newton method performs well when the nonlinear components of the function $F(\cdot)$ are small at the solution. In general, the convergence rate is *Q-linear*, but it might become *superlinear* when at the solution $x^\star$:

$$\text{for } j = 1, \ldots, m : \qquad \nabla^2 F_j(x^\star) = 0 \ \text{ or } F_j(x^\star) = 0. \tag{B.40}$$

29. Statistical motivation of least squares terms in estimation problems?

Assume that some measurements $\eta_j \in \mathbb{R}$ are available, and modeled as follows:

$$\eta_j \approx M_j(\bar{x}). \tag{B.41}$$

where the model $M_j : \mathbb{R}^n \to \mathbb{R}$ is a function that depends on some parameters $\bar{x} \in \mathbb{R}^n$. Then a good estimate of the parameters $\bar{x}$ is given by solving the least squares problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \sum_{j=1}^{m} (\eta_j - M_j(x))^2. \tag{B.42}$$

This is motivated statistically by assuming that the noise is normally distributed:

$$\eta_j = M_j(\bar{x}) + \varepsilon_j, \tag{B.43a}$$
$$\varepsilon_j \sim \mathcal{N}(0, \sigma^2). \tag{B.43b}$$

In this case, the maximum likelihood problem is equivalent to the least squares problem, i.e.:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad -\log p(\text{measurements} \mid x) = \sum_{j=1}^{m} \frac{1}{2\sigma^2} (\eta_j - M_j(\overline{x}))^2. \tag{B.44}$$

30. What are the differences and similarities between line search and trust region methods?

These two methods are for ensuring global convergence by preventing the algorithm from taking too large steps.
In trust-region algorithm, the next iterate is restricted to be in a *trust region* around the current iterate x_k, so the step is constrained *before* computing it.
In line-search algorithm, *after* computing a descent direction, the algorithm searches for the right size of step to take in this direction.

31. List two ways to compute derivatives with help of computers.

- Finite differences;
- Automatic differentiation;

32. What errors occur when computing derivatives with finite differences? Do you know a rule of thumb of how large to choose the perturbation?

In finite differences, for computing the derivative w.r.t. one direction, we use:

$$\nabla f(x)^\top p \approx \frac{f(x+\varepsilon p) - f(x)}{\varepsilon}. \tag{B.45}$$

If ε is too small the derivative will suffer from **numerical noise (round-off error)**. On the other hand, if ε is too large the **linearization error** will be dominant.
A good rule of thum is to use $t = \sqrt{\varepsilon_{\text{mach}}}$ which ensures an error on the scale of $\sqrt{\varepsilon_{\text{mach}}}$.
For example, if the computer computes $f(x)$ with a precision of $\pm 10^{-10}$ (i.e. 10 digits of accuracy), then the derivatives will be computed with an error of about $\pm 10^{-5}$ (i.e. 5 digits of accuracy.)

33. What is the idea behind Automatic Differentiation (AD)? What is its main advantage?

Automatic Differentiation takes a function as a list of elementary operations to computes, where each of the elementary operations has a known derivative. Then, it computes the derivative of the function by applying the chain rule to each elementary operation, and combining the results.
The advantage is that it computes derivatives with machine precision and that the computational cost scales nicely with the dimension of the function.

34. Can AD be applied to compute second order derivatives?

Yes, by applying twice the AD principle, second-derivatives are seamlessly computed.

35. There are two ways of AD. Describe briefly. What are the advantages / disadvantages of the two?

There are two modes of AD:

- **Forward mode:** Applying the chain rule from the input to the output. This can be done while computing the value of the function itself. The cost of this algorithm scales nicely with the number of outputs. Hence this is useful when the function has a small number of inputs and a large number of outputs.
- **Reverse/Backward mode:** Applying the chain rule from the output to the input. For this, the algorithm needs the value of all of the intermediate variables, so a forward sweep is needed first (i.e. what we do when we evaluate the function). The disadvantage is that one needs to save in memory all of the intermediate variables before computing the derivative. The advantage is that the computational cost of this algorithm scales nicely with the number of inputs. Hence this is useful when the function has a large number of inputs and a small number of outputs. Note that this is typically the case when one is required to compute the gradient of the objective function.

36. Write a nonlinear program (NLP) in its standard form. How is the lagrangian function defined?

The standard form of a nonlinear program (NLP) is typically:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(B.46a)} \\ \text{subject to} \quad & g(x) = 0, && \text{(B.46b)} \\ & h_j(x) \geq 0. && \text{(B.46c)} \end{aligned}$$

The Lagrangian function of this NLP is defined as:

$$\mathcal{L}(x, \lambda, \mu) = f(x) - \lambda^\top g(x) - \mu^\top h(x), \tag{B.47}$$

where λ is a vector with the same dimension as the number of equality constraints, and μ is a vector with the same dimension as the number of inequality constraints.

37. What is the constraint qualification (CQ)? What is the linear independence constraint qualification (LICQ) at some point $\bar{x}$?

CQ refers to some regularity conditions on the active constraints at some point $\bar{x}$. One of the most common CQ is the *linear independence constraint qualification* (LICQ), which states that the gradients of the active constraints at x are linearly independent, i.e. the vectors $\nabla g_j(\bar{x})$ (for all index j) and $\nabla h_k(\bar{x})$ (for indices k such that $h_k(\bar{x}) = 0$) are linearly independent.

38. What are the Karush-Kuhn-Tucker (KKT) conditions for optimality? Why is it useful?

The KKT conditions are necessary conditions for a point to be a minimizer when LICQ holds. More formally, if $x \in \mathbb{R}^n$ is such that LICQ holds, and is a local minimizer, then there exists some Lagrange multiplier vectors λ and μ,such the following conditions hold:

$$\begin{aligned} \nabla f(x) - \lambda^\top \nabla g(x) - \mu^\top \nabla h(x) &= 0, && \text{(B.48a)} \\ g(x) &= 0, && \text{(B.48b)} \\ \begin{cases} h(x) \geq 0, \\ \mu = 0, \end{cases} \quad \text{or} \quad & \begin{cases} h(x) = 0, \\ \mu \geq 0. \end{cases} && \text{(B.48c)} \end{aligned}$$

39. What are the first order necessary conditions for optimality (FONC) (constrained)?

If $x^\star$ is a local minimizer of the NLP and LICQ holds at $x^\star$, then there exists a $\lambda^\star \in \mathbb{R}^m$ and $\mu^\star \in \mathbb{R}^q$ such that:

$$\begin{aligned} \nabla f(x^\star) - \nabla g(x^\star)\lambda^\star - \nabla h(x^\star)\mu^\star &= 0, && \text{(B.49a)} \\ g(x^\star) &= 0, && \text{(B.49b)} \\ h(x^\star) &\geq 0, && \text{(B.49c)} \\ \mu^\star &\geq 0, && \text{(B.49d)} \\ \mu_i^\star h_i(x^\star) &= 0, \quad i = 1, \ldots, q. && \text{(B.49e)} \end{aligned}$$

40. What are the second order necessary conditions for optimality (SONC) (constrained)?

Regard $x^\star$ with LICQ. If $x^\star$ is a local minimizer of the NLP, then:

i. $\exists \lambda^\star, \mu^\star$ so that KKT conditions hold;

ii. $\forall p \in C(x^\star, \mu^\star)$ holds that $p^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star, \mu^\star) p \geq 0$.

41. What are the second order sufficient conditions for optimality (SOSC) (constrained)?

If $x^\star$ satisfies LICQ and:

i. $\exists \lambda^\star, \mu^\star$ so that KKT conditions hold;

ii. $\forall p \in C(x^\star, \mu^\star),\ p \neq 0$, holds that $p^\top \nabla_x^2 \mathcal{L}(x^\star, \lambda^\star, \mu^\star) p > 0$,

then $x^\star$ is a strict local minimizer.

42. What is the "active set of constraints"?

This is the set of indices of inequality constraints such that the inequality is asctually an equality. More precisely:

$$\mathbb{A}(x) \coloneqq \left\{ k \in \{, \dots, m\} \;\middle|\; h_k(x) = 0 \right\}. \tag{B.50}$$

43. Give a standardform of a QP.

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^\top x + \frac{1}{2} x^\top B x && \text{(B.51a)} \\ \text{subject to} \quad & Ax - b = 0, && \text{(B.51b)} \\ & Cx - d \geq 0. && \text{(B.51c)} \end{aligned}$$

44. When is a QP convex?

A QP is convex whenever its Hessian B is positive semi-definite, i.e. $B \succcurlyeq 0$.

45. What is the main idea of an active set strategy?

An active set strategy tries to identify the active set of constraints at each iteration such that thez problem that is solved is equivalent to an *equality constrained optimization problem* (instead of an *inequality constrained optimization problem*).

46. What is the main idea behind an SQP method (for inequality constrained problems)?

In SQP, at each iteration, the constaints are linearized, and the objective function is approximated by a quadratic function.

47. What is the L_1-penalty method for equality-constrained problems? Under which condition is it "exact", i.e. has the same local minima as the original NLP?

The L_1-penalty method approximates the constrained problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \tag{B.52a}$$

$$\text{subject to} \quad g(x) = 0, \tag{B.52b}$$

with:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) + \rho \left\| g(x) \right\|_1 \quad = f(x) + \rho \sum_{i=1}^{m} \left| g_i(x) \right|. \tag{B.53}$$

The approximation yields the exact solution whenever:

$$\rho \geq \left\| \lambda^\star \right\|_\infty = \max_{i=1,\ldots,m} \left| \lambda_i^\star \right|, \tag{B.54}$$

where $\lambda^\star$ is the vector of Lagrange multipliers of the original problem at the solution $x^\star$.

48. How works Newton's method for equality constrained optimization?

Newton's method for equality constrained optimization involves iteratively solving a system of equations derived from the first-order optimality conditions:

$$\nabla f(x_k) + \lambda_k^\top \nabla g(x_k) = 0, \tag{B.55a}$$

$$g(x_k) = 0 \tag{B.55b}$$

At each iteration, the method updates the current solution x_k together with the associated multipliers λ_k by solving the linearized version of the KKT conditions:

$$\begin{bmatrix} \nabla^2 f(x_k) - \lambda_k^\top \nabla^2 g(x_k) & \nabla g(x_k)^\top \\ \nabla g(x_k) & 0 \end{bmatrix} \begin{bmatrix} x - x_k \\ \lambda - \lambda_k \end{bmatrix} + \begin{bmatrix} \nabla f(x_k) + \lambda_k^\top \nabla g(x_k) \\ g(x_k) \end{bmatrix} = 0. \tag{B.56}$$

49. What local convergence rate does an SQP method with exact Hessian usually have?

It usually has a quadratic convergence rate under reasonable assumptions (i.e. LICQ, second order sufficient conditions).

50. What is the basic idea of interior point methods? Gives two views this class of method.

Interior point methods are method to solve problems with inequality constraints. They basically gives guideline to treat these inequalities in order to use tools from unconstrained or equality-constrained optimization. In this method, the iterates should always be feasible for the inequality constraints, and the inequalities have to be fulfilled in a strict sense.
There are two similar kinds of interior point methods (they are kind of equivalent, even though they produce different iterates):

- In the primal-dual interior point method, we modify slightly the KKT conditions to make them them less illed-conditionned, then we apply Newton's method to iteratively solve them. More precisely, in the KKT conditinos, the complementarity equations $\mu_j h_j(x) = 0$ are replaced with $\mu_j h_j(x) = \tau$ with $\tau > 0$ quite small. To ensure consistency of the method, the parameter τ decreases and converges to 0 (or to some tolerance value).
- In the logarithmic barrier interior point method: we add a barrier function $-\tau \log h(x)$ to the objective that pushes the solution to stay within the inequality constraints. More precisely, we solve the following problem using the Newton's method:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) - \tau \sum_{j=1}^{m} \log h_j(x) && \text{(B.57a)} \\ \text{subject to} \quad & g(x) = 0. && \text{(B.57b)} \end{aligned}$$

  The parameter $\tau > 0$ is chosen as above.

In both of the variations, the parameter $\tau > 0$ is chosen quite small. To ensure consistency of the method, the parameter τ decreases and converges to 0 (or to some tolerance value).

51. What is the Lagrangian dual function of a general NLP?

The Lagrangian dual function is defined as follows:

$$q(\lambda, \mu) \coloneqq \min_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu) = \min_{x \in \mathbb{R}^n} f(x) - \lambda^\top g(x) - \mu^\top h(x). \tag{B.58}$$

52. What is the dual problem of a general NLP?

$$\begin{aligned} \underset{\lambda \in \mathbb{R}^p, \mu \in \mathbb{R}^q}{\text{maximize}} \quad & q(\lambda, \mu) \quad = \min_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu) && \text{(B.59a)} \\ \text{subject to} \quad & \mu \geq 0. && \text{(B.59b)} \end{aligned}$$

53. What is weak duality? To which problems does it apply?

Weak duality always hold, and it states that the optimal value of the dual problem is a lower than the optimal value of the original (or primal) problem.

54. What is strong duality? Under which sufficient conditions does it apply?

Strong duality states that the optimal value of the dual problem is equal to the optimal value of the original (or primal) problem.
It holds for convex problems under Slater's condition, which states that there exists a feasible point for which the *nonlinear inequality constraints are inactive*.

55. What is a semidefinite program (SDP)? Give a standardform.

An SDP is an optimization problem where the constraints are affine are Linear Matrix Inequalities (LMI), and the objective is linear. More precisely, it takes the following form:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^\top x && \text{(B.60a)} \\ \text{subject to} \quad & Ax - b = 0, && \text{(B.60b)} \\ & B_0 + \sum_{i=1}^{n} B_i x_i \succcurlyeq 0. && \text{(B.60c)} \end{aligned}$$

56. How would you reformulate the following eigenvalue optimization problem into an SDP for A_0, A_1, A_2 three symmetric matrices?

$$\underset{x \in \mathbb{R}^2}{\text{minimize}} \quad \lambda_{\max}(A_0 + x_1 A_1 + x_2 A_2). \tag{B.61}$$

$$\begin{aligned} \underset{s \in \mathbb{R}, x \in \mathbb{R}^2}{\text{minimize}} \quad & s && \text{(B.62a)} \\ \text{subject to} \quad & A_0 + x_1 A_1 + x_2 A_2 \preccurlyeq s\mathbb{I}. && \text{(B.62b)} \end{aligned}$$

Appendix C

Some basics of mathematics

In this short appendix, we provide a few basic mathematical concepts that are used throughout the lectures. These are basic definitions and well-known properties/theorems that will be stated without proof. The goal is to make it easier for the students to quickly check mathematical properties that are used in the rest of the book. To learn and study the concepts mentionned here, we invite the students to read more detailed maths textbooks, or to follow an appropriate lecture.

Basics of linear algebra

Definition C.1: Euclidean norm
The norm $\|\cdot\|$ denotes the L2-norm, also called the *Euclidean norm*, defined as follows:

$$\forall x \in \mathbb{R}^n, \quad \|x\| \coloneqq \sqrt{x^\top x} = \sqrt{\sum_{i=1}^{n} x_i^2}. \tag{C.1}$$

Proposition C.1: The Cauchy-Schwarz inequality
If x and y are vectors in $\mathbb{R}^n$, then:

$$\left| x^\top y \right| \leq \|x\| \, \|y\| \, . \tag{C.2}$$

Definition C.2: Eigenvalues of a matrix
Let $A \in \mathbb{R}^{n \times n}$ be a square matrix. An eigenvalue of A is a scalar $\lambda \in \mathbb{C}$ such that there exists a nonzero vector $x \in \mathbb{C}^n$ satisfying:

$$Ax = \lambda x. \tag{C.3}$$

We write $\mathrm{sp}(A)$ as the set of eigenvalues of A.

Proposition C.2: Non-singular matrices
Let $A \in \mathbb{R}^{n\times n}$ be a square matrix. Then, A is invertible if and only if $0 \notin \mathrm{sp}(A)$, i.e., there is no nonzero vector $x \in \mathbb{R}^n$ such that $Ax = 0$.

Theorem C.1: Spectral theorem
Let $S \in \mathbb{R}^{n\times n}$ be a symmetric matrix (i.e., $S^\top = S$).
Then $\mathrm{sp}(S) \subset \mathbb{R}$, i.e., the eigenvalues of S are real.
Furthermore, S is orthogonally diagonalizable:

$$S = P\Lambda P^\top. \tag{C.4}$$

where $P \in \mathbb{R}^{n\times n}$ is an orthogonal matrix, i.e., such that $P^\top P = I$, and Λ is a diagonal matrix whose diagonal elements are the eigenvalues of S (possibly repeated).

Definition C.3: Positiveness of symmetric matrices
Let $S \in \mathbb{R}^{n\times n}$ be a symmetric matrix.
We say that S is positive semi-definite (resp. positive definite) if for all $x \in \mathbb{R}^n \setminus \{0\}$, $x^\top Sx \geq 0$ (resp. $x^\top Sx > 0$).
We denote this property by $S \succeq 0$ (resp. $S \succ 0$).

For $S_1, S_2 \in \mathbb{R}^{n\times n}$, two symmetric matrices, we write $S_1 \succcurlyeq S_2$ (resp. $S_1 \succ S_2$) when $S_1 - S_2 \succcurlyeq 0$ (resp. $S_1 - S_2 \succ 0$).

Proposition C.3: Positive-definite matrices are non-singular
Let $S \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix, i.e., $S \succ 0$.
Then it is invertible.

Proposition C.4: Characterization of positive symmetric matrices
Let $S \in \mathbb{R}^{n\times n}$ be a symmetric matrix. The two following properties are equivalent:

$$S \succcurlyeq 0 \text{ (resp. } S \succ 0), \tag{C.5a}$$
$$\forall \lambda \in \mathrm{sp}(S),\ \lambda \geq 0 \text{ (resp. } \lambda > 0). \tag{C.5b}$$

Proposition C.5: Inequalities for symmetric matrices
Let $S \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Let c_1, c_2 be two scalars. Then the three following properties are equivalent:

$$c_1 I_n \preccurlyeq S \preccurlyeq c_2 I_n, \tag{C.6a}$$
$$\forall x \in \mathbb{R}^n, \quad c_1 \|x\|^2 \leq x^\top Sx \leq c_2 \|x\|^2, \tag{C.6b}$$
$$\forall \lambda \in \mathrm{sp}(S), \qquad c_1 \leq \lambda \leq c_2. \tag{C.6c}$$

Definition C.4: Orthogonal sets
Let $E \subset \mathbb{R}^n$ be a subspace of $\mathbb{R}^n$. Then, we define the orthogonal of E as:

$$E^\perp = \{x \in \mathbb{R}^n \mid \forall y \in E,\ x^\top y = 0\}. \tag{C.7}$$

Note that $E^\perp$ is also a subspace of $\mathbb{R}^n$.

Proposition C.6: Inclusion of orthogonal sets
Let $E, F \subset \mathbb{R}^n$ be two subspaces of $\mathbb{R}^n$ such that $E \subset F$. Then $F^\perp \subset E^\perp$.

Proposition C.7: Orthogonal sets of images and kernels
Let $A \in \mathbb{R}^{n\times m}$ be a matrix. Then the following properties hold:

$$\mathrm{Im}(A)^\perp = \mathrm{Ker}(A^\top), \tag{C.8}$$
$$\mathrm{Ker}(A)^\perp = \mathrm{Im}(A^\top). \tag{C.9}$$

Basics of differential calculus

Definition C.5: Differentiability
Let $\Omega \subset \mathbb{R}^n$. Let $f : \Omega \to \mathbb{R}$ be a function.
We say that f is differentiable at $x \in \mathbb{R}^n$ if there exists a vector $\nabla f(x) \in \mathbb{R}^n$ such that:

$$\forall p \in \mathbb{R}^n, \quad \frac{f(x+\varepsilon p) - f(x)}{\varepsilon} \underset{\varepsilon\to 0}{\longrightarrow} \nabla f(x)^\top p. \tag{C.10}$$

The vector $\nabla f(x)$ is called the gradient of f at x.
We say that f is differentiable on Ω if it is differentiable at all points of Ω.
If $\nabla f(x)$ is a continuous function, we say that f is continuously differentiable.

Definition C.6: Twice differentiable functions
We say that f is twice differentiable if it is differentiable, and if there exists a matrix $\nabla^2 f(x) \in \mathbb{R}^{n\times n}$ such that:

$$\forall x \in \Omega,\ \forall p \in \mathbb{R}^n, \quad \frac{\nabla f(x+\varepsilon p) - \nabla f(x)}{\varepsilon} \underset{\varepsilon\to 0}{\longrightarrow} \nabla^2 f(x) p. \tag{C.11}$$

The matrix $\nabla^2 f(x)$ is called the Hessian of f at x.
We say that f is twice continuously differentiable if $\nabla^2 f(x)$ is continuous.

Theorem C.2: Schwarz's theorem
If f is twice continuously differentiable, then $\nabla^2 f(x)$ is symmetric for all $x \in \Omega$.

Proposition C.8: First order Taylor's approximation
Assume that f is continuously differentiable. Then the following formula holds:

$$f(x+p) = f(x) + \nabla f(x)^\top p + r(x,p), \tag{C.12}$$

where $r(x,p)$ is defined as follows:

$$r(x,p) = \left(\int_0^1 \nabla(\nabla f(x+sp) - \nabla f(x)) \mathrm{d}s \right)^\top p. \tag{C.13}$$

Proposition C.9: First-order Taylor's approximation with limits
The residual $r(x,p)$ satisfies $r(x,p) = o(\|p\|)$, which means:

$$\frac{r(x,p)}{\|p\|} \underset{p \to 0}{\longrightarrow} 0. \tag{C.14}$$

Proposition C.10: First order Taylor's approximation for a twice differentiable function
If f is twice continuously differentiable, then:

$$r(x,p) = p^\top \left(\int_0^1 s \nabla^2 f(x+sp) \mathrm{d}s \right) p. \tag{C.15}$$

Proposition C.11: Inequality for first order Taylor's approximation
If f is twice continuously differentiable, and:

$$\forall s \in [0,1], \quad \lambda_{\min} I_n \preccurlyeq \nabla^2 f(x+sp) \preccurlyeq \lambda_{\max} I_n, \tag{C.16}$$

then:

$$\lambda_{\min} \|d\|^2 \leq r(x,d) \leq \lambda_{\max} \|d\|^2 . \tag{C.17}$$

Basics of topology

Definition C.7: Neighborhoods
For $x \in \mathbb{R}^n$, we say that $\mathcal{N} \subset \mathbb{R}^n$ is a neighborhood of x if for some $\varepsilon > 0$:

$$\text{for all } y \in \mathbb{R}^n \text{ such that } \|x-y\| < \varepsilon \quad \text{ then } y \in \mathcal{N}. \tag{C.18}$$

Definition C.8: The interior of a set
We say that x is in the interior of Ω when it admits a neighborhood $\mathcal{N}$ such that $x \in \mathcal{N} \subset \Omega$.

Definition C.9: Open sets
We say that a set $\mathcal{O} \subset \mathbb{R}^n$ is open when it contains a neighborhood of every point, i.e., when it is its own interior.

Definition C.10: Closed set
We say that a set $\mathcal{C} \subset \mathbb{R}^n$ is closed when its complement $\mathbb{R}^n \setminus \mathcal{C}$ is open.

Remark
Interestingly, the empty set and $\mathbb{R}^n$ are both open and closed.

Proposition C.12: Characterization of closed-sets
A set $\mathcal{C}$ is closed if and only if when $(x_k)_{k\in\mathbb{N}}$ in $\mathcal{C}$ converges to $x \in \mathbb{R}^n$, then $x \in \mathcal{C}$.

Definition C.11: Compact sets
We say that $\mathcal{K} \subset \mathbb{R}^n$ is compact when it is closed and bounded, i.e.

$$\exists M > 0, \quad \forall x \in \mathcal{K}, \quad \|x\| \leq M. \tag{C.19}$$

Definition C.12: Limit points
Let $(x_k)_{k\in\mathbb{N}}$ be a sequence in $\mathbb{R}^n$. We say that $\bar{x} \in \mathbb{R}^n$ is a limit point of the sequence $(x_k)_{k\in\mathbb{N}}$ if there exists an increasing sequence of integers k_j such that $x_{k_j} \underset{j\to+\infty}{\longrightarrow} \bar{x}$.

Theorem C.3: Bolzano-Weierstrass theorem
If $\mathcal{K} \subset \mathbb{R}^n$ is a compact set and $(x_k)_{k\in\mathbb{N}}$ is a sequence in $\mathcal{K}$, then it has (at least) a limit point in $\mathcal{K}$.

Remark
The converse part is also true: a set is compact if and only if every sequence has at least a limit point. This is actually a valid definition of compactness for general vector spaces, which coincides with Definition C.11 only for finite-dimensional vector spaces, i.e., the definition of compactness adopted here holds only for finite dimensions.

Proposition C.13: Continuous functions preserve compactness
If $\mathcal{K} \subset \mathbb{R}^n$ is compact and $f : \mathcal{K} \to \mathbb{R}^m$ is continuous, then $f(\mathcal{K})$ is compact.

Bibliography

[1] Amir Beck. *Introduction to Nonlinear Optimization: Theory, Algorithms and Applications with MATLAB*. MOS-SIAM, 2014.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. University Press, Cambridge, 2004.

[3] A. Griewank and A. Walther. *Evaluating Derivatives*. SIAM, 2 edition, 2008.

[4] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.