Università
Ca' Foscari
Venezia

Dipartimento
di Economia

Quaderni di Didattica

Matteo Iacopini

Basics of Optimization Theory with Applications in MATLAB and R*

University of Venice - Ca' Foscari

Department of Economics

Basics of Optimization Theory with Applications in MATLAB and R*

Matteo Iacopini†

Spring, 2016

Abstract

This document is the result of a reorganization of lecture notes used by the author during the Teaching Assistantship of the Optimization course at the M.Sc. in Economics program at the University of Venice. It collects a series of results in Static and Dynamic Optimization, Differential and Difference Equations useful as a background for the main courses in Mathematics, Finance and Economics both at the M.Sc. and at the Ph.D. level. In addition, it offers a brief critical summary of the power of programming and computational tools for applied mathematics, with an overview of some useful functions of the softwares MATLAB and R. All the codes used are available upon request to the author.

*The author gratefully acknowledges professors Paola Ferretti and Paolo Pellizzari from Ca' Foscari University of Venice and professors Mauro Sodini, Riccardo Cambini, Alberto Cambini and Laura Martein from Univesity of Pisa for the teaching material which upon which part of this work is based.

†PhD Student in Economics, Department of Economics, Ca' Foscari University of Venice.
e-mail: matteo.iacopini@unive.it.

Contents

List of Figures

List of Tables

List of symbols

$\mathbb{R}^+$	set of non-negative real numbers
$\mathbb{R}^{++}$	set of strictly positive real numbers
$\mathbb{C}$	set of complex numbers
ι	imaginary number
$\wedge$	logic operator "and", all conditions true
$\vee$	logic operator "or", either condition true (or more than one)
x	real (or complex) scalar $x \in \mathbb{R}$ ($x \in \mathbb{C}$)
$\mathbf{x}$	vector of real (or complex) numbers $\mathbf{x} \in \mathbb{R}^n$ ($\mathbf{x} \in \mathbb{C}^n$)
$\mathbf{A}$	matrix of real (or complex) numbers $\mathbf{A} \in \mathbb{R}^{(m \times n)}$ ($\mathbf{A} \in \mathbb{C}^{(m \times n)}$)
$\|\|\mathbf{x}\|\|$	norm of vector $\mathbf{x}$
$(x,y) \in \mathbb{R}^2$	vector (couple of scalars) in $\mathbb{R}^2$
(a,b)	open and unbounded set $\{x \in \mathbb{R} : a < x < b\}$
$[a,b]$	compact (i.e. closed and bounded) set $\{x \in \mathbb{R} : a \leq x \leq b\}$
$f(\mathbf{x})$	scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$
$F(f_1, \ldots, f_m)$	functional $F : \mathcal{P} \to \mathcal{Q}$, for some sets of functions $\mathcal{P}$,$\mathcal{Q}$
$f'(x)$	first derivative of the function $f(x)$ with respect to (w.r.t.) its argument x
$f''(x)$	second derivative of the function $f(x)$ w.r.t. its argument x (twice)
$f^n(x)$	n-th derivative of the function $f(x)$ w.r.t. its argument x (n times)
$\dfrac{\partial f(\mathbf{x})}{\partial x_i}$	(first order) partial derivative of the function $f(\cdot)$ w.r.t. the variable x_i
$\dfrac{\partial^m f(\mathbf{x})}{\partial x_{i_1} \ldots \partial x_{i_m}}$	(m-th order) partial derivative of the function $f(\cdot)$ w.r.t. the variables $x_{i_1} \ldots x_{i_m}$
$\dot{x}$	first order derivative of $x(t)$ w.r.t. t, for continuous supported $x(t)$ (i.e. $t \in \mathbb{R}^+$)
$\ddot{x}$	second order derivative of $x(t)$ w.r.t. t (twice)
x_s	value of the function $x(t)$ at $t = s$, for discrete supported $x(t)$ (i.e. $t \in (\mathbb{N} \cup 0)$)
$\displaystyle\int_a^b f(\mathbf{x})\, \mathrm{d}x_i$	definite integral of $f(\mathbf{x})$ w.r.t. x_i over the interval $[a,b]$
$\det(\mathbf{A})$, $\|\mathbf{A}\|$	determinant of the matrix $\mathbf{A}$
$\mathrm{tr}(\mathbf{A})$	trace of the matrix $\mathbf{A}$
$\rho(\mathbf{A})$	rank of the matrix $\mathbf{A}$
$\mathbf{x}'$, $\mathbf{A}'$	transpose of vector or matrix
$\max\{f\}$	maximum value attained by f
$\arg\max\{f\}$	argument which maximizes f
$\{x_k\}$	sequence (or succession) of scalars $x_k \in S \, \forall \, k$

Chapter 1

Introduction

The purpose of this work is to give a brief overview to some basic concepts in Optimization Theory which represent the fundamental background for any advanced course in Applied Mathematics, Financial Mathematics and Economics.

The document is divided in three main parts: Chapter 2 offers a review of the definitions and results most useful for the next chapters, starting from a review of complex numbers in Section 2.1, then moving to topology (Section 2.2), calculus (Section 2.3) and linear algebra (Section 2.4). In Chapter 3 is provided the theoretical framework of each topic in detail, standard static optimization tools are presented in Section 3.1. Sections 3.2 and 3.2 are devoted to the study of differential and difference equations, which are the building block of the dynamic optimization theory developed in Section 3.4. In second instance, Chapter 6 presents in a concise way the main insights and critical points underneath the use of professional softwares and programming languages for solving mathematical problems (Section 6.1). Furthermore, some applications concerning the topics discussed in the previous chapter are developed, respectively, in Sections 6.3, 6.4, 6.5 and 6.6. Then, some subtle issues concerning programming are analysed in more detail in Section 6.2. We are going to present the languages MATLAB® and R, but it is important to stress that the results and observations preserve their validity in a much more general framework, as far as programming is concerned. As for the applications of the theoretical results, in Chapter 4 we discuss in detail and then solve step by step some exercises following the analytical approach, while the same is done in Chapter 6 for what concerns the computational approach. Chapter 5 lists further exercises, for which no solution is provided. Finally, Chapter 7 presents a summary of the most important findings and concludes.

Chapter 2

Theoretical Background

The purpose of this Chapter is to give a summary of the most important results in calculus (Section 2.3) and linear algebra (Section 2.4), as well as some other useful notions form topology (Section 2.2) and complex numbers (Section 2.1), with the aim of providing the basic tools necessary for the study of optimization theory.

The main references for this Chapter are: [1] and online resources[1], for what concerns complex numbers; [16], [14], [3] and [5] for calculus and linear algebra; [12] for calculus and topology.

2.1 Complex Numbers

The set of complex numbers is denoted by $\mathbb{C}$ and is a superset of the real numbers, i.e. $\mathbb{R} \subset \mathbb{C}$. The reason for this relation stands in the fact that complex numbers include all real numbers together with the imaginary number. The imaginary number ι has this name because it is not "real", but it is a trick adopted in order to carry out operations like computing the square root of a negative (real) number. To clarify this idea, see the next definitions.

Definition 2.1.1 *The **imaginary number** ι is the number such that: $\iota^{2n} = -1$ and $\iota^{2n+1} = -1 \quad \forall n \in \mathbb{N} \setminus \{0\}$.*

Definition 2.1.2 *A **complex number** $z \in \mathbb{C}$ is defined as: $z = a + \iota b$ with $a \in \mathbb{R}$, $b \in \mathbb{R}$ and being ι the imaginary number. The real number a is called **real part** of the complex number, while the real number b is called **imaginary part** of the complex number. In addition, $\forall z \in \mathbb{C}$ $\bar{z} = a - \iota b$ is the complex **conjugate** of the complex number z; hence $z = \bar{z}$ if and only if z is a real (and not complex) number.*

From definition 2.1.2 it is immediate to notice that any real number is also a complex number with null imaginary part (i.e. $b = 0$), hence the "value added" of this set is given by the imaginary part. The following lemma gives an overview of how to carry out some basic operations with complex numbers.

Lemma 2.1.1 *Let $z = a + \iota b$ and $w = c + \iota d$ be two complex numbers. Then:*

- $z = w \Leftrightarrow a = c \wedge b = d$
- $z + w = (a + c) + \iota(b + d)$
- $z \cdot w = (ac - bd) + \iota(ad + bc)$
- $\frac{1}{z} = \frac{a}{a^2+b^2} - \frac{b}{a^2+b^2}\iota$.

[1] See WikiBooks.

Differently from real numbers, which can be represented as points on a line (that is, the real line), complex numbers cannot be represented on a line, but on the plane (which is called **complex plane**[2]). In fact, be defining the horizontal axis as the real part (the component a) and the vertical axis as the imaginary part (the component b), it is possible to represent any complex number by the couple of real numbers $(a,b) \in \mathbb{R}$. According to this representation, the horizontal line has $b = 0$, hence it contains only real numbers (i.e. it is the real line), while any point outside the horizontal line in no more real, but complex in the strict sense.

There are additional ways to represent a complex number, one of the most useful is the polar form, which derives from trigonometry.

Definition 2.1.3 *A point $(a,b) \in \mathbb{R}$ on the Cartesian plane can be uniquely identified by its **polar coordinates** (ρ,θ), where:*

- *$\rho^2 = a^2 + b^2$ is the distance of the point from the origin $(0,0)$*
- *θ is the oriented angle formed by the positive abscissa halfline and by the helfline passing through the origin and the point (a,b). This angle (as any angle) is identified up to multiples of 2π. From trigonometry: $a = \rho\cos(\theta)$ and $b = \rho\sin(\theta)$*

Definition 2.1.4 *Let $z \in \mathbb{C}$. Then it holds:*

- *algebraic form: $z = a + \iota b$, with $(a,b) \in \mathbb{R}^2$*
- *polar form: $z = \rho(\cos(\theta) + \iota\sin(\theta))$. In addition, it holds that $|z| = \rho$, where $|\cdot|$ is the modulo.*

Notice that, since two complex numbers are equal if and only if both they real and imaginary parts are equal, when we use the polar form notation we have that it must hold $\rho_1 = \rho_2$ and $\theta_1 = \theta_2 + 2k\pi$ with $k \in \mathbb{R}$, since the trigonometric functions $\cos(\cdot)$ and $\sin(\cdot)$ are periodic with period (that is, they repeat themselves every period) equal to 2π. From the polar form representation it is possible to give a geometric interpretation to the operations of sum and multiplication between two complex numbers, which is exactly the same as those concerning bidimensional real vectors (in fact, a complex number can be represented on the complex plane).

The following Theorem is a fundamental one for complex calculus and it is known as "De Moivre's formula".

Theorem 2.1.2 *Let $z = \rho(\cos(\theta) + \iota\sin(\theta))$ and $n \in \mathbb{N} \setminus \{0\}$. Then:*

$$z^n = \rho^n(\cos(n\theta) + \iota\sin(n\theta))$$

In the field of complex numbers, there is a way to represent exponential functions which is very useful in differential calculus[3].

Definition 2.1.5 *Let $z = x + \iota y$ be a complex number and define the **exponential function** in the **complex field** as:*

$$e^{\iota y} = \cos(y) + \iota\sin(y) \tag{2.1}$$
$$e^z = e^{x+\iota y} = e^x e^{\iota y} = e^x(\cos(y) + \iota\sin(y)) \tag{2.2}$$

[2]The only difference between the complex plane and the real plane is the the former has one axis (the vertical one) which contains real values, but gives information about the imaginary part; instead the real plane has two axis of real numbers containing information only about the real part.

[3]For example, this representation is used to express the solutions of a linear differential equation with constant coefficients, or of linear systems of differential equations with constant coefficients.

It can be proved that:

Lemma 2.1.3 *Let* $(z_1,z_2) \in \mathbb{C}^2$ *be two complex numbers, then it holds:*

$$e^{z_1+z_2} = e^{z_1} e^{z_2}$$

From which it follows the relation: $e^{\iota\pi} = \cos(\pi) + \iota \sin(\pi) = -1$, which relates ι, π and 1.

The following theorems complete this brief introduction to complex numbers and are particularly useful in calculus; the first one is known as "d'Alembert-Gauss theorem".

Theorem 2.1.4 *Every polynomial with complex coefficients and degree* $n \geq 1$ *admits at least one complex root.*

Recalling that every real number is also a complex number, this theorem says that every polynomial has at least one root, which is in the complex field (that is, it may be complex in a strict sense or real).

Theorem 2.1.5 *Every polynomial with complex coefficients and degree* $n \geq 1$ *can be decomposed in the product of* n *factors of degree* 1.

Theorem 2.1.6 *The complex and not real roots of a polynomial with real coefficients are pairwise conjugate. Furthermore, if* α *is a root with multiplicity* $q \geq 1$ *then the conjugate of* α *(*$\bar{\alpha}$*) is a root with the same multiplicity* q*.*

Taken all together, these three theorems state that some operations (like factorization of polynomials) that were prevented in some cases in the real field, are instead always possible in the complex field. In addition, they state that the solutions of polynomial equations always exist in the complex field, enlarging the set of problems that admit a solution.

2.2 Topology

We begin this section concerning topology with the definition of Euclidean norm and Euclidean distance on vector spaces, since they are the building blocks for many subsequent results.

Definition 2.2.1 *The* ***Euclidean norm*** *of* $\mathbb{R}^n$ *is a function* $||\cdot|| : \mathbb{R}^n \to \mathbb{R}$ *defined as:*

$$\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

Properties:

- $\|\mathbf{x}\| \geq 0 \; \forall \mathbf{x} \in \mathbb{R}^n$ *and* $\|\mathbf{x}\| = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$
- $\|\lambda \mathbf{x}\| = |\lambda| \, \|\mathbf{x}\|$
- *triangular inequality:* $\|\mathbf{x} + \mathbf{y}\| \leq \|x\| + \|y\|$

Definition 2.2.2 *The* ***Euclidean distance*** *of* $\mathbb{R}^n$ *is a function* $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ *defined as:*

$$d(\mathbf{x},\mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| .$$

Properties:

- $d(\mathbf{x},\mathbf{y}) \geq 0 \; \forall (\mathbf{x},\mathbf{y})$ *and* $d(\mathbf{x},\mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$
- $d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x})$

- $d(\mathbf{x},\mathbf{z}) \leq d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{z})$

Now, we can start with the basic definitions of topology concerning open balls, open and closed sets. These are useful since when we want to optimize functions we must know the properties of the sets (domain and image set) on which the function is defined and gives values, respectively.

Definition 2.2.3 *Let $\mathbf{x_0} \in \mathbb{R}^n$. The **open ball** $B(\mathbf{x}_0,r)$ with center $\mathbf{x}_0$ and radius $r \in \mathbb{R}^{++}$ is given by:*

$$B(\mathbf{x}_0,\mathbf{r}) = \{\mathbf{x} \in \mathbb{R}^n : d(\mathbf{x}_0,\mathbf{x}) < r\}.$$

*A subset $S \subseteq \mathbb{R}^n$ is said to be **open** if:*

$$\forall \mathbf{x} \in S \ \exists r > 0 : B(\mathbf{x}_0,\mathbf{r}) \subset S.$$

*A subset $S \subseteq \mathbb{R}^n$ is said to be **closed** if its complement $S^C = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \notin S\}$.*

The following proposition lists some useful properties relating sets and sequences.

Proposition 2.2.1 *A set S is closed iff for all sequences $\{x_k\} : x_k \in S \ \forall k$ and $x_k \to x$ (i.e. the sequence converges to the element x), then $x \in S$. Moreover:*

- *let A, B open sets. Then $A \cup B$ and $A \cap B$ are open sets*
- *let A, B closed sets. Then $A \cup B$ and $A \cap B$ are closed sets*
- *let A_n a sequence of open sets. Then $\bigcup_n A_n$ is open*
- *let B_n a sequence of closed sets. Then $\bigcap_n B_n$ is closed*

Other concepts strictly related to openness and closeness and fundamental for calculus are those of neighbourhood of a point and frontier of a set, defined below.

Definition 2.2.4 *Given a set $S \subseteq \mathbb{R}^n$, then:*

- *if $\mathbf{x} \in S$ and $B(\mathbf{x},r) \subset S$, then S is called **neighbourhood** of $\mathbf{x}$*
- *$\mathbf{x} \in \mathbb{R}^n$ belongs to the **frontier** ∂S of S if each neighbourhood of $\mathbf{x}$ contains at least an element $\mathbf{y}_1 \in S$ and an element $\mathbf{y}_2 \notin S$*
- *$\bar{S} = S \cup \partial S$ is the **closure** of S. $\bar{S}$ is closed*
- *$int(S) = S \backslash \partial S$ is the **interior** of S and it is an open set*
- *$\mathbf{x} \in S$ belongs to the interior of S if $\exists r > 0 : B(\mathbf{x},r) \subset S$*

Definition 2.2.5 *A subset $S \subseteq \mathbb{R}^n$ is **bounded** when $\exists r > 0 : S \subset B(\mathbf{0},r)$*
*A subset $S \subseteq \mathbb{R}^n$ is **compact** when it is closed and bounded*

In the context of optimization it is important to stress that a continuous function maps compact sets into compact sets. Furthermore, a cornerstone of optimization theory, that is, the Weierstrass Theorem, relies on the assumption of compactness of the domain.

In the following is defined the convexity of a set.

Definition 2.2.6 *Given a collection of points $(\mathbf{x_1},\ldots,\mathbf{x_m})$ each one in $\mathbb{R}^n$, a point $\mathbf{z} \in \mathbb{R}^n$ is a **convex combination** of the points $(\mathbf{x_1},\ldots,\mathbf{x_m})$ if $\exists \lambda \in \mathbb{R}^m$ satisfying:*

$$\lambda_i \geq 0 \, \forall i \in [1,m] \qquad \text{and} \quad \sum_{i=1}^{m} \lambda_i = 1$$

such that: $\mathbf{z} = \sum_{i=1}^{m} \lambda_i \mathbf{x}_i$.

*A subset $S \subseteq \mathbb{R}^n$ is **convex** if it contains all the convex combinations of any two points (i.e. elements) in S*

When dealing with an optimization problem we are always interested in particular points, such as maxima and minima, therefore their general definition plays a central role in the following analysis. However, maximum and minimum points does not always exist, so it is necessary to provide some more general definitions.

Definition 2.2.7 *Let $A \subseteq \mathbb{R}$ be nonempty.*
*The set of **upper bounds** of A is defined as:*

$$U(A) = \{u \in \mathbb{R} : u \geq a \quad \forall\, a \in A\}$$

*If $U(A)$ is nonempty, then A is said to be **bounded above**.*
*The set of **lower bounds** of A is defined as:*

$$L(A) = \{l \in \mathbb{R} : l \leq a \quad \forall\, a \in A\}$$

*If $L(A)$ is nonempty, then A is said to be **bounded below**.*

Definition 2.2.8 *The **supremum** of A ($\sup(A)$) is the least upper bound of A.*

- *if $U(A)$ is nonempty, then*

$$\sup(A) = \{a^* \in U(A) : a^* \leq u \quad \forall\, u \in U(A)\}$$

- *if $U(A)$ is empty, then $\sup(A) = +\infty$*

*The **infimum** of A ($\inf(A)$) is the greatest lower bound of A.*

- *if $L(A)$ is nonempty, then*

$$\inf(A) = \{a^* \in L(A) : a^* \geq l \quad \forall\, l \in L(A)\}$$

- *if $L(A)$ is empty, then $\inf(A) = -\infty$*

Therefore the supremum and infimum of a set always exists, though they are finite only when the set of upper bounds and lower bounds, respectively, are nonempty. Finally, the definition of maximum and minimum point follows.

Definition 2.2.9 *Let $A \subseteq \mathbb{R}$ be nonempty.*
*The **maximum** of A ($\max(A)$) is defined as:*

$$\max(A) = \{x \in A : x \geq a \quad \forall\, a \in A\}$$

*The **minimum** of A ($\min(A)$) is defined as:*

$$\min(A) = \{y \in A : y \leq a \quad \forall\, a \in A\}$$

We need to stress that the maximum (minimum) exists iff $\sup(A) \in A$ ($\inf(A) \in A$), therefore they are more stringent definitions than those of supremum and infimum, which are always defined (though not always finite). Note that a maximum and minimum of a compact set always exists. Furthermore, the maximum (minimum) is defined as a set of points and not as a unique point. In the latter case we use the term **strict maximum** (**strict minimum**).

2.3 Calculus

This section is devoted to the presentation of the definitions and fundamental results in multivariate calculus. Given its importance, we begin by stating the notion of continuity.

Definition 2.3.1 *Let $f : S \to T$, where $S \subseteq \mathbb{R}^n$ and $T \subseteq \mathbb{R}^m$. Then f is **continuous at** $\mathbf{x} \in S$ if:*

$$\forall\, \epsilon > 0 \;\; \exists\, \delta > 0 \quad \textit{such that} \quad \mathbf{y} \in S \quad \textit{and} \quad d(\mathbf{x},\mathbf{y}) < \delta \quad \Rightarrow \quad d(f(\mathbf{x}),f(\mathbf{y})) < \epsilon.$$

*Moreover, f is **continuous** if it is continuous in each $\mathbf{x} \in S$.*

The concept of differentiability is a cornerstone of calculus and several definitions in the multivariate case are available. We provide the following two:

Definition 2.3.2 *A function $f : S \to \mathbb{R}^m$, $S \subseteq \mathbb{R}^n$ is **differentiable** at $\mathbf{x} \in S$ if there exists a linear map $L_x : S \to \mathbb{R}^m$ such that:*

$$\forall\, \epsilon > 0 \;\; \exists\, \delta > 0 \quad \textit{such that} \quad (\mathbf{x}+\mathbf{h}) \in S \;\; \textit{and} \;\; \|\mathbf{h}\| < \delta \quad \Rightarrow \quad \left\| f(\mathbf{x}+\mathbf{h}) - f(\mathbf{x}) - L_x(\mathbf{h}) \right\| < \epsilon \|\mathbf{h}\|.$$

Equivalently, if:

$$\lim_{\mathbf{h}\to\mathbf{0}} \frac{\left\| f(\mathbf{x}+\mathbf{h}) - f(\mathbf{x}) - L_x(\mathbf{h}) \right\|}{\|\mathbf{h}\|} = 0.$$

*The map L_x is called **differential** of f at $\mathbf{x}$ and denoted by $df(\mathbf{x})$. It can also be represented by a matrix J_x: $L_x(\mathbf{h}) = df(\mathbf{x})(\mathbf{h}) = J_x \cdot \mathbf{h}$.*

Moreover, if f is differentiable for all $\mathbf{x} \in S$, then it is differentiable on S.

A couple of remarks are useful: first, if $df(\mathbf{x})$ is continuous, then $f \in \mathcal{C}^1$. In general a function is $\mathcal{C}^n$ if it is continuous with continuous differentials up to order n. Second, if $f : S \to \mathbb{R}$ (i.e. when the output of the function is a scalar), then df is called **gradient** of f.

In the multivariate case we can define also the directional derivative, which is the derivative along a given direction, which can differ from the "canonical" ones represented by the axes. In fact, differentiating with respect to a variable (i.e. taking the partial derivative) is equivalent to differentiate along the direction of the corresponding axis. The following definition presents the new concept of derivative along an arbitrary direction (identified by a vector, so a straight line passing through the origin) and the subsequent theorem highlights the link between the directional derivative and the gradient vector.

Definition 2.3.3 *Let $f : \mathbb{R}^n \to \mathbb{R}$. The One sided **directional derivative** of f at $\mathbf{x}$ along the normalized direction $\mathbf{h}$ is:*

$$Df(\mathbf{x};\mathbf{h}) = \lim_{t\to 0} \frac{f(\mathbf{x}+t\mathbf{h}) - f(\mathbf{x})}{t}$$

when the limits exists.

Theorem 2.3.1 ([15, p. 94]) *Let $f : S \to \mathbb{R}$, with $S \subseteq \mathbb{R}^n$, be differentiable at $\mathbf{x}$. Then for any $\mathbf{h} \in \mathbb{R}^n$ the one sided directional derivative $Df(\mathbf{x};\mathbf{h})$ of f at $\mathbf{x}$ in the direction $\mathbf{h}$ exists and:*

$$Df(\mathbf{x};\mathbf{h}) = df(\mathbf{x}) \cdot \mathbf{h}.$$

Now we can show that the partial derivative is just a particular case of the more general directional derivative. In this case, in fact, we are fixing all the $n-1$ coordinates apart from the one along which we are differentiating (i.e. moving).

Definition 2.3.4 *Let $f : \mathcal{D} \to \mathbb{R}$, $\mathcal{D} \subseteq \mathbb{R}^n$, f differentiable on $\mathcal{D}$ (i.e. $f \in \mathcal{C}^1$). The $i-th$ **first order partial derivative** of f at $\mathbf{x}$ are defined as:*

$$\lim_{t\to 0} \frac{f(\mathbf{x}+t\mathbf{e}_i) - f(\mathbf{x})}{t} = \frac{\partial f}{\partial x_i}(\mathbf{x}) \qquad i = 1,\ldots,n$$

The introduce the concept of higher order partial derivatives:

Definition 2.3.5 *Let $f : \mathcal{D} \to \mathbb{R}$, $\mathcal{D} \subseteq \mathbb{R}^n$, f differentiable on $\mathcal{D}$, with differentiable partial derivatives up to order $n-1$ (i.e. $f \in \mathcal{C}^n$). The **n-th order partial derivatives** of f at $\mathbf{x}$ are defined as:*

$$d^n f(\mathbf{x}) = \frac{\partial^n f}{\partial x_1 \dots \partial x_n}$$

and it holds that:

$$\frac{\partial^n f}{\partial x_{i_1} \dots \partial x_{i_n}} = \frac{\partial^n f}{\partial x_{\Pi(i_1)} \dots \partial x_{\Pi(i_n)}}$$

for any permutation $\Pi(\cdot)$ of the indices.

The property stated above simply means that the order of derivation does not affect the final result, as long as we derive for the same variables overall. In the case $n = 2$ we obtain a matrix of second order partial derivatives, which is called **Hessian** matrix:

$$d^2 f(\mathbf{x}) = \mathcal{H} = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_n^2} \end{bmatrix} \tag{2.3}$$

From the above property we know that, if the function $f \in \mathcal{C}^2$, then the Hessian matrix is symmetric meaning that (pairwise) all cross derivatives are equal.

We now state some results concerning the concavity and convexity of functions with more than one variable. These are useful since many sufficient conditions for (global) optimality impose soma assumptions about the concavity (or convexity) of a function.

Remark 2.3.1 *There are many different results in this field, every one imposes certain conditions different from the others. One may read through all of these looking for those assumptions that are easiest to be matched in the practical problem at hand and use them to draw the desired conclusions. All the results are valid, but some paths may have straightforward implementation in some cases that others may not have, of course.*

We start from the definition of subgraph and epigraph of a function, then we formulate the well known conditions for a function to be concave or convex.

Definition 2.3.6 *Let $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$, $\mathcal{D}$ convex. Then:*

$$sub(f) = \{(\mathbf{x},\alpha) \in (\mathcal{D} \times \mathbb{R}) : f(\mathbf{x}) \geq \alpha\}$$
$$epi(f) = \{(\mathbf{x},\alpha) \in (\mathcal{D} \times \mathbb{R}) : f(\mathbf{x}) \leq \alpha\}$$

*are the **subgraph** and **epigraph** of f, respectively.*

Definition 2.3.7 *If $sub(f)$ is convex $\Rightarrow$ f is a **concave function** on $\mathcal{D}$*
*If $epi(f)$ is convex $\Rightarrow$ f is a **convex function** on $\mathcal{D}$*

Theorem 2.3.2 *f is concave on $\mathcal{D}$ iff:*

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \geq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \quad \forall\, \mathbf{x},\mathbf{y} \in \mathcal{D}, \quad \forall\, \lambda \in [0,1]$$

f is convex on $\mathcal{D}$ iff:

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \quad \forall\, \mathbf{x},\mathbf{y} \in \mathcal{D}, \quad \forall\, \lambda \in [0,1]$$

If the inequalities are strict, then f is strictly concave or strictly convex, respectively.

The geometrical interpretation of the previous theorem is quite simple: a function is said to be concave if, for any two points in its domain, any convex combination of the values the the function takes at these points is always not higher than the value assume by the function at the a convex combination of the two points (with the same weights). The opposite holds for a convex function. The next theorem instead is a fundamental one, since it highlights the strict relation between concave and convex functions. It is useful when one has to prove concavity/convexity of a linear combination of different (but known) functions as well as when one wants to switch from concave to convex functions (or vice versa).

Theorem 2.3.3 *f is (strictly) concave $\Leftrightarrow -f$ is (strictly) convex.*
f is (strictly) convex $\Leftrightarrow -f$ is (strictly) concave.

The next series of theorems provides a link between concavity and differentiability in the multivariate case.

Theorem 2.3.4 *Let $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ be concave on the open, convex set $\mathcal{D}$. Then:*

- *$Df(\mathbf{x},\mathbf{h})$ exists for all $\mathbf{x} \in \mathcal{D}$ and for all $\mathbf{h}$*
- *f is differentiable almost everywhere on $\mathcal{D}$*
- *the differential df is continuous (where it exists)*

Theorem 2.3.5 *Let $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ be differentiable on the open, convex set $\mathcal{D}$. It holds:*
f is concave on $\mathcal{D}$ iff:

$$df(\mathbf{x})(\mathbf{y}-\mathbf{x}) \geq f(\mathbf{y}) - f(\mathbf{x}) \quad \forall\, \mathbf{x},\mathbf{y} \in \mathcal{D}$$

f is convex on $\mathcal{D}$ iff:

$$df(\mathbf{x})(\mathbf{y}-\mathbf{x}) \leq f(\mathbf{y}) - f(\mathbf{x}) \quad \forall\, \mathbf{x},\mathbf{y} \in \mathcal{D}$$

Theorem 2.3.6 *Let $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ be differentiable on the open, convex set $\mathcal{D}$. It holds:*
f is concave on $\mathcal{D}$ iff:

$$(df(\mathbf{y}) - df(\mathbf{x}))(\mathbf{y}-\mathbf{x}) \leq \mathbf{0} \quad \forall\, \mathbf{x},\mathbf{y} \in \mathcal{D}$$

f is convex on $\mathcal{D}$ iff:

$$(df(\mathbf{y}) - df(\mathbf{x}))(\mathbf{y}-\mathbf{x}) \geq \mathbf{0} \quad \forall\, \mathbf{x},\mathbf{y} \in \mathcal{D}$$

The following results are very useful since they represent the theoretical basis for the study of the definiteness of the Hessian matrix in both unconstrained and constrained optimization problems.

Theorem 2.3.7 *Let $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ be of class $\mathcal{C}^2$ on the open, convex set $\mathcal{D}$. It holds:*

- *f is concave on $\mathcal{D} \Leftrightarrow d^2 f(\mathbf{x})$ is a negative semidefinite matrix $\forall\, \mathbf{x} \in \mathcal{D}$*
- *f is convex on $\mathcal{D} \Leftrightarrow d^2 f(\mathbf{x})$ is a positive semidefinite matrix $\forall\, \mathbf{x} \in \mathcal{D}$*
- *f is strictly concave on $\mathcal{D} \Leftrightarrow d^2 f(\mathbf{x})$ is a negative definite matrix $\forall\, \mathbf{x} \in \mathcal{D}$*
- *f is strictly convex on $\mathcal{D} \Leftrightarrow d^2 f(\mathbf{x})$ is a positive definite matrix $\forall\, \mathbf{x} \in \mathcal{D}$*

Theorem 2.3.8 *Any local maximum (minimum) of a concave (convex) function is a global maximum (minimum) of the function.*

The following theorem uses concavity and convexity for characterizing the set of optima, providing also a condition for the existence of a unique optimum.

Theorem 2.3.9 *If f is concave (strictly concave) on $\mathcal{D} \Rightarrow$ the set $\arg\max\{f(\mathbf{x}) : \mathbf{x} \in \mathcal{D}\}$ is either empty or convex (or contains a single point).*
If f is convex (strictly convex) on $\mathcal{D} \Rightarrow$ the set $\arg\min\{f(\mathbf{x}) : \mathbf{x} \in \mathcal{D}\}$ is either empty or convex (or contains a single point).

2.4 Linear Algebra

In the following the notation $\mathbf{A} \in M(m,n)$ will be used to define a generic $m \times n$ matrix.

Definition 2.4.1 *A **quadratic form** on $\mathbb{R}^n$ is a function $g_A : \mathbb{R}^n \to \mathbb{R}$ such that:*

$$g_a(x) = \sum_{i,j=1}^{n} a_{ij} x_i x_j = \mathbf{x}'\mathbf{A}\mathbf{x}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a symmetric matrix and $\mathbf{x} \in \mathbb{R}^n$. We call $\mathbf{A}$ a quadratic form itself.

Definition 2.4.2 *A quadratic form (i.e. a symmetric matrix) $\mathbf{A}$ is said to be:*

- ***positive definite** if $\mathbf{x}'\mathbf{A}\mathbf{x} > 0 \quad \forall \mathbf{x} \in \mathbb{R}^n \backslash \{\mathbf{0}\}$*
- ***positive semidefinite** if $\mathbf{x}'\mathbf{A}\mathbf{x} \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n \backslash \{\mathbf{0}\}$*
- ***negative definite** if $\mathbf{x}'\mathbf{A}\mathbf{x} < 0 \quad \forall \mathbf{x} \in \mathbb{R}^n \backslash \{\mathbf{0}\}$*
- ***negative semidefinite** if $\mathbf{x}'\mathbf{A}\mathbf{x} \leq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n \backslash \{\mathbf{0}\}$*
- ***indefinite** if $\exists \mathbf{x}_1 \in \mathbb{R}^n$ and $\mathbf{x}_2 \in \mathbb{R}^n$, such that $\mathbf{x}_1'\mathbf{A}\mathbf{x}_1 > 0$ and $\mathbf{x}_2'\mathbf{A}\mathbf{x}_2 < 0$*

Definition 2.4.3 *Given a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, the scalar $r \in \mathbb{C}$ is called **eigenvalue** of $\mathbf{A}$ if it satisfies:*

$$\mathbf{A}\mathbf{v} = r\mathbf{v}.$$

*In this case, r is called the **eigenvalue** associated to the **eigenvector** $\mathbf{v} \in \mathbb{R}^n$. The definition requires that the scalar r is a solution of the **characteristic equation**:*

$$|\mathbf{A} - r\mathbf{I_n}| = 0.$$

Proposition 2.4.1 *Properties:*

- *the characteristic equation is a polynomial of degree n*
- *a symmetric matrix admits only real eigenvalues*

Theorem 2.4.2 ([15, p. 23]) *If $\mathbf{A}$ is a $n \times n$ matrix with eigenvalues $r_1, \ldots, r_n$, then:*

- $|\mathbf{A}| = \prod_{i=1}^{n} r_i$
- $\mathrm{tr}(\mathbf{A}) = \sum_{i=1}^{n} r_i$

Definition 2.4.4 *Given $\mathbf{A} \in M(n,n)$, symmetric, we define:*

- *A_k the **leading principal minor** of order $k \leq n$, as the minor (i.e. square submatrix) of order k obtained by cutting the **last** (starting from south-east) $n-k$ rows and columns of $\mathbf{A}$*
- *$\mathcal{A}_k$ the **principal minor** of order k, as the minor of order k obtained by cutting $n-k$ rows and the corresponding columns of $\mathbf{A}$*

The following theorems state the relation between the definiteness of a symmetric square matrix, its principal minors and eigenvalues.

Theorem 2.4.3 ([15, p. 32]) *Let $\mathbf{A} \in M(n,n)$, symmetric. Then:*

- **A** *(g_A) is positive definite* $\Leftrightarrow A_k > 0 \quad \forall\, k \in [1,n]$
- **A** *(g_A) is positive semidefinite* $\Leftrightarrow \mathcal{A}_k \geq 0 \quad \forall\, k \in [1,n]$
- **A** *(g_A) is negative definite* $\Leftrightarrow (-1)^k A_k > 0 \quad \forall\, k \in [1,n]$
- **A** *(g_A) is negative semidefinite* $\Leftrightarrow (-1)^k \mathcal{A}_k \geq 0 \quad \forall\, k \in [1,n]$

Theorem 2.4.4 ([15, p. 33]) *Let* $\mathbf{A} \in M(n,n)$*, symmetric and let* $r_1, \ldots, r_n$ *be its eigenvalues. Then:*

- **A** *(g_A) is positive definite* $\Leftrightarrow r_k > 0 \quad \forall\, k \in [1,n]$
- **A** *(g_A) is positive semidefinite* $\Leftrightarrow r_k \geq 0 \quad \forall\, k \in [1,n]$
- **A** *(g_A) is negative definite* $\Leftrightarrow r_k < 0 \quad \forall\, k \in [1,n]$
- **A** *(g_A) is negative semidefinite* $\Leftrightarrow r_k \leq 0 \quad \forall\, k \in [1,n]$
- **A** *(g_A) is indefinite* $\Leftrightarrow$ **A** *has both positive and negative eigenvalues*

Chapter 3

The Analytical Approach in Optimization

This Chapter covers a wide variety of topics from a theoretical perspective: Section 3.1 presents the theory of static optimization, which is a cornerstone of mathematical analysis, then we move to two different topics which provide the necessary background for the discussion of dynamic optimization, which is presented in Section 3.4. In fact, Section 3.2 and Section 3.3 discuss the basic theory of ordinary differential equations and difference equations, respectively.

The main references for this Chapter are: [15], [12], [14], [3] and [5] for static optimization; [15] and [12] for differential equations; [15] for difference equations; [10] for calculus of variations; [7] for dynamic optimization.

3.1 Static Optimization

In this section we define the general framework of an optimization problem, providing some definitions that are valid both in unconstrained and in the constrained optimization case. In Section 2.2 we defined the concept of maximum and minimum for a generic set; now we provide a definition for the same concept considering functions instead of sets. For the rest of the section, if not specified differently, we will use $\mathcal{D} \subseteq \mathbb{R}^n$ to define the domain of a generic function $f : \mathcal{D} \to \mathbb{R}$.

The most general optimization problem is formulated as the search for maxima and minima of a given function f (which is called *objective function*) over a particular set $\mathcal{D}$ (the *feasible set*), which may correspond to the domain of f, in which case we talk about uncontrained optimization, or may be a subset of it, so we use term constrained optimization. In the following we are going to use always the symbol $\mathcal{D}$ for denoting the feasible set, the distinction between constrained and unconstrained optimization being clear in each case.

Definition 3.1.1 *Let $\mathbf{x}^* \in \mathcal{D}$. If $f(\mathbf{x}^*) \geq f(\mathbf{y})\ \forall \mathbf{y} \in \mathcal{D}$ then $\mathbf{x}^*$ is a **global maximum point** (global maximizer) for f on $\mathcal{D}$ and $f(\mathbf{x}^*)$ is a **global maximum** for f on $\mathcal{D}$.*

*Let $\mathbf{x}^* \in \mathcal{D}$. If $f(\mathbf{x}^*) \leq f(\mathbf{y})\ \forall \mathbf{y} \in \mathcal{D}$ then $\mathbf{x}^*$ is a **global minimum point** (global minimizer) for f on $\mathcal{D}$ and $f(\mathbf{x}^*)$ is a **global minimum** for f on $\mathcal{D}$.*

Global maximum (minimum) points are characterized by the fact that the function never takes higher (lower) values at any point of the feasible set: this is a stringent condition, rarely satisfied. A looser concept requires the inequality to hold only in a neighbourhood (topologically, in an open ball) of the optimum point. This is the concept of local optimum, stated below.

Definition 3.1.2 *Let $\mathbf{x}^* \in \mathcal{D}$. If $\exists r > 0 : f(\mathbf{x}^*) \geq f(\mathbf{y})\ \forall \mathbf{y} \in B(\mathbf{x}^*,r) \cap \mathcal{D}$ then $\mathbf{x}^*$ is a **local maximum point** (local maximizer) for f on $\mathcal{D}$ and $f(\mathbf{x}^*)$ is a **local maximum** for f on $\mathcal{D}$.*

*Let $\mathbf{x}^* \in \mathcal{D}$. If $\exists r > 0 : f(\mathbf{x}^*) \leq f(\mathbf{y})\ \forall \mathbf{y} \in B(\mathbf{x}^*,r) \cap \mathcal{D}$ then $\mathbf{x}^*$ is a **local minimum point** (local minimizer) for f on $\mathcal{D}$ and $f(\mathbf{x}^*)$ is a **local minimum** for f on $\mathcal{D}$.*

Problem	Optimizer	Value of the function
$\max\{f(\mathbf{x})\}$	$\mathbf{x}^*$	$f(\mathbf{x}^*)$
$\min\{-f(\mathbf{x})\}$	$\mathbf{x}^*$	$-f(\mathbf{x}^*)$

Table 3.1: Relations for optimization problems.

A general class of problems which we are interested in can be formulated in **parametric form**. This involves expressing the objective function and/or the feasible set depend on the value of one or more parameters $\theta \in \Theta$ (validity is maintained with min):

$$\max\{f(\mathbf{x},\theta) : \mathbf{x} \in \mathcal{D}(\theta)\}. \tag{3.1}$$

It is worthwhile to stress that, for any optimization problem, the maximization and minimization tasks are strictly related (see Table (3.1)), hence it is possible to move from one to other one when it is convenient.

Now that the general problem has been described, we may define the main steps to be undertaken in order to solve it:

1. identify the set of conditions on f and $\mathcal{D}$ under which the existence of a solution is guaranteed
2. obtain a characterization of optimal points:
   - necessary conditions for an optimum point
   - sufficient conditions for an optimum point
   - conditions for uniqueness of solutions
   - study parametric variation of the optima

Concerning the first point, that is, the existence of solutions (i.e. optimal points), one of the most important result both in the univariate and multivariate case is the Weierstrass Theorem.

Theorem 3.1.1 (Weierstrass) *Let $\mathcal{D} \subseteq \mathbb{R}^n$ be compact and $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous function on $\mathcal{D}$. Then f attains a maximum and a minimum on $\mathcal{D}$.*

Note that this theorem hinges on two stringent conditions: compact feasible set and continuity of the objective function, which are not always satisfied in practical problems. Therefore in specific applications one may rely on different theorems or start the search for optimal points without any guarantee about their existence.

The necessary, sufficient and uniqueness conditions at the second point of the procedure above strictly depend on the nature of the problem at hand, therefore we are going to present some of the most useful results for each of the cases we consider.

3.1.1 Unconstrained Optimization

The concept of unconstrained optimality, as previously defined, means that either no constraints are imposed to the problem at hand or that those imposed are not effective (i.e. they are not active or irrelevant). Notice that this does not mean that there are no constraints imposed on the domain itself, but that all of them are not binding at the optimal point. In practice, this may give rise to three different cases:

1) $\mathcal{D} = \mathbb{R}^n$: there are no constraints ant all

2) $\mathcal{D} \subset \mathbb{R}^n$ and open

3) $\mathcal{D} \subset \mathbb{R}^n$ not necessarily open, but we are looking for optima in the interior of $\mathcal{D}$ (which is an open set, see Section 2.2)

Notice that in this framework the Weierstrass Theorem does not apply, irrespective of the characteristics of the objective function, because the feasible set is open, hence not compact. As a consequence we have no guarantee that one or more optimal points exist. The procedure we are going to apply in this setting consists of two main steps involving the search for points satisfying necessary and sufficient conditions and is described schematically in Algorithm (1).

Algorithm 1 Solution of Unconstrained Optimization problem

```
procedure OPTIMIZATION(f,D)
    x^N ← points that satisfy necessary conditions
    if x^N = ∅ then
        return No solution
    else
        if some points x^N among satisfy sufficient conditions then
            x^S ← points that satisfy sufficient conditions, among x^N
            return Solution: x^S
        else
            return Solution (only candidate): x^N
        end if
    end if
end procedure
```

Let now define the necessary and sufficient conditions for the unconstrained optimization case, paying attention to the fact that the nomenclature "first/second order" arises from the use of first or second order derivatives (in this context, of the objective function), respectively.

Theorem 3.1.2 (First Order Necessary Conditions (FOC)) *Suppose* $\mathbf{x}^* \in \int(\mathcal{D})$ *and* f *is differentiable at* $\mathbf{x}^*$*. If* $\mathbf{x}^*$ *is a local maximum (or local minimum) for* f *on* $\mathcal{D}$ *then:*

$$df(\mathbf{x}^*) = \mathbf{0}.$$

We call **critical points** all the points at which the gradient is null. A couple of remarks are required: first, this theorem does not allow to distinguish between maximum and minimum points, but it gives the same requirement in both cases; second, there may be more than one critical point; finally, since these conditions are only necessary, there is no guarantee that any of the critical points is indeed a (local or global) maximum or minimum.

We now state another necessary condition, which hinges on the second order derivatives. Remember that these are still necessary conditions, so they do not provide any guarantee that the candidate critical point $\mathbf{x}^*$ is indeed a maximum nor a minimum.

Theorem 3.1.3 (Second Order Necessary Conditions) *Suppose* $\mathbf{x}^* \in int(\mathcal{D})$ *and* $f \in \mathcal{C}^2$ *on* $\mathcal{D}$*. It holds:*

- *if* f *has a local maximum at* $\mathbf{x}^* \Rightarrow d^2 f(\mathbf{x}^*)$ *is negative semidefinite*
- *if* f *has a local minimum at* $\mathbf{x}^* \Rightarrow d^2 f(\mathbf{x}^*)$ *is positive semidefinite*

We are now in the position to state the sufficient conditions for an optimum: they rely on the second order derivatives and are require more stringent constraints on the Hessian matrix, which must be definite not just semidefinite.

Theorem 3.1.4 (Second Order Sufficient Conditions (SOC)) *Suppose* $\mathbf{x}^* \in int(\mathcal{D})$ *and* $f \in \mathcal{C}^2$ *on* $\mathcal{D}$*. It holds:*

- *if* $df(\mathbf{x}^*) = 0$ *and* $d^2f(\mathbf{x}^*)$ *is negative definite* $\Rightarrow \mathbf{x}^*$ *is a (strict) local maximum of* f *on* $\mathcal{D}$
- *if* $df(\mathbf{x}^*) = 0$ *and* $d^2f(\mathbf{x}^*)$ *is positive definite* $\Rightarrow \mathbf{x}^*$ *is a (strict) local minimum of* f *on* $\mathcal{D}$

Notice that this theorem follows directly from the more general result stated below.

Theorem 3.1.5 $\mathbf{x}^*$ *is an interior maximum of* f*, which is concave and differentiable on* $\mathcal{D} \Leftrightarrow df(\mathbf{x}^*) = 0$.
$\mathbf{x}^*$ *is an interior minimum of* f*, which is convex and differentiable on* $\mathcal{D} \Leftrightarrow df(\mathbf{x}^*) = 0$

3.1.2 Constrained Optimization

Many times we are interested in looking for optimum points not on the whole domain of the objective function, but only on a limited part (which may be finite or infinite) of it. In this case we are imposing some restrictions on the domain, that is we are preventing the objective function to assume some specific values. The resulting feasible set is a subset of the domain, but it may also coincide with it. In the latter case the constraints imposed are not binding, that is they are "not active", in words it is "like they were not existing" and we are back in the free (unconstrained) optimization case.

It is intuitively clear that the optimal value that a given function may attain on its domain is always not worse than the optimal value attainable on whatsoever feasible set defined by one or more constraints. This can provide a useful tool for checking the correctness of the computations when moving from unconstrained to constrained problems (or vice versa) with the same objective function.

Example 3.1.1 *Consider the objective function* $f(x) = \ln(x)$*, whose domain is* $\mathcal{D} = \mathbb{R}^{++}$*, and impose the constraint* $G = \{x \in \mathbb{R} : x > -5\}$*. It is straightforward to see that this constraint is irrelevant for this problem, since it holds:* $\mathcal{D} \subset G$*, that is all values in the domain of the objective function belong to (i.e. satisfy) the "constrained set"* G*. Therefore we are exactly in the same case of free optimization.*

■

Example 3.1.1 (Cont'd) *Consider the same function in the previous example, but impose the constraint* $G = \{x \in \mathbb{R} : x > 3\}$*. In this case it holds* $G \subset \mathcal{D}$*, since all the values* (0,3] $\in \mathcal{D}$ *but* (0,3] $\notin G$*. As a consequence all the values that the function* f *may attain in this region in the case of unconstrained optimization are ruled out imposing the constraint given by* G*.*

■

3.1.2.1 Equality Constraints

Consider now the formal definition of a constrained optimization problem with equality constraints (again, max and min can be interchanged):

$$\begin{cases} \max\limits_{(x_1,\ldots,x_n)\in\mathcal{U}} f(x_1,\ldots,x_n) \\ s.t. \\ \quad g_1(x_1,\ldots,x_n) = b_1 \\ \quad \vdots \\ \quad g_m(x_1,\ldots,x_n) = b_m \end{cases} \tag{3.2}$$

where $\mathcal{U} \subseteq \mathbb{R}^n$ is the domain of the function f and is open; $g : \mathbb{R}^n \to \mathbb{R}^m$, $\mathbf{b} \in \mathbb{R}^m$.

The vector function g may be linear or nonlinear. The numbers n and m in practice are the number of the variables and that of the constraints imposed in the problem, respectively. The first order necessary conditions for this kind of problems are similar those for the unconstrained case, but

apply to a modified function, not directly to the objective function f. This new function is called Lagrangian and is defined as $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$:

$$\mathcal{L}(x_1, \ldots, x_n; \lambda_1, \ldots, \lambda_m) = f(x_1, \ldots, x_n) + \sum_{i=1}^{m} \lambda_i g_i(x_1, \ldots, x_n) \tag{3.3}$$

where f is the objective function, g is the vectorial function identifying the constraints and $\mathbf{b}$ is a vector of coefficients specifying the value of each constraint. In the equality constraint case it is possible to use either the plus or minus sign between the objective and the constraint functions. We can state the main Theorem now.

Theorem 3.1.6 (Lagrange) *Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ be $\mathcal{C}^1$ functions. Suppose that $\mathbf{x}^*$ is a local maximum (minimum) of f on:*

$$\mathcal{D} = \mathcal{U} \cap \{\mathbf{x} \in \mathbb{R}^n : g_j(\mathbf{x}) = \mathbf{b},\ j = 1, \ldots, m\},$$

where $\mathcal{U} \subseteq \mathbb{R}^n$ is open. Suppose (constraint qualification conditions (CQC)) that $\rho\left(dg(\mathbf{x}^*)\right) = m$.

Then there exists a vector $\lambda^* \in \mathbb{R}^m$ (Lagrange multipliers) such that:

$$df(\mathbf{x}^*) - \sum_{j=1}^{m} \lambda_j^* dg_j(\mathbf{x}^*) = \mathbf{0}.$$

For this theorem to hold it is necessary that the candidate point $\mathbf{x}^*$ verifies two conditions:

- it is a critical point of the Lagrangian function
- it satisfies the constraint qualification conditions

Remark 3.1.1 *In presence of equality constraints the feasible set is always closed, but may be bounded or not*[1]*. Therefore, when the feasible region is also bounded and the objective function is continuous, since the assumptions of the Weierstrass Theorem are satisfied, we have the certainty of the existence of at least one minimum and one maximum point.*

The following proposition provides an interpretation of the Lagrange multipliers, which in many economic applications represent the "shadow price": in words, they represent the marginal effect on the objective function of releasing the constraint.

Proposition 3.1.7 *Let $\mathbf{x}^*$ be a strict local optimum for the problem:*

$$\begin{cases} \max\limits_{x \in \mathcal{D}} f(\mathbf{x}) \\ s.t. \\ g(\mathbf{x}) = \mathbf{b} \end{cases}$$

If f and g are of class $\mathcal{C}^2$ and $dg(\mathbf{x}^*)$ is of full rank, then the value function

$$V(\mathbf{b}) = \sup\{f(\mathbf{x}) : g(\mathbf{x}) = \mathbf{b}\}$$

Is differentiable and:

$$\frac{\mathrm{d}V(\mathbf{b})}{\mathrm{d}\mathbf{b}} = \lambda^*.$$

where $\lambda+^*$ is the Lagrange multiplier associated to $\mathbf{x}^*$.

[1] Closeness follows from the fact that all constraints are satisfied with equality, while boundedness may be ruled out, for example, when the only restriction is represented by a straight line (or plane or hyperplane in higher dimensions. In this case there are no limits to the values that can be taken by the function along this line, since the region in unbounded.

As for the unconstrained case, we are in the need for sufficient conditions that grants the optimality of a particular point. Again, these require the computation of the second order partial derivatives, which are arranged in a modified version of the Hessian matrix:

$$\mathcal{H}(\mathbf{x}^*,\lambda^*) = \left[\begin{array}{c|c} \dfrac{\partial^2 \mathcal{L}(\mathbf{x}^*,\lambda^*)}{\partial \lambda^2} & \dfrac{\partial^2 \mathcal{L}(\mathbf{x}^*,\lambda^*)}{\partial \lambda \partial x} \\ \hline \dfrac{\partial^2 \mathcal{L}(\mathbf{x}^*,\lambda^*)}{\partial x \partial \lambda} & \dfrac{\partial^2 \mathcal{L}(\mathbf{x}^*,\lambda^*)}{\partial x^2} \end{array}\right] \tag{3.4}$$

where each of the four blocks is a matrix.

The sufficient conditions in this framework are twofold: one set grants local optimality and relies on the principal minors of the modified Hessian; the other exploits the concavity/convexity of the Lagrangian function in order to ensure global optimality.

Theorem 3.1.8 (Second Order Sufficient Conditions for Local optimality) *Suppose that* $(\mathbf{x}^*,\lambda^*)$ *is a critical point for* $\mathcal{L}(\mathbf{x},\lambda)$*. It holds:*

- *if the last* $(n-m)$ *leading principal minors of* $\mathcal{H}$ *have opposite signs starting from* $(-1)^{m+1}$*, then* $\mathbf{x}^*$ *is a local maximizer*
- *if the last* $(n-m)$ *leading principal minors of* $\mathcal{H}$ *have all sign* $(-1)^m$*, then* $\mathbf{x}^*$ *is a local minimizer*

Remark 3.1.2 *by "last* $(n-m)$*" we intend to proceed as follows:*

1. *determine the number* $(n-m)$ *of minors to compute*
2. *compute the determinant of each of them, starting from that of smallest dimension and recalling that the whole matrix* $\mathcal{H}$ *counts as a minor*
3. *in the maximization case, identify the correct sign to associate to each minor, starting from the one with smallest dimension*
4. *check whether the conditions of the theorem are satisfied*

The following example will clarify the practical implications of the Theorem.

Example 3.1.2 *Consider the maximization case with* $n = 3$ *and* $m = 1$*, that is* f *is a function of three unknowns and there is just one constraint. Then, it is necessary to compute* $n - m = 2$ *minors. Recall that the modified Hessian matrix has dimensions* $(n+m) \times (n+m) = (4 \times 4)$*. Since the whole matrix is a leading principal minor and we need to compute two leading principal minors, these two are* H_3 *and* $H_4 = \mathcal{H}$*, where* H_3 *stands for the whole matrix without the last row and the last column. The next step consists in matching each minor with its sign. The Theorem states that we should start with* $(-1)^{m+1}$ *and alternate, which in this case means to start with* $(-1)^2 = 1$ *to associate to the first minor and then associate* -1 *to the second minor. Since we need to start from the minor with smallest dimension, the conditions of the Theorem are met if:* $sign(H_3) = +$ *and* $sign(H_4) = -$*.*

■

Theorem 3.1.9 (Second Order Sufficient Conditions for Global optimality) *Suppose that* $\mathbf{x}^*$ *is a critical point for* $\mathcal{L}(\mathbf{x},\lambda)$*. It holds:*

- *if* $\mathcal{L}(\mathbf{x},\lambda)$ *is concave in* $\mathbf{x}$*, then* $\mathbf{x}^*$ *is a global maximum*
- *if* $\mathcal{L}(\mathbf{x},\lambda)$ *is convex in* $\mathbf{x}$*, then* $\mathbf{x}^*$ *is a global minimum*

The following proposition states the necessary and sufficient conditions for the concavity and convexity of the Lagrangian function (with respect to $\mathbf{x}$).

Proposition 3.1.10 *Suppose that $\mathcal{L}(\mathbf{x},\lambda)$ is of class $\mathcal{C}^2$. It holds:*

- *$\mathcal{L}$ is concave $\Leftrightarrow \dfrac{\partial \mathcal{L}}{\partial \mathbf{x}}$ is negative semidefinite for all $\mathbf{x}$*
- *$\mathcal{L}$ is convex $\Leftrightarrow \dfrac{\partial \mathcal{L}}{\partial \mathbf{x}}$ is positive semidefinite for all $\mathbf{x}$*

3.1.2.2 Inequality Constraints

We now turn to the study of optimization problems with inequality constraints. The setup is similar to that of the equality constraints:

$$\begin{cases} \max\limits_{(x_1,\ldots,x_n)\in\mathcal{U}} f(x_1,\ldots,x_n) \\ s.t. \\ \quad g_1(x_1,\ldots,x_n) \leq b_1 \\ \quad \vdots \\ \quad g_m(x_1,\ldots,x_n) \leq b_m \end{cases} \tag{3.5}$$

where $\mathcal{U} \subseteq \mathbb{R}^n$ is the domain of the function f and is open; $g : \mathbb{R}^n \to \mathbb{R}^m$, $\mathbf{b} \in \mathbb{R}^m$. In all cases in which one constraint holds with equality, we say that it is active (or effective or binding) at that point: this means that we are moving along the frontier of the set defined by that constraint. In all other cases, we are exploring its interior. We can then define the process of looking for optima in inequality constrained problems as a three step task:

- look for optima in the interior of the feasible set (which is given by the intersection of the domain of the objective function and all the constraints). This can be done by applying unconstrained optimization techniques, since no constraint is binding
- look for optima on the frontier of the set described by the constraints, one by one
- look for optima at the intersection of the constraint sets (if any).

Remark 3.1.3 *In presence of inequality constraints it is possible (but not certain) that the feasible set represents a closed and bounded set. If this is the case and the objective function is also continuous, then the assumptions of the Weierstrass Theorem are satisfied and we have the certainty of the existence of at least one minimum and one maximum point. In practical cases, it is very useful to draw the feasible region when the dimension of the problem makes it possible (i.e. when there are not more than three variables), for understanding whether the region is compact.*

We can proceed as before by forming the Lagrangian function:

$$\mathcal{L}(x_1,\ldots,x_n;\lambda_1,\ldots,\lambda_m) = f(x_1,\ldots,x_n) + \sum_{i=1}^{m} \lambda_i g_i(x_1,\ldots,x_n), \tag{3.6}$$

however in this case it is necessary to respect some rules concerning the sign of the Lagrange multipliers. In general, three elements need to be checked for the determination of the conditions on the sign of the multipliers:

- maximization or minimization problem
- constraints expressed as smaller than or greater than the vector of scalars $\mathbf{b}$

- the sign put in front of the multipliers in building up the Lagrangian function

The necessary conditions state, among the others, the constraints on the sign of the multipliers and are provided by the Karush-Khun-Tucker Theorem (KKT).

Theorem 3.1.11 (KKT Necessary conditions) *Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ be $\mathcal{C}^1$ functions. Suppose that $\mathbf{x}^*$ is a local maximum of f on $\mathcal{D} = \mathcal{U} \cap \{\mathbf{x} \in \mathbb{R}^n : g_j(\mathbf{x} \leq \mathbf{b}),\ j = 1, \ldots, m\}$, where $\mathcal{U} \subseteq \mathbb{R}^n$ is open. Let $E \subset \{1, \ldots, m\}$ denote the set of effective constraints at $\mathbf{x}^*$ and let $g_E = (g_j)_{j \in E}$. Suppose that (constraint qualification condition) $\rho\left(dg_E(\mathbf{x}^*)\right) = |E|$. Then there exists a vector $\lambda^* \in \mathbb{R}^m$ such that:*

- $df(\mathbf{x}^*) - \sum_{j=1}^{m} \lambda_j^* dg_j(\mathbf{x}^*) = 0$
- $\lambda_j^* \geq 0$ *and (complementary slackness conditions)* $\lambda_j^* \left[b_j - g_j(\mathbf{x}^*)\right] = 0 \ \forall j$
- $g(\mathbf{x}^* \leq \mathbf{b}$

Notice that the complementary slackness conditions impose that either a constraint is binding (i.e. $g_j(\mathbf{x}^*) = b_j$) or the corresponding multiplier is null (i.e. $\lambda_j = 0$), however the two conditions are not exclusive and can happen together. The equivalent of the constraint qualification conditions in the Lagrange Theorem are defined with reference to the set of binding constraints (requiring full rank of the Jacobian matrix associated to the set of active constraints).

Remark 3.1.4 *The previous formal statement of the Khun-Tucker Theorem assumes that we are given a maximization problem, with inequality constraints expressed in the form $g(\mathbf{x} \leq \mathbf{b}$. Many times this may not be the initial form of the problem to be solved, notwithstanding we can solve the problem by simply restating it via suitable multiplications by -1 (this is the suggested solution). Alternatively, we can modify the sign restrictions on the Lagrange multipliers, as summarized in Table (3.2). The third column reports the sign which is used in writing the Lagrangian function, that is $\mathcal{L} = f(\mathbf{x}) \pm \lambda[g(\mathbf{x}) - \mathbf{b}]$. By contrast, the sign restriction refers to the "proper" sign of the multiplier, namely $\lambda \gtreqless \mathbf{0}$.*

Problem	**Inequality of constraint**	**Sign in $\mathcal{L}$**	**Sign restriction**
max	$g_j(\mathbf{x}) \leq b_j$	-	$\lambda_j \geq 0$
max	$g_j(\mathbf{x}) \leq b_j$	+	$\lambda_j \leq 0$
max	$g_j(\mathbf{x}) \geq b_j$	-	$\lambda_j \leq 0$
max	$g_j(\mathbf{x}) \geq b_j$	+	$\lambda_j \geq 0$
min	$g_j(\mathbf{x}) \leq b_j$	-	$\lambda_j \leq 0$
min	$g_j(\mathbf{x}) \leq b_j$	+	$\lambda_j \geq 0$
min	$g_j(\mathbf{x}) \geq b_j$	-	$\lambda_j \geq 0$
min	$g_j(\mathbf{x}) \geq b_j$	+	$\lambda_j \leq 0$

Table 3.2: Sign restriction for KKT necessary conditions.

The procedure for applying the Khun-Tucker Theorem in practice requires the steps illustrated in Algorithm (**??**): the idea is, first of all, to check the behaviour of the objective function without any binding constraint (i.e. in the interior of the feasibility set, where $\lambda_i = 0 \, \forall i$), then move on each single constraint at time (hence $\lambda_i \neq 0$ only for the active constraint). Finally, one should check all the points that lie at the intersection of more than one constraint (where $\lambda_i \neq 0$ for all the constraints contemporary binding). It is clear that the number of possibilities grows with te number of constraints with the order of 2^m. However many cases would lead to contradictions and hence no candidate point. As a bottom line, recall in any case that these are just necessary conditions, therefore they does not provide any guarantee about the existence or not of optima.

Also in this case there are sufficient conditions that, under particular assumptions, allow us to draw conclusions about the nature of the candidate points found by direct application of the KKT.

Algorithm 2 Solution of Constrained Optimization problem

```
1: procedure OPTIMIZATION(f,𝒟,g,b)
2:     ℒ ← compose Lagrangian function
3:     if inequality constraints problem then
4:         x^N ← stationary points of Lagrangian, satisfying also sign restrictions
5:     else if equality constraints problem then
6:         x^N ← stationary points of Lagrangian
7:     end if
8:     if x^N = ∅ or x^N does not satisfy CQC then
9:         return No solution
10:    else
11:        ℋ ← compute Hessian matrix
12:        if some points x^N among satisfy sufficient conditions then
13:            x^S ← points that satisfy sufficient conditions, among x^N
14:            return Solution: x^S
15:        else
16:            return Solution (only candidate): x^N
17:        end if
18:    end if
19: end procedure
```

Theorem 3.1.12 (Second Order Sufficient Conditions for Global optimum) *Consider the maximization problem:*

$$\max\{f(\mathbf{x} : g(\mathbf{x}) \leq \mathbf{b})\}.$$

Suppose that $\mathbf{x}^*$ *satisfies the KKT necessary conditions. It holds:*

- *if* $\mathcal{L}(\mathbf{x},\lambda)$ *is concave in* $\mathbf{x} \Rightarrow \mathbf{x}^*$ *is a global maximum of* f *on* $\mathcal{D}$
- *if* $\mathcal{L}(\mathbf{x},\lambda)$ *is convex in* $\mathbf{x} \Rightarrow \mathbf{x}^*$ *is a global minimum of* f *on* $\mathcal{D}$

Two important remarks are due before concluding this section.

Remark 3.1.5 *The necessary and sufficient conditions for the concavity/convexity of the Lagrangian function provided in the previous subsection obviously still hold in this case. More importantly, the proposition giving the interpretation of the Lagrange multipliers still holds true too.*

Remark 3.1.6 *From a theorem in Section 2.3 we know that a linear combination of concave functions is still concave, provided that all weights are positive. Therefore we may conclude that the concavity of the Lagrangian function (required by the sufficient conditions above) follows directly from the concavity of the objective function together with the convexity of all the constraints.*

A general procedure for finding out constrained (equality as well as inequality case) optima following the analytical approach described above is described in Algorithm (2).

3.1.2.3 Level Curves Method

The general problem of constrained optimization can be solved using a graphical device when the dimension of the problem (i.e. the number of variables) is not greater than three, and both the objective function and the constraints are easy to be drawn. The procedure is called **level curve method** and requires the steps described in Algorithm (3). First of all, draw the feasible set and check whether the assumptions of the Weierstrass Theorem does apply, in order to have guidance about the existence of optima. Recall that, if these assumptions are not met, this does not imply that

there are no minimum and/or no maximum points. Then, choose an arbitrary value k_i and draw the function $f(\mathbf{x}) = k_i$ (this is called **level curve of value** k_i), saving all the points that lie in the feasible set. Repeat this for different values of k, in order to understand the direction of growth of the objective function (notice that the values of k are those of the objective function). If the region is unbounded in that direction, then there are no maximum points, equivalently there does not exist minimum points if the region is unbounded in the opposite direction (of course, both cases may occur contemporaneously, so that there are no minimum, nor maximum points). If instead the region is bounded, then save the location of the tangency points between the highest (and the lowest) possible level curves and the feasible set. These two sets of points correspond, respectively to the set of maximum and minimum points.

Remark 3.1.7 *This procedure is suitable to find constrained optima when the lie on the boundary of the feasible region, but also internal optima may be found. In this case the shape of the level curves will shrink towards the local optima for increasing (in case of maximum, or decreasing for a minimum) values of* k_i*.*

Algorithm 3 Level Curves Method

procedure OPTIMIZATION(f,$\mathcal{D}$,g,$\mathbf{b}$,$\mathbf{k}$)
  draw the feasible set as the intersection of the domain of f and all the constraints g
  for $i = 1, \ldots, p$ **do**
    draw the function $f_k : f(\mathbf{x}) = k_i$
    find out the direction of growth of f
    if region is unbounded in that direction **then**
      return Solution: No maximum
    else if region is unbounded in the opposite direction **then**
      return Solution: No minimum
    else if region is unbounded in both directions **then**
      return Solution: No maximum, nor minimum
    end if
    $x^M \leftarrow$ tangency points between highest f_k and feasibility set
    $x^m \leftarrow$ tangency points between lowest f_k and feasibility set
  end for
  return Solution: (x^M,x^m)
end procedure

The following example will clarify how the method works in a simple environment.

Example 3.1.3 *Consider the solution of the following problem by means of the level curves method:*

$$\begin{cases} \max\limits_{(x,y)} \quad 3x + y + 5 \\ s.t. \\ \quad 15x + 8y \leq 0 \\ \quad 5x + y \leq 0 \end{cases} \tag{3.7}$$

Following the instructions in Algorithm (3) we proceed by drawing the feasible set, which in this case is given by the intersection of three sets: the domain of $f(x,y)$*, which is simply* $\mathbb{R}^2$*; the half-plane given by* $y \leq -\frac{15}{8}x$ *(from the first constraint) and the half-plane defined by* $y \leq -5x$ *(from the second constraint). This region is plotted in red in the bottom panel of Fig. (3.1). The next step consists in choosing a range of values (*$\mathbf{k}$*) at which to evaluate the objective function. Consider* $\mathbf{k} = [-10, -5, 0, 5, 10, 15]$*, then draw the level curve* $f(x,y) = k_i$ *for each value of* k_i*: this leads to the equation* $3x + y + 5 = k_i$*, which is a downward sloping straight line on the plane. The outcome are*

the lines added on the bottom panel of Fig. (3.1) (the upper panel draws the objective function in the 3D space). By labelling each line with the corresponding value of the objective function, i.e. with the associate k_i, we are able to find out the direction of growth, which in this case is "towards north-east".

Next, consider the feasible region: it is clearly unbounded from below, that is it has no lower bound; by contrast it has an upper bound, since neither x nor y are allowed to approach $+\infty$. Since problem (3.1.3) requires to maximize a function and the feasible region in bounded from above, a solution might exist. We stress that the Weierstrass Theorem does not hold since the feasible region is unbounded from below, hence not closed, therefore we have no certainty about the existence of a maximum point.

In order to assess graphically whether such an optimum point really exists or not, we proceed by checking which is the highest value assumed by the objective function on the region: in other words we look for a tangency points between the highest possible level curve (in this case, the one "as far to the north-east" as possible, given the direction of fastest growth of $f(x,y)$) and the feasible set. In this example this point actually exists and is unique: it is the origin, drawn in blue in Fig. (3.1).

We can conclude that this problem admits a solution, which is also unique and coincides with the origin, where the level curve of value $k_i = 5$ (the maximum constrained value of the function is therefore 5) is tangent to both the constraints (in this case, but in general it may be tangent to just one).

■

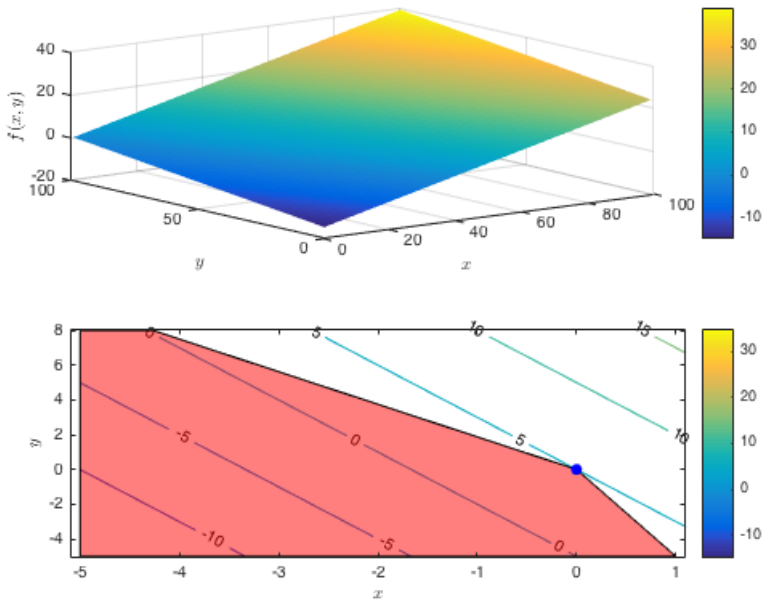


Figure 3.1: Solution of problem (3.1.3) via level curves.

3.2 Differential Equations

An ordinary differential equation (ODE) is a functional, that is a function of functions, whose arguments includes a function and its derivatives. Denoting by $x(t)$ the function $x : \mathbb{R} \to \mathbb{R}$ (from now on we will omit the argument and denote simply $x = x(t)$) and by $\dot{x} = \frac{\partial x(t)}{\partial t}$ its derivative with respect to its argument t, a generic $n-$order differential equation can be written as:

$$F\left(t, x, \frac{\partial x}{\partial t}, \ldots, \frac{\partial^n x}{\partial t^n}\right) = 0 \tag{3.8}$$

where n is the order, that is the highest degree of derivation of the function x that appears in the equation.

Definition 3.2.1 *A **general solution** of the $n-$order ordinary differential equation in (3.8) is a family of functions $x^*(t;c_1,\ldots,c_n)$ depending on n constants.*
*A **particular solution** of the $n-$order ordinary differential equation in (3.8) is a function $x^*(t)$ obtained from the general solution by fixing n initial conditions.*

3.2.1 First Order Differential Equations

A closed form solution of an ODE is not always obtainable, but in practice there are many different and simpler cases than eq. (3.8) which admit a closed form specification. In the following is given a short list of some of particular cases of first order differential equations, with the corresponding general solution.

- **separable**:

$$\dot{x} = f(t)g(x). \tag{3.9}$$

  Under the condition $g(x) \neq 0$ and additional assumptions[2] a general solution is obtained by decomposing the product and then integrating with respect to t:

$$\dot{x} = \frac{\mathrm{d}x}{\mathrm{d}t} = f(t)g(x) \tag{3.10}$$

$$\int \frac{1}{g(x)} \frac{\mathrm{d}x}{\mathrm{d}t}\, \mathrm{d}t = \int f(t)\, \mathrm{d}t \tag{3.11}$$

$$\int \frac{1}{g(x)}\, \mathrm{d}x = \int f(t)\, \mathrm{d}t \tag{3.12}$$

  which is just the computation of two integrals. Notice that in the end we come out with two constants (one from each integration), but they can be summed and renamed (the sum of two constants is a constant too) as to obtain a unique constant (as should be, since the equation has order 1).

- **exact**:

$$P(t,x) + Q(t,x)\dot{x} = 0. \tag{3.13}$$

  The necessary and sufficient condition for a solution of eq. (3.13) is:

$$\frac{\partial P}{\partial x} = \frac{\partial Q}{\partial t} \tag{3.14}$$

  Equation (3.13) can be solved in a similar fashion as in (3.12), integrating with respect to t:

$$\int P(t,x)\, \mathrm{d}t + \int Q(t,x) \frac{\mathrm{d}x}{\mathrm{d}t}\, \mathrm{d}t = 0 \tag{3.15}$$

$$\int P(t,x)\, \mathrm{d}t + \int Q(t,x)\, \mathrm{d}x = 0$$

  again we obtain a unique constant by a similar reasoning.

The particular case to which we will devote particular attention is represented by the class of **linear differential equations**, whose general formulation in the first order case is:

$$\dot{x} = a(t)x + b(t) \tag{3.16}$$

[2] See an advanced book on mathematical analysis for the details.

where $a(t)$ and $b(t)$ are general functions of the variable t. The formula for finding out a general solution is obtained by noting that the unique function whose derivative "contains" the original function is the exponential (with base e). As a consequence we know that the general solution will be given by an exponential and an additional part which depends on the term $b(t)$. More precisely:

$$x^* = e^{A(t)} \left[c + \int e^{-A(t)} b(t) \, \mathrm{d}t \right] \quad c \in \mathbb{R} \tag{3.17}$$

where:

$$A(t) = \int a(t) \, \mathrm{d}t \tag{3.18}$$

and c is a constant.

A simpler case is the one with $a(t) = a$, whose solution is:

$$x^* = e^{at} \left[c + \int e^{-at} b(t) \, \mathrm{d}t \right] \quad c \in \mathbb{R}. \tag{3.19}$$

An additional simplification of computations arises in the more particular case of **linear differential equations with constant coefficients**, when:

$$\dot{x} = ax + b \tag{3.20}$$

where, by simply applying (3.17), it is obtained:

$$x^* = e^{at} \left[c - \frac{b}{a} e^{-at} \right] = ce^{at} - \frac{b}{a} \quad c \in \mathbb{R}. \tag{3.21}$$

This case is at the core of the following analysis since its simple structure allows to obtain an analytical closed form solution, which is a desirable result in economics (as well as other sciences). When studying a particular function (for example the consumption function), we are often interested in its equilibrium point, which is defined as the limiting value that the function achieves (if it exists), and can then be interpreted as the long run value of the function[3].

Definition 3.2.2 *An **equilibrium point** for an ordinary differential equation (3.8) is any constant solution of the equation.*

Here a constant solution is one such that its derivative is null: $\dot{\bar{x}} = 0$. In general the particular solution of almost all equations like eq. (3.8) are functions of the variable t, hence not constant; nonetheless we can study their asymptotic properties as $t \to +\infty$ and check whether they admit or not an equilibrium point in the limit. This is done by computing the limit of the general solution of the differential equation. Denoting $\bar{x}$ the equilibrium point of the solution $x^*(t)$, three cases may arise:

Definition 3.2.3 *Given a general solution $x^*(t)$ of an ordinary differential equation, with equilibrium point $\bar{x}$, the latter is classified as:*

(i) ***globally asymptotically stable**, if*

$$\lim_{t \to +\infty} x^*(t) = \bar{x}$$

for all initial conditions[4], since the solution always converges to the equilibrium;

[3] Notice that in this case we have interpreted the variable t as time and the function $x(t)$ as consumption. Nonetheless, the theory reported in these sections is a general one, which applies even when we attach to t and $x(t)$ very different other meanings

[4] Equivalently, one may say "for all solutions $x(t)$". This equivalence is due to the fact that changing the initial conditions implies changing the particular solution.

(ii) ***locally asymptotically stable****, if*

$$\lim_{t\to+\infty} x^*(t) = \bar{x}$$

for all initial conditions in a neighbourhood of the equilibrium $\bar{x}$, since the solution converges to the equilibrium only if it starts from a sufficiently close point;

(iii) ***unstable****, if*

$$\lim_{t\to+\infty} x^*(t) = \pm\infty$$

since the solution diverges from the equilibrium as t increases.

3.2.2 Second Order Differential Equations

In this subsection we constrain to the analysis of the linear case. A **linear second order ordinary differential equation** is given by:

$$\ddot{x} + a_0(t)\dot{x} + a_1(t)x = b(t). \tag{3.22}$$

We will focus on the case in which the coefficients on the left hand side are constant, that is:

$$\ddot{x} + a_0\dot{x} + a_1 x = b(t) \tag{3.23}$$

and on the **linear second order ordinary differential equation with constant coefficients**:

$$\ddot{x} + a_0\dot{x} + a_1 x = b. \tag{3.24}$$

The method for finding a general solution of eq. (3.23), which applies also to eq. (3.24), consists in two steps:

I. find a solution of the characteristic polynomial:

$$\lambda^2 + a_1\lambda + a_0 = 0 \tag{3.25}$$

associated to the homogeneous equation:

$$\ddot{x} + a_0\dot{x} + a_1 x = 0$$

II. find a particular solution of the whole equation (3.23), whose formula depends on the function $b(t)$.

The solutions of eq. (3.25) can fall in three cases:

- $\lambda_1 \neq \lambda_2$, $(\lambda_1,\lambda_2) \in \mathbb{R}^2$, two real and distinct solutions. The the solution of the homogeneous equation is:
$$x^*(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} \quad c_1 \in \mathbb{R},\ c_2 \in \mathbb{R} \tag{3.26}$$
- $\lambda_1 = \lambda_2 = \lambda \in \mathbb{R}$, two real and coincident roots. The the solution of the homogeneous equation is:
$$x^*(t) = c_1 e^{\lambda t} + c_2 t e^{\lambda t} \quad c_1 \in \mathbb{R},\ c_2 \in \mathbb{R} \tag{3.27}$$
- $\lambda_1 = \alpha + i\beta \neq \lambda_2 = \alpha - i\beta$, $(\lambda_1,\lambda_2) \in \mathbb{C}^2$, two complex and conjugate solutions. The the solution of the homogeneous equation is:
$$x^*(t) = e^{\alpha t}(c_1 \cos(\beta t) + c_2 \sin(\beta t)) \quad c_1 \in \mathbb{R},\ c_2 \in \mathbb{R} \tag{3.28}$$

The particular solution, instead, should be found in the same class of the function $b(t)$, by an application of the *method of undetermined coefficients*. It is necessary to stress that, in order to have linearly independent "components" of the solution, is is necessary to multiply the candidate solution by t or t^2 before applying the method of undetermined coefficients, according to the following scheme:

- $b(t) = Q(t)$ polynomial of degree m. Then the candidate is ($P(t)$ is a complete polynomial of degree m):
$$\begin{cases} x(t) = P(t) & \text{if } a_0 \neq 0 \\ x(t) = tP(t) & \text{if } a_0 = 0,\ a_1 \neq 0 \\ x(t) = t^2 P(t) & \text{if } a_0 = 0,\ a_1 = 0 \end{cases}$$
- $b(t) = ae^{\alpha t}$. Then the candidate is:
$$\begin{cases} x(t) = ke^{\alpha t} & \text{if } \alpha \text{ is not root of characteristic polynomial} \\ x(t) = kte^{\alpha t} & \text{if } \alpha \text{ is simple root of characteristic polynomial} \\ x(t) = kt^2 e^{\alpha t} & \text{if } \alpha \text{ is double root of characteristic polynomial} \end{cases}$$
- $b(t) = a\cos(\gamma t) + b\sin(\gamma t)$. Then the candidate is:
$$\begin{cases} x(t) = k_1 \cos(\gamma t) + k_2 \sin(\gamma t) & \text{if roots of characteristic polynomial have nonzero real part} \\ x(t) = t(k_1 \cos(\gamma t) + k_2 \sin(\gamma t)) & \text{if roots of characteristic polynomial have zero real part} \end{cases}$$

Concerning the stability of the equilibrium solutions (if existing), the following conditions hold:

Lemma 3.2.1 *Given a second order linear ordinary differential equation with $a_0(t) = a_0$ and $a_1(t) = a_1$ and let $\bar{x}$ be an equilibrium point. Then:*

- *in case of two real distinct roots of the characteristic polynomial, if $\lambda_1 < 0$ and $\lambda_2 < 0$ then the equilibrium is globally asymptotically stable;*
- *in case of two real coincident roots of the characteristic polynomial, if $\lambda < 0$ then the equilibrium is globally asymptotically stable;*
- *in case of two complex conjugate roots of the characteristic polynomial, if $\alpha < 0$ then the equilibrium is globally asymptotically stable.*

3.2.3 Dynamical Systems

A system of ordinary differential equations is given by a set of n ODEs:

$$\begin{cases} \dot{x}_1 = f_1(t, x_1, \ldots, x_n) \\ \vdots \\ \dot{x}_n = f_n(t, x_1, \ldots, x_n) \end{cases} \tag{3.29}$$

Clearly, in presence of n first order differential equations, the general solution of the system will be a family of functions depending on n constants, which can be found by specifying the same number of initial conditions. In this subsection we focus on the method for solving bidimensional systems ($n = 2$); more in detail, we consider only two special cases of the general formulation in eq. (3.29).

As a starting point, consider the case in which the objective system is:

$$\begin{cases} \dot{x} = f(t, x, y) \\ \dot{y} = g(t, y) \end{cases} \tag{3.30}$$

then it is possible to solve the system by **recursion**:

i) solve the second equation, using the ordinary methods for first order differential equations and get a solution $y^*(t)$

ii) plug it in the first equation and solve for $x^*(t)$

The second case we consider is a **system of first order linear differential equations with constant coefficient matrix** ($A(t) = A$) and variable vector $\mathbf{b(t)}$, which can be written in matrix form as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1(t) \\ b_2(t) \end{bmatrix} \quad \Rightarrow \quad \dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b(t)}. \tag{3.31}$$

This problem can be solved following two main approaches: the first is similar to the recursive method described previously. We may recover, for example, $y(\dot{x},x,t)$ from the first equation (if $a_{12} \neq 0$), then we compute its first derivative and plug both in the second equation, which becomes a second order linear differential equation in x. Now, by using the techniques for second order differential equations we solve for $x^*(t)$. Finally, plug this solution together with its first derivative back into $y(\dot{x},x,t)$ to obtain $y^*(t)$.

The second approach is based on the study of the eigenvalues of the matrix A: denoting them as λ_1 and λ_2, we have three possible cases, as for the second order linear differential equations in eq. (3.23) (in fact a system of two first order differential equations can be restated a single second order differential equation and vice versa).

- $\lambda_1 \neq \lambda_2$ $(\lambda_1,\lambda_2) \in \mathbb{R}^2$, then the solution of the homogeneous system is:

$$\mathbf{x} = c_1\mathbf{v_1}e^{\lambda_1 t} + c_2\mathbf{v_2}e^{\lambda_2 t} \quad (c_1,c_2) \in \mathbb{R}^2$$

- $\lambda_1 = \lambda_2 = \lambda \in \mathbb{R}$, then the solution of the homogeneous system is:

$$\mathbf{x} = c_1\mathbf{v_1}e^{\lambda t} + c_2\mathbf{v_1}te^{\lambda t} \quad (c_1,c_2) \in \mathbb{R}^2$$

- $\lambda_1 \neq \lambda_2$ $(\lambda_1,\lambda_2) \in \mathbb{C}^2$, then the solution of the homogeneous system is:

$$\mathbf{x} = e^{\alpha t}(c_1\mathbf{v_1}\cos(\beta t)) + c_2\mathbf{v_2}\sin(\beta t)) \quad (c_1,c_2) \in \mathbb{R}^2$$

where $\mathbf{v_1}$ and $\mathbf{v_2}$ are the eigenvectors associated to the eigenvalues λ_1 and λ_2, respectively. We do not consider here the nonhomogeneous case.

As a final step in this short analysis of dynamical systems, we give an insight to the study of the properties of equilibrium points of a dynamical systems in two particular cases of bidimensional systems: the linear with constant coefficients and the autonomous case.

In the first case we have a bidimensional system of linear first order differential equations with constant coefficients in matrix form:

$$\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b} \tag{3.32}$$

and assume it admits an equilibrium point $(\bar{x}(t),\bar{y}(t))$. Then it holds:

Theorem 3.2.2 *[15, p 244] Given the system in eq. (3.32) with $|A| \neq 0$ and equilibrium point $(\bar{x}(t),\bar{y}(t))$, then the latter is globally asymptotically stable if and only if:*

$$\mathrm{tr}(A) < 0 \quad \text{and} \quad \det(A) > 0$$

or, equivalently, if both the eigenvalues of A have negative real part.

The second case we consider is that of a homogeneous nonlinear system:

$$\begin{cases} \dot{x} = f(x,y) \\ \dot{y} = g(x,y) \end{cases} \tag{3.33}$$

for which hold the following two theorems:

Theorem 3.2.3 *[15, p. 252] Given the system in eq. (3.33) where f and g are $\mathcal{C}^1$ functions, let $(\bar{x}(t),\bar{y}(t))$ be the equilibrium point and J the Jacobian:*

$$J = \begin{bmatrix} f_1'(\bar{x},\bar{y}) & f_2'(\bar{x},\bar{y}) \\ g_1'(\bar{x},\bar{y}) & g_2'(\bar{x},\bar{y}) \end{bmatrix}.$$

If:

$$\mathrm{tr}(J) < 0 \quad \text{and} \quad \det(J) > 0$$

or, equivalently, if both the eigenvalues of J have negative real part, then the equilibrium point is locally asymptotically stable.

Theorem 3.2.4 *[15, p. 255] Given the system in eq. (3.33) where f and g are $\mathcal{C}^1$ functions, let $(\bar{x}(t),\bar{y}(t))$ be the equilibrium point and J the Jacobian:*

$$J = \begin{bmatrix} f_1'(\bar{x},\bar{y}) & f_2'(\bar{x},\bar{y}) \\ g_1'(\bar{x},\bar{y}) & g_2'(\bar{x},\bar{y}) \end{bmatrix}.$$

If:

$$\det(J) < 0$$

or, equivalently, if the eigenvalues of J are nonzero real numbers with opposite sign, then the equilibrium point is saddle point.

3.2.4 Qualitative Analysis

So far it has been proposed and described the **analytical** approach, which aims mainly at finding out the exact solution (when existent) to the problem under observation. In some particular cases the same problems can be tackled from a different, but complementary point of view. The **qualitative** approach has the goal of characterizing the solution of a differential equation or of a system of differential equations in terms of:

(i) the existence and uniqueness of equilibrium points;

(ii) the stability of equilibrium points.

In this subsection we consider only first order differential equations and systems of first order differential equations. It is necessary to stress that the applicability of this kind of analysis is restricted to the satisfaction of particular conditions by the differential equation, which must be autonomous.

Definition 3.2.4 *A first order ordinary differential equation is said to be **autonomous** if it does not depend explicitly on the variable t, that is:*

$$\dot{x} = f(x)$$

In the following we restate the definition of equilibrium point:

Definition 3.2.5 *Given a generic ordinary differential equation, an **equilibrium point** is any constant solution for it.*

*Given a first order autonomous ordinary differential equation, an **equilibrium point** is any constant solution, in particular it is a point such that the derivative of the function evaluated at that point is zero.*

3.2.4.1 First order single equation

In the case of a single autonomous differential equation, we can represent it on the plane, in which the horizontal axis reports x, while the vertical one gives the values $\dot{x} = f(x)$. This representation is called **phase diagram** and is illustrated in Figure (3.2).

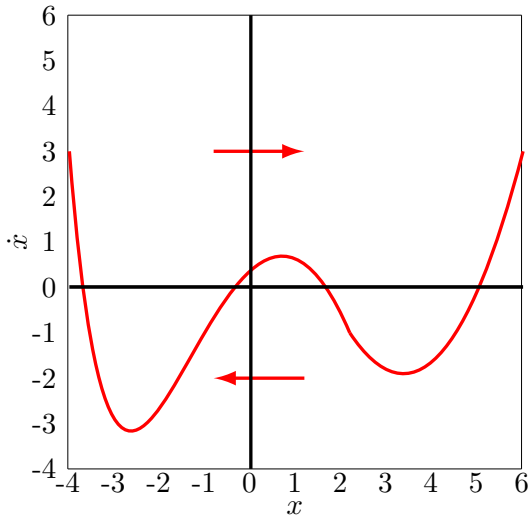


Figure 3.2: Phase diagram for single differential equation.

An important remark is due: since the graph reports on the vertical axis $\dot{x} = f(x)$, it holds:

- $\forall x(t) : f(x) = \dot{x} > 0$ the "next" values of $x(t)$ (for successive t) will grow, since the derivative $\frac{\mathrm{d}x}{\mathrm{d}t}$ is positive. Hence above the horizontal axis has been drawn an arrow pointing right.
- $\forall x(t) : f(x) = \dot{x} < 0$ the "next" values of $x(t)$ (for successive t) will decrease, since the derivative $\frac{\mathrm{d}x}{\mathrm{d}t}$ is negative. Hence above the horizontal axis has been drawn an arrow pointing left.
- $\forall x(t) : f(x) = \dot{x} = 0$ the "next" values of $x(t)$ (for successive t) will remain the same, since the derivative $\frac{\mathrm{d}x}{\mathrm{d}t}$ is null. Hence these points are the equilibrium points of the autonomous differential equation.

The graph in which are reported many small arrows pointing in the direction of movement (not just the horizontal or vertical) on the plane is called **flow field** (see example 3.2.1). Generally the phase diagram and the flow field come together in a graphical analysis and can be superimposed, as in the case of dynamical systems (see example 3.2.2).

As a consequence of the previous remark, it is possible to check the existence and uniqueness of equilibrium points by looking for the intersection points of the graph with the horizontal axis or, alternatively, by solving: $f(x) = 0$. In particular, the function in Figure (3.2) intersects the $x-$axis four times, hence it admits four equilibria.

In order to characterize the equilibrium points, we can exploit the information about the dynamics of the solution $x(t)$ which are represented by the arrows. Take for example the first equilibrium point to the left of Fig. (3.2). Since for values of $x(t)$ in its left neighbourhood the function $\dot{x} = f(x)$ is positive, then the value of $x(t)$ is increasing, pointing towards the equilibrium; for values of $x(t)$ in its right neighbourhood, instead, the function $\dot{x} = f(x)$ is negative, then the value of $x(t)$ is decreasing, again pointing towards the equilibrium. The same happens for the third equilibrium point. By contrast, if we take a value in the left neighbourhood of the second equilibrium point the resulting value of the derivative is negative, hence $x(t)$ is decreasing, and the movement is departing from the equilibrium; taking a value in the right neighbourhood, the resulting value of the derivative is positive, hence $x(t)$ is increasing, and the movement is again departing from the equilibrium.

Now that the movements have been described, we can state the following rule for the determination of the stability properties of an equilibrium point:

Lemma 3.2.5 *Given an autonomous first order ordinary differential equation, with equilibrium point $\bar{x}$, it holds that:*

- *if $f'(\bar{x}) < 0$ then $\bar{x}$ is locally asymptotically stable;*
- *if $f'(\bar{x}) > 0$ then $\bar{x}$ is unstable;*
- *if $f'(\bar{x}) = 0$ then it is necessary to check the behaviour of the function $f(x)$ locally in a neighbourhood of $\bar{x}$.*

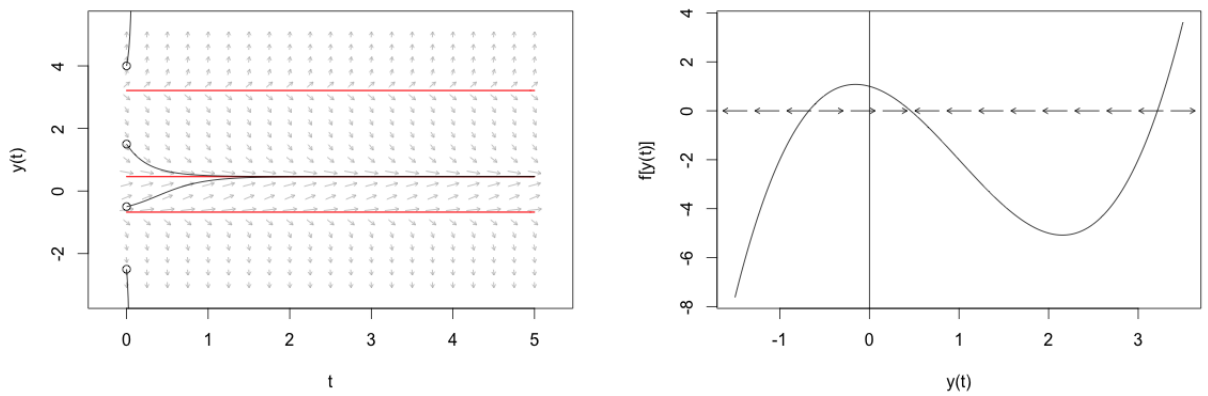
In words, it suffices to check the sign of the first derivative of the function in the equilibrium point to determine its properties.

Example 3.2.1 *Consider the following first order autonomous ordinary differential equation:*

$$\dot{y} = ay^3 + by^2 - y - 1, \qquad (a,b) \in \mathbb{R}^2. \tag{3.34}$$

Let $a = 1, b = -3$, then draw the phase diagram and study the stability of the equilibrium points after having found them.

We start by drawing the function $f(y) = y^3 - 3y^2 - y - 1$, as results in Figure (3.3(b)).



(a) *Flow field* (b) *Phase diagram*

Figure 3.3: Flow field and Phase diagram of (3.34).

The graph in Fig. (3.3(a)) plots the solution function $y^*(t)$ for four different initial values. The three red lines correspond to the three points in Fig. (3.3(b)) where the function intersects the horizontal axis, that is the equilibrium points of the solution. Notice the dynamics: for initial values $y(0)$ below the lowest equilibrium the function goes to $-\infty$, while for initial values between the third (i.e. smallest) equilibrium and the one in the middle the function converges to the middle equilibrium. The same happens als for any initial value between the middle and the highest equilibrium, while for values bigger than the latter, the solution explodes to $+\infty$. We can conclude that the "lowest" and "highest" equilibria are unstable, while the "middle" is locally asymptotically stable[5].

Exactly the same conclusions can be drawn be looking at the arrows drawn on the horizontal axis ($x-$axis) in Fig. (3.3(b)). These arrows in fact represent the direction of movement of $y(t)$ in the corresponding intervals where they are drawn.

[5]The meaning of "locally" is now clear: changing the initial condition we may not converge towards the middle equilibrium.

Finally, coming back to analytical approaches, one could also study the stability of the equilibria by computing the sign of the derivative at each equilibrium point. This method is correct, nonetheless visual inspect in this case is more direct because the shape of the curve in the neighbourhood of the intersections with the horizontal is simple. It is possible to say without any error that the function has negative slope at the "middle" equilibrium, while in the other two points its slope is positive. By applying theorem 3.2.5 we obtain the conclusions.

■

3.2.4.2 First order system of equations

The idea at the basis of the previous subsection can be carried on when studying a system of autonomous equation. Here we give a formal definition, similar to Def. (3.2.4).

Definition 3.2.6 *Given a system of first order ordinary differential equations, it is said to be* ***autonomous*** *if:*

$$\begin{cases} \dot{x}_1 = f_1(x_1, \dots, x_n) \\ \vdots \\ \dot{x}_n = f_n(x_1, \dots, x_n) \end{cases}$$

that is, none of the differential equations depends explicitly on the variable t.

In this subsection we focus on (2×2) systems. The concept of equilibrium point in a system is a generalization of that for single equations in Def. (3.2.5):

Definition 3.2.7 *Given an autonomous system of first order ordinary differential equations, an* ***equilibrium point*** *(or stationary point) is any solution of the system such that:*

$$\begin{cases} \dot{x}_1^* = 0 \\ \vdots \\ \dot{x}_n^* = 0 \end{cases}$$

where $\dot{x}_i^*$ specifies the equation evaluated at the equilibrium point.

For a generic (2×2) system like:

$$\begin{cases} \dot{x} = f(x,y) \\ \dot{y} = g(x,y) \end{cases} \tag{3.35}$$

it can be proved that the equilibrium point, when existing, is located at the intersection of the two curves $f(x,y) = 0$ and $g(x,y) = 0$, which are called the **nullclines** (or stationary loci) of the system. These are nothing more than the set of points where the first and second differential equations are null, respectively. We can represent them on the x-y plane, which is called the phase diagram. Notice that this is different from that used in the single equation case: the former has two axis, each of which is one of the two functions of the system $(x(t),y(t))$, while the latter has on the horizontal axis the function $(x(t))$ and the vertical one the derivative of the function $(\dot{x}(t))$.

A possible phase diagram for a bidimensional system is drawn in Figure (3.4), where the blue curve represent the $\dot{x} = 0$ nullcline (or stationary locus), while the red curve is the $\dot{y} = 0$ nullcline. These curves are graphically drawn by simply putting the first and the second equation (respectively) to zero and drawing the resulting curves as $\dot{x} = 0 \to y = h_f(x)$ and $\dot{y} = 0 \to y = h_g(x)$.

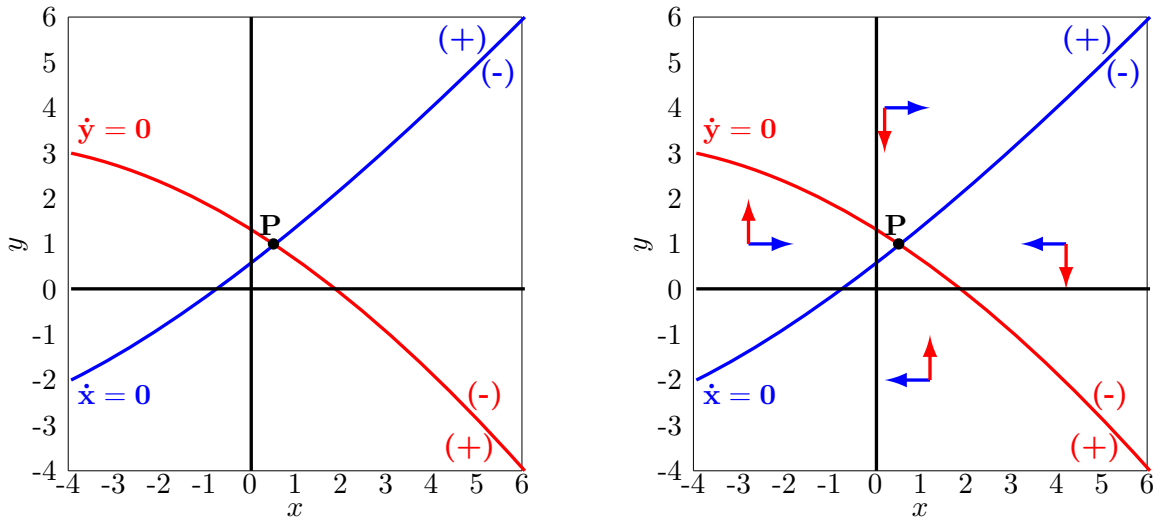


Figure 3.4: Phase diagram for system of two differential equations.

It is necessary to pay attention in interpreting this graph. The first step after drawing the nullclines is to find out in which part of the semi-plane the corresponding function is increasing and in which it is decreasing, this is done in the left part of Fig. (3.4). In this case, consider first the nullcline $\dot{x} = 0$. First of all, remind that all along the this blue curve the function $x(t)$ is stationary (its derivative is null), but the function $y(t)$ is not (and the converse result holds for the red nullcline).

In order to find out what is happening to the $x(t)$ above and below the corresponding nullcline, it is possible to take a point (x^*,y^*) on the nullcline itself, such that $\dot{x}(x^*,y^*) = f(x^*,y^*) = 0$. Then increase by a small amount (ϵ) the value of the coordinate y[6] if $\dot{x}(x^*,y^* + \epsilon) = f(x^*,y^* + \epsilon) > 0$ then we know that above the nullcline $\dot{x} = 0$ we have a region in which $\dot{x} > 0$, hence $x(t)$ is increasing and for this reason we put a blue "+" above the blue curve. By a similar token, we proceed by decreasing the coordinate y by a small amount (δ): in this case it turns out that $\dot{x}(x^*,y^* + \delta) = f(x^*,y^* + \delta) < 0$ then we know that below the nullcline $\dot{x} = 0$ we have a region in which $\dot{x} < 0$, hence $x(t)$ is decreasing and for this reason we put a blue "−" below the blue curve. We redo the same analysis on the other nullcline and we end up with the situation depicted on the left part of Fig. (3.4). Notice that the basic idea behind this approach is to look at what happens to the function $f(x,y)$ (and $g(x,y)$, respectively) as we move a little bit away (above and below, respectively) from the nullcline, where it is zero.

The "+" and "−" symbols can be substituted by arrows indicating the dynamics of each function. Again, starting from the dynamics of $x(t)$, we know that above the blue nullcline (where $\dot{x} = 0$) it holds $\dot{x} > 0$, hence we can put a horizontal arrow above the nullcline pointing to the right, meaning that for all points above the curve $\dot{x} = 0$ the function $x(t)$ is increasing. By contrast we draw a horizontal arrow pointing left below the curve. For what concerns the $\dot{y} = 0$ nullcline, since "+" is put below the red curve, we know that at all the points below this nullcline the function $y(t)$ is increasing, that is the direction of movement is upwards. Therefore we put an upwards pointing red arrow in the region below the nullcline and by a similar reasoning a downward pointing red arrow in the region above the nullcline. We end up in a situation like the one drawn in the right graph of Fig. (3.4). Notice that the two nullclines have formed four different regions in the plane, and for each of them there is a pair of arrows, one indicating the direction of movement of $x(t)$ (blue) and another giving that of $y(t)$ (red).

Notice that the equilibrium point is given by the intersection of the two nullclines: here we have only one equilibrium point, P. The stationarity of this point can be studied by checking the direction of the arrows in the diagram:

- if all the pairs of arrows are pointing towards the equilibrium point, then it is globally asymptotically stable;

[6] Alternatively, we can take the coordinate y fixed and move the coordinate x.

- if two pairs of arrows (in opposite regions) are pointing towards the equilibrium point, while the other two pairs (in the other two opposite regions) are pointing away from it, then the equilibrium is stable saddle and there exists a saddlepath leading to it;
- if all the pairs of arrows are pointing away from the equilibrium point, then it is unstable.

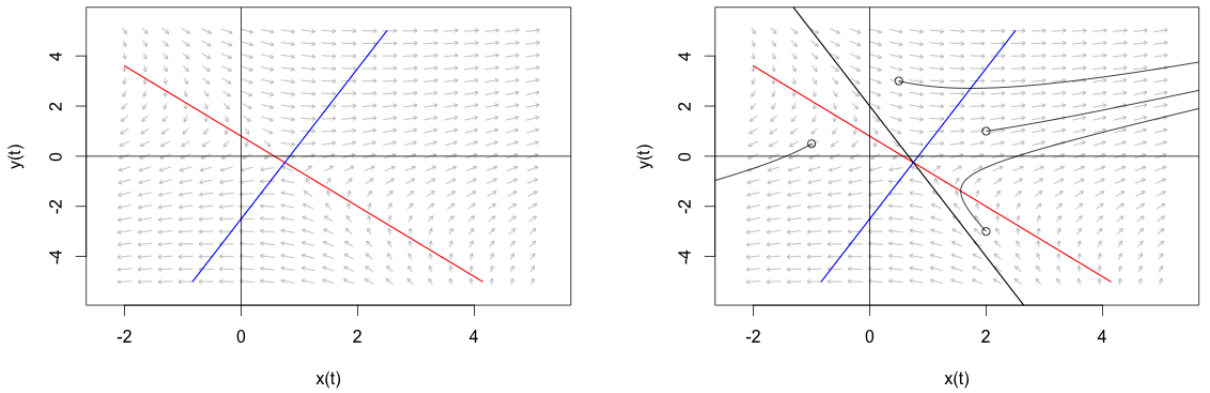
Clearly, the example drawn in Fig. (3.4) shows that P is a globally asymptotically stable equilibrium point for the system.

Example 3.2.2 *Consider the following first order autonomous ordinary differential equation:*

$$\begin{cases} \dot{x} = 7x + 5y - 4 \\ \dot{y} = 6x - 2y - 5 \end{cases} \tag{3.36}$$

Draw the phase diagram and study the stability of the equilibrium points after having found them.

We start by drawing the functions $f(x,y) = 7x + 5y - 4 = 0$ *and* $g(x,y) = 6x - 2y - 5 = 0$*, as results in Figure* (3.5(b)).



(a) *Phase diagram with nullclines*

(b) *Phase diagram with equilibrium and trajectories*

Figure 3.5: Phase diagram of (3.36).

The graph in Fig. (3.5(a)) *plots the nullclines of the system: the red line is the solution of* $f(x,y) = 0$*, while the blue is the solution of* $g(x,y) = 0$*. In the panel* (3.5(b)) *are reported four initial conditions and the corresponding trajectories: notice that the latter follow the path indicated by the small grey arrows as in the previous example. The nullclines intersect only in one point, hence there is one equilibrium point.*

In order to study its stability properties, notice that the four regions in which the plane is divided by the two nullclines have the different dynamics described by the small grey arrows. In particular, it is shown that, for whatever point we choose apart from those on the black line, the trajectory will lead either to $-\infty$ *or* $+\infty$*. The black line identify the* ***stable saddle path*** *and the equilibrium point is in fact a saddle: for all points along the saddle path the trajectory leads to the saddle point (in this respect the point is "stable"), however, for any point outside the saddle path the trajectory will lead far apart from it and never cross it (in this respect, roughly speaking, it is a "saddle").*

■

3.3 Difference Equations

In the discrete domain the changes of the variable t are no more infinitesimal, as in the continuous case, but are discrete. As a consequence we do not use anymore the derivative of the function $x(t)$ for describing its change, rather we consider the difference between two subsequent values of t. The outcome of such a "discretization" process is a difference equation, whose general form (n order) is:

$$F\left(t,x_t,\ldots,x_{t+n}\right)=0. \tag{3.37}$$

It is immediate to note the similarity that characterizes the difference equation in (3.37) and the differential equation in (3.8). Indeed the logical steps and solution methods developed for the latter apply (with slight modifications due to the discrete domain instead of the continuous one) also to the former. As a consequence in the following subsections only the most remarkable differences will be highlighted without entering too much into the details of the procedures and methods that are analogous to those developed for the differential equations.

3.3.1 First Order Difference Equations

The general formulation of a first order difference equation is:

$$x_{t+1}=f(t,x_t)$$

and, as well as first order differential equations, the solution of this equation is a family of functions $x_t^*(c)$ where $c\in\mathbb{R}$ is a constant.

In this context too we focus on the special case of **linear difference equations** which can be expressed as:

$$x_{t+1}=a(t)x_t+b(t) \tag{3.38}$$

which has a general solution (see [15, 395]) of the form:

$$x_t^*=\left(\prod_{s=0}^{t-1}a(t)\right)x_0+\sum_{k=0}^{t-1}\left(\prod_{s=k+1}^{t-1}a(t)\right)b(k). \tag{3.39}$$

In the following, however, we study more in detail two particular cases; the first is obtained when $a(t)=a$:

$$x_{t+1}=ax_t+b(t) \tag{3.40}$$

while the second is the general **first order difference equation with constant coefficients**:

$$x_{t+1}=ax_t+b \tag{3.41}$$

The solution of these two cases is similar in its structure to that of first order linear differential equations in eq. (3.17) and eq. (3.21), with the exponential function replaced by the power function[7]. The general solution of eq. (3.41) is given by:

$$\begin{cases} x_t^*=ca^t+\frac{b}{1-a} & c\in\mathbb{R} \quad \text{if } a\neq 1 \\ x_t^*=c+bt & c\in\mathbb{R} \quad \text{if } a=1 \end{cases} \tag{3.42}$$

The constant can be found out by fixing an initial condition.

Definition 3.3.1 *An **equilibrium point** for a difference equation is any of its constant solutions.*

[7] The intuition behind this solution can be found by trying to solve iteratively the equation.

Notice that in this framework a solution is said to be constant when $x_{t+1} = x_t = \bar{x}$, differently from the differential equation case in which it was $\dot{x} = 0$[8].

The stability of an equilibrium point $\bar{x}$ for a generic first order difference equation follows the same concepts as in the case of differential equations; in particular:

Definition 3.3.2 *Given a general solution x_t^* of an difference equation, with equilibrium point $\bar{x}$, the latter is classified as:*

(i) ***globally asymptotically stable****, if*

$$\lim_{t \to +\infty} x_t^* = \bar{x}$$

for all initial conditions, since the solution always converges to the equilibrium;

(ii) ***locally asymptotically stable****, if*

$$\lim_{t \to +\infty} x_t^* = \bar{x}$$

for all initial conditions in a neighbourhood of the equilibrium $\bar{x}$, since the solution converges to the equilibrium only if it starts from a sufficiently close point;

(iii) ***unstable****, if*

$$\lim_{t \to +\infty} x^*(t) = \pm\infty$$

since the solution diverges from the equilibrium as t increases.

Having a quick look at the structure of the general solution in eq. (3.42) (but it also holds for the case with non constant $b(t)$):

Lemma 3.3.1 *In a linear difference equation with constant coefficients like eq. (3.41), the equilibrium point $\bar{x}$, when it exists is:*

- *globally asymptotically stable if $|a| < 1$*
- *unstable if $|a| \geq 1$*

3.3.2 Second Order Difference Equations

As for the second order difference equations, we consider only the linear case. A **linear second order difference equation** is given by:

$$x_{t+2} + a_0(t)x_{t+1} + a_1(t)x_t = b(t). \tag{3.43}$$

Two cases are interesting: the first is that in which the coefficients on the left hand side are constant, that is:

$$x_{t+2} + a_0 x_{t+1} + a_1 x_t = b(t) \tag{3.44}$$

while on the other hand we have the **linear second order difference equation with constant coefficients**:

$$x_{t+2} + a_0 x_{t+1} + a_1 x_t = b. \tag{3.45}$$

The method for finding a general solution of eq. (3.44), which applies also to eq. (3.45), consists in the same two steps developed for the differential equations (with the change due to the power function instead of the exponential):

[8] In reality this is the same condition, just rephrased in the continuous or discrete case.

I. find a solution of the characteristic polynomial:

$$\lambda^2 + a_1\lambda + a_0 = 0 \tag{3.46}$$

associated to the homogeneous equation:

$$x_{t+2} + a_0 x_{t+1} + a_1 x_t = 0$$

II. find a particular solution of the whole equation (3.44), whose formula depends on the function $b(t)$.

The solutions of eq. (3.46) can fall in three cases:

- $\lambda_1 \neq \lambda_2$, $(\lambda_1,\lambda_2) \in \mathbb{R}^2$, two real and distinct solutions. The the solution of the homogeneous equation is:
$$x_t^* = c_1\lambda_1^t + c_2\lambda_2^t \quad c_1 \in \mathbb{R},\ c_2 \in \mathbb{R} \tag{3.47}$$
- $\lambda_1 = \lambda_2 = \lambda \in \mathbb{R}$, two real and coincident roots. The the solution of the homogeneous equation is:
$$x_t^* = c_1\lambda^t + c_2 t\lambda^t \quad c_1 \in \mathbb{R},\ c_2 \in \mathbb{R} \tag{3.48}$$
- $\lambda_1 = \alpha + i\beta \neq \lambda_2 = \alpha - i\beta$, $(\lambda_1,\lambda_2) \in \mathbb{C}^2$, two complex and conjugate solutions. The the solution of the homogeneous equation is:
$$x_t^* = \rho^t(c_1\cos(\theta t) + c_2\sin(\theta t)) \quad c_1 \in \mathbb{R},\ c_2 \in \mathbb{R} \tag{3.49}$$
where $\rho = \sqrt{\alpha^2 + \beta^2}$ and $\cos(\theta) = \frac{\alpha}{\rho}$.

The particular solution, instead, should be found in the same class of the function $b(t)$, by an application of the *method of undetermined coefficients*. It is necessary to stress that, in order to have linearly independent "components" of the solution, is is necessary to multiply the candidate solution by t or t^2 before applying the method of undetermined coefficients, according to the following scheme:

- $b(t) = Q(t)$ polynomial of degree m. Then the candidate is ($P(t)$ is a complete polynomial of degree m):
$$\begin{cases} x_t = P(t) & \text{if } \alpha = 1 \text{ is not root of characteristic polynomial} \\ x_t = tP(t) & \text{if } \alpha = 1 \text{ is simple root of characteristic polynomial} \\ x_t = t^2P(t) & \text{if } \alpha = 1 \text{ is double root of characteristic polynomial} \end{cases}$$
- $b(t) = a\gamma^t$. Then the candidate is:
$$\begin{cases} x_t = k\gamma^t & \text{if } \gamma \text{ is not root of characteristic polynomial} \\ x_t = kt\gamma^t & \text{if } \gamma \text{ is simple root of characteristic polynomial} \\ x_t = kt^2\gamma^t & \text{if } \gamma \text{ is double root of characteristic polynomial} \end{cases}$$

Concerning the stability of the equilibrium solutions (if existing), the following conditions hold:

Lemma 3.3.2 *Given a second order linear difference equation with $a_0(t) = a_0$ and $a_1(t) = a_1$ and let $\bar{x}$ be an equilibrium point. Then:*

- *in case of two real distinct roots of the characteristic polynomial, if $|\lambda_1| < 1$ and $|\lambda_2| < 1$ then the equilibrium is globally asymptotically stable;*
- *in case of two real coincident roots of the characteristic polynomial, if $|\lambda| < 1$ then the equilibrium is globally asymptotically stable;*
- *in case of two complex conjugate roots of the characteristic polynomial, if $|\rho| < 1$ then the equilibrium is globally asymptotically stable.*

3.3.3 Dynamical Systems

A system of difference equations is given by a set of n equations:

$$\begin{cases} x_{t+1}^1 = f_1(t,x_t^1,\dots,x_t^n) \\ \vdots \\ x_{t+1}^n = f_n(t,x_t^1,\dots,x_t^n) \end{cases} \tag{3.50}$$

In presence of n first order difference equations, the general solution of the system will be a family of functions depending on n constants, which can be found by specifying the same number of initial conditions. As for systems of differential equations, also in this subsection we focus on the method for solving bidimensional systems ($n = 2$); more in detail, we consider only two special cases of the general formulation in eq. (3.50).

Consider a **system of first order linear difference equations with constant coefficient matrix** ($\mathbf{A(t)} = \mathbf{A}$) and variable vector $\mathbf{b(t)}$, which can be written in matrix form as follows:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} b_1(t) \\ b_2(t) \end{bmatrix} \quad \Rightarrow \quad \mathbf{x_{t+1}} = \mathbf{Ax_t} + \mathbf{b(t)}. \tag{3.51}$$

This problem can be solved using a similar approach to the recursive method described previously in the context of systems of linear differential equations. We may recover, for example, $y(x_{t+1},x_t,t)$ from the first equation (if $a_{12} \neq 0$), then we compute its one step forward value and plug both in the second equation, which becomes a second order linear difference equation in x_t. Now, by using the techniques for second order difference equations we solve for x_t^*. Finally, plug this solution together with its one step forward value back into $y(x_{t+1},x_t,t)$ to obtain y_t^*.

Another way to solve it is to resort to the iterative logic adopted to get an intuition of the solution of single first order difference equations. In this case one gets:

$$\mathbf{x_t^*} = c\mathbf{A^t} + \sum_{k=1}^{t} \mathbf{A^{t-k}b(k-1)} \quad c \in \mathbb{R} \tag{3.52}$$

In the **linear system of first order difference equations with constant coefficients**, which in matrix form is:

$$\mathbf{x_{t+1}} = \mathbf{Ax_t} + \mathbf{b} \tag{3.53}$$

which leads to the simplified version of the solution in eq. (3.39):

$$\mathbf{x_t^*} = c\mathbf{A^t} + (\mathbf{I} - \mathbf{A^t})(\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} \quad c \in \mathbb{R} \tag{3.54}$$

Let $(\bar{x}_t,\bar{y}_t)$ be an equilibrium point for the system (3.53), then the following Theorem gives the conditions for its stability.

Theorem 3.3.3 *[15, p 417] Given the system in eq. (3.51) with arbitrary dimension n and equilibrium point $\bar{\mathbf{x}}_\mathbf{t}$, then a necessary and sufficient condition for the latter to be globally asymptotically stable is that all the eigenvalues of $\mathbf{A}$ have moduli smaller than 1.*

3.3.4 Qualitative Analysis

The **qualitative** analysis of difference equation can be carried out following the same intuition undergoing as that of differential equations, with some slight change in terms of graphical interpretation, due to the discrete framework. Nonetheless, the way of reasoning is similar and the conclusions we can draw are the same.

Definition 3.3.3 *A first order difference equation is said to be* ***autonomous*** *if it does not depend explicitly on the variable t, that is:*

$$x_{t+1} = f(x_t)$$

In the following we restate the definition of equilibrium point:

Definition 3.3.4 *Given a generic difference equation, an* ***equilibrium point*** *is any constant solution for it.*

Given a first order autonomous difference equation, an ***equilibrium point*** *is any constant solution (in this case it is also called fixed point), in particular it is a point such that the value of the function at $t+1$ is equal to the value of the function evaluated at t.*

3.3.4.1 First order single equation

In the case of a single autonomous difference equation, we can represent it on the plane, in which the horizontal axis reports x_t, while the vertical one gives the values $x_{t+1} = f(x_t)$. This representation is called **phase diagram** and is illustrated in Figure (3.6).

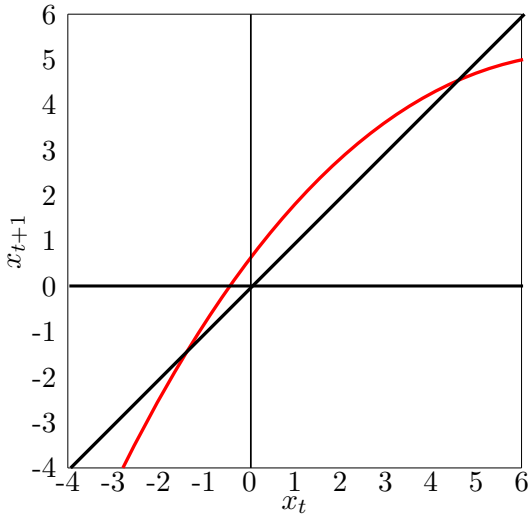


Figure 3.6: Phase diagram for single difference equation.

Recall that an equilibrium point is defined as a constant solution for the equation, which in the case of autonomous difference equations means that $\bar{x} = f(\bar{x})$. In words, all the equilibrium points are located on the first (and third) quadrant bisector (where $x_{t+1} = x_t$) and precisely on the intersection between the curve $f(x_t)$ and the bisector: $x_{t+1} = f(x_t) = x_t$. In the case drawn in Figure (3.6) it is clear that the difference equation admits three equilibrium points.

In order to study the stability of these points, it is necessary to remind that, as t increases one unit at time, the value on the vertical axis at time $t = 1$ (that is x_2) will become a value on the horizontal axis the time after $t = 2$. Following this rule and fixing an initial condition x_0 we can proceed as follows in order to draw a path from the initial condition, which is called the **cobweb**:

- fix x_0 on the horizontal axis and reach vertically the curve $f(x_t)$ at $f(x_0)$;
- move "horizontally" until the bisector is reached. This simply means that we are reporting the value x_1 on the horizontal axis for the next iteration;
- move vertically until the function is reached. This means that we are "updating" the value of the function and computing $x_2 = f(x_1)$;

- iterate the process.

The outcome of this path can be threefold: (i) it converges to the equilibrium point, in which case we say that the point is locally asymptotically stable; (ii) it diverges away from the equilibrium, which is then unstable; (iii) it jumps around the point without getting closer, nor farther, in which case we have a cycle.

The conditions for having a convergent or divergent path are strictly related to the slope of the funtion $f(\cdot)$ at the equilibrium point: it is possible to see in Fig. (3.6) that the first equilibrium point (starting from left) is located on the red curve where it has slope (in absolute value) greater than 1, which is the slope of the bisector. As a consequence for any point in the neighbourhood of the equilibrium the dynamics pushes away from it, therefore the equilibrium is unstable. The converse holds for the second point, where the slope of the curve is smaller than 1 and local asymptotic stability is achieved.

These results are formally stated in the next Theorem:

Theorem 3.3.4 *[15, p. 419] Given a first order autonomous difference equation like $x_{t+1} = f(x_t)$, where f is $\mathcal{C}^1$ in an open interval of the equilibrium point $\bar{x}$. Then:*

- *if $|f'(\bar{x})| < 1$ then $\bar{x}$ is locally asymptotically stable;*
- *if $|f'(\bar{x})| > 1$ then $\bar{x}$ is unstable;*
- *if $|f'(\bar{x})| = 1$ then nothing can be said and it is necessary to check the behaviour of the function locally in the neighbourhood of the equilibrium.*

Example 3.3.1 *Consider the following first order autonomous difference equation:*

$$x_{t+1} = (x_t)^2 \tag{3.55}$$

Draw the phase diagram and study the stability of the equilibrium points after having found them.

We start by drawing the functions $f(x_t) = (x_t)^2$, as results in Figure (3.7).

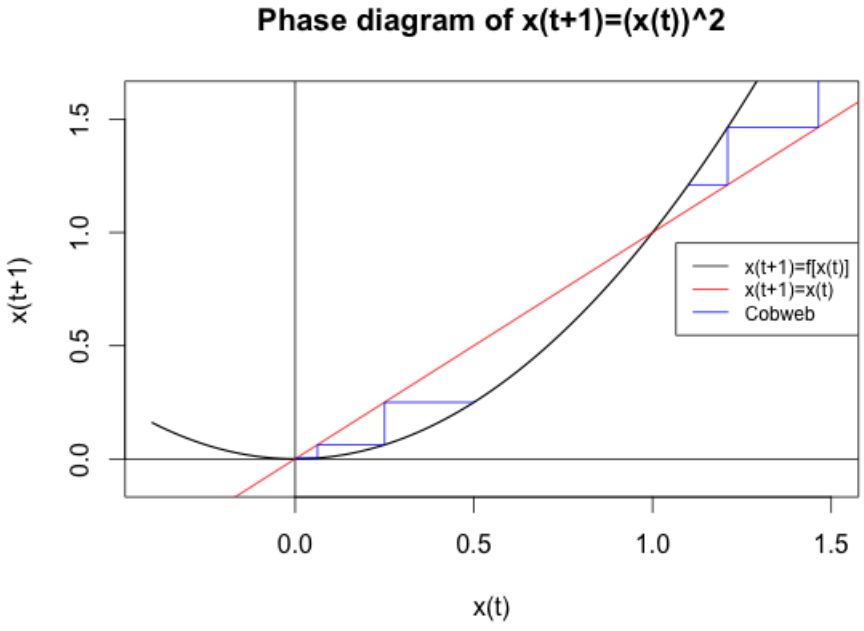


Figure 3.7: Phase diagram of (3.55).

The red line in the graph is the bisector of the first (and third) quadrant, where $x_{t+1} = x_t$: along this line there is no change over t of the values of the function x_t, hence the equilibrium points are located on it, precisely on the intersection with the graph of the function. In this case we have two equilibrium points (0,0) *and* (1,1).

To study the stability, we can draw the "cobweb" for values of x_t in suitable intervals. The blue lines in Fig. (3.7) *represent the "path" of x_t for starting values $x_0 = 0.5$ (right interval of* (0,0)*) and $x_0 = 1.1$ (right interval of* (1,1)*) respectively. It appears evident that in the first case the blue lines converge towards the equilibrium* (0,0)*, while in the other they diverge from* (1,1)*. We can repeat the same procedure taking values in the left interval of each equilibrium point and in this case we will get the same results. We conclude that* (0,0) *is locally asymptotically stable, while* (1,1) *is unstable.*

Another way to check the stability relies on the value of the derivative of the function at the equilibrium: we should check whether its absolute value exceeds or not the slope of the bisector, which is 1*. However, in cases like this, from the shape of the function the answer is evident. Nonetheless, computing the derivative of the function $f(x_t)$ and evaluating at the equilibria yields:*

$$|f'(0)| = 0 < 1 \qquad |f'(1)| = 2 > 1 \tag{3.56}$$

then, by applying theorem 3.3.4 we get that the first equilibrium is locally asymptotically stable, while the second one is unstable.

■

3.3.4.2 First order system of equations

The study of systems of difference equations is carried on in this subsection; we start by giving a formal definition, similar to Def. (3.3.3).

Definition 3.3.5 *Given a system of first order difference equations, it is said to be* ***autonomous*** *if:*

$$\begin{cases} x^1_{t+1} = f_1(x^1_t, \dots, x^n_t) \\ \vdots \\ x^n_{t+1} = f_n(x^1_t, \dots, x^n_t) \end{cases}$$

that is, none of the difference equations depends explicitly on the variable t.

As for systems of differential equations, also in this subsection we focus on (2×2) systems. The concept of equilibrium point in a system is a generalization of that for single equations in Def. (3.3.4):

Definition 3.3.6 *Given an autonomous system of first order difference equations, an* ***equilibrium point*** *(or stationary point) is any solution of the system such that:*

$$\begin{cases} x^1_{t+1} = x^1_t = \bar{x}^1 \\ \vdots \\ x^n_{t+1} = x^n_t = \bar{x}^n \end{cases}$$

where $\bar{\mathbf{x}}$ is the equilibrium point.

For a generic (2×2) system like:

$$\begin{cases} x_{t+1} = f(x_t, y_t) \\ y_{t+1} = g(x_t, y_t) \end{cases} \tag{3.57}$$

it can be proved that the equilibrium point, when existing, is located at the intersection of the two curves $f(x_t,y_t) = \bar{x}^1$ and $g(x_t,y_t) = \bar{x}^2$, which are called the **nullclines** (or stationary loci) of the system. These represent the set of points where the first and second difference equations are constant, respectively. We can represent them on the x_t-y_t plane, which is called the phase diagram. Again, this is different from that used in the single equation case: the former has two axis, each of which is one of the two functions of the system (x_t,y_t), while the latter has on the horizontal axis the function (x_t) and the vertical one the value of the function one period ahead (x_{t+1}).

A possible phase diagram for a bidimensional system is drawn in Figure (3.8), where the blue curve represent the $x_{t+1} = x_t$ nullcline (or stationary locus), while the red curve is the $y_{t+1} = y_t$ nullcline. These curves are graphically drawn by solving $f(x_t,y_t) = x_t$ and $g(x_t,y_t) = y_t$ respectively and drawing the resulting curves as $x_{t+1} = x_t \to y_t = h_f(x_t)$ and $y_{t+1} = y_t \to y_t = h_g(x_t)$. For simplicity the same curves as in Fig. (3.4) have been reported also in the system of difference equations, since the method and logic to be used are unaffected by the shape of the graph.

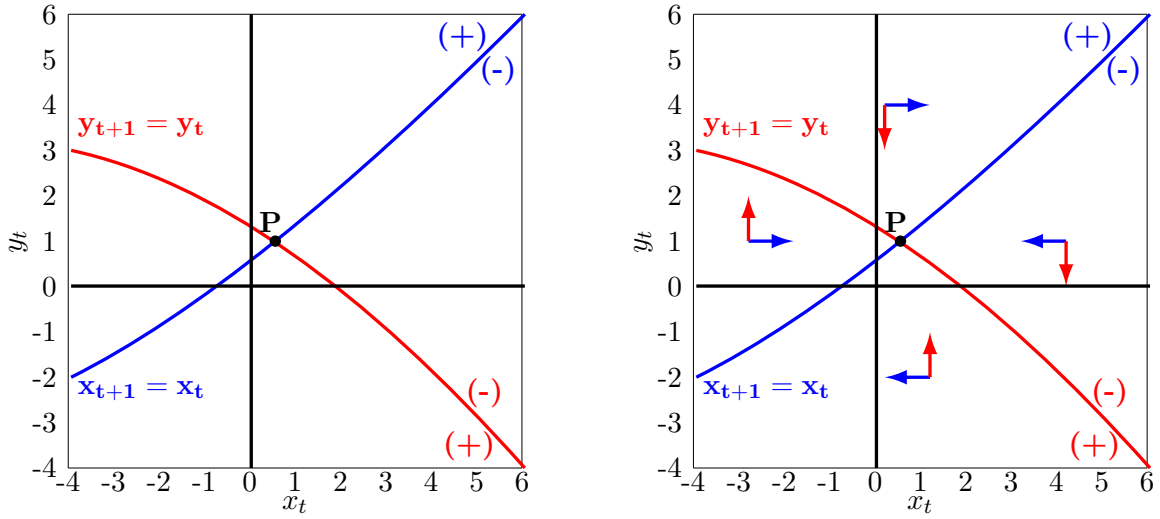


Figure 3.8: Phase diagram for system of two difference equations.

The interpretation of the graph is exactly the same as that in Fig. (3.4) for systems of differential equations, and the procedure for putting the signs "+" and "−" are equal too, therefore we remind to subsection § 3.2.4 for more details.

Here we report again the list of possible cases for the stability of the equilibrium, to stress the fact that the interpretation and conclusions that can be drawn in both continuous (differential) and discrete (difference) domain in terms of qualitative analysis are the same. The equilibrium point is given by the intersection of the two nullclines: here we have only one equilibrium point, P. The stationarity of this point can be studied by checking the direction of the arrows in the diagram:

- if all the pairs of arrows are pointing towards the equilibrium point, then it is globally asymptotically stable;
- if two pairs of arrows (in opposite regions) are pointing towards the equilibrium point, while the other two pairs (in the other two opposite regions) are pointing away from it, then the equilibrium is stable saddle and there exists a saddlepath leading to it;
- if all the pairs of arrows are pointing away from the equilibrium point, then it is unstable.

Clearly, the example drawn in Fig. (3.8) shows that P is a globally asymptotically stable equilibrium point for the system.

Problem	Objective	Time	Solution
Static optim	function $f(\mathbf{x})$	*none*	vector of scalars $\mathbf{x}^*$
Dynamic optim	functional $F(t,x(t),\ldots)$	discrete or continuous	function $x^*(t)$

Table 3.3: Comparison static vs dynamic optimization.

3.4 Dynamic Optimization

In Section 6.3 we focused on the optimization of a function in a static framework, then in Sections (3.2) and (3.3) the general theory of differential and difference equations has been presented in light of its importance for solving problems of dynamic optimization, which are the core of this Section. First of all, the main differences between the static and the dynamic optimization are highlighted in Table (3.3): the objective to be optimized is no more a function of n unknown variables, but a functional, that is a function F of functions $x(t)$, both depending on the variable t (which is generally referred as "time", but can be whatever else). As a consequence, it is not surprising that the solution is a function $x^*(t)$ of the unknown t, rather than a vector of scalars.

The dynamic optimization, like the static one, gather together a wide variety of problems and in the following subsection we are going to review two of them: the calculus of variations in its most simple setup and the problems of deterministic dynamic programming which can be solved via the backward induction method. In particular, in the first case we will consider continuous time and differential equations, therefore the notation will follow that of Section 3.2, while in the second subsection we will focus on discrete time problems, hence exploiting the notation of Section 3.3.

3.4.1 Calculus of Variations

In the calculus of variations we aim at maximizing (or minimizing) an area under a curve, subject to some conditions regarding the initial and final values of the curve, where the specific form of the curve has to be determined. Since the area under a curve between two extrema is defined as by means of a definite integral, the general formulation of the problem is:

$$\begin{cases} \max \displaystyle\int_{t_0}^{t_1} F(t,x,\dot{x})\,\mathrm{d}t \\ s.t. \\ \quad x(t_0) = k_1 \quad k_1 \in \mathbb{R} \\ \quad x(t_1) = k_2 \quad k_2 \in \mathbb{R} \end{cases} \tag{3.58}$$

Here t_0 and t_1 are the extremum points, while $x(t_0)$ and $x(t_1)$ are the corresponding values that the solution function must attain at those points (also called initial and terminal or final conditions, respectively). Other alternative terminal conditions are possible:

$$\left.\begin{array}{l} x(t_1) \geq k_2 \\ x(t_1)\ \ free \end{array}\right\} \quad \text{additional condition for determining particular solution} \tag{3.59}$$

It is possible to think about the problem in Eq. (3.58) as the "dynamic equivalent" of the static unconstrained optimization problem[9]. By the same token, a problem which is similar to the equality constrained static optimization is the isoperimetric problem: in this case an equality constraint is added in the form of another definite integral (over the same interval):

$$\int_{t_0}^{t_1} G(t,x,\dot{x})\,\mathrm{d}t = b. \tag{3.60}$$

[9]Of course, some constraints must be imposed in the dynamic context on the function $x(t)$ for the problem to be valid.

Remark 3.4.1 *In the calculus of variation class of problems there does not exist simple conditions under which the existence of a solution is guaranteed, as was the case for static optimization by means of the Weierstrass Theorem. As a consequence we may proceed by the study of the necessary conditions and then the sufficient conditions hoping that a solution really exists.*

In the basic class of problems like (3.58) we make the following preliminary assumptions:

- F is continuous as function of $(t,x,\dot{x})$ and of class $\mathcal{C}^1$ as a function of $(x,\dot{x})$;
- the admissible functions $x(t)$ are continuously differentiable defined on $[t_0,t_1]$ and satisfy the boundary conditions $x(t_0) = x_0$ and $x(t_1) = x_1$ (or one of the other cases);
- for simplifying the notation we will use: $F'_1 = \dfrac{\partial F(t,x,\dot{x})}{\partial t}$, $F'_2 = \dfrac{\partial F(t,x,\dot{x})}{\partial x}$, $F'_3 = \dfrac{\partial F(t,x,\dot{x})}{\partial \dot{x}}$.

The first order necessary condition are summarized by the Euler equation and stated in the following theorem.

Theorem 3.4.1 (First Order Necessary Conditions [15, p. 294]) *Suppose $x^*(t)$ is an optimum for:*

$$\begin{cases} \max \displaystyle\int_{t_0}^{t_1} F(t,x,\dot{x})\,\mathrm{d}t \\ s.t. \\ \quad x(t_0) = k_1 \quad k_1 \in \mathbb{R} \\ \quad x(t_1) = k_2 \quad k_2 \in \mathbb{R} \end{cases}$$

*Then $x^*(t)$ is a solution of the **Euler equation**:*

$$\frac{\partial F(t,x,\dot{x})}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial F(t,x,\dot{x})}{\partial \dot{x}}\right) = 0.$$

Notice that these conditions hold irrespective of the type of optimization problem. Under a more stringent assumption it is possible to state also the second order necessary condition, also known as Legendre's condition.

Theorem 3.4.2 (Second Order Necessary Condition - Legendre) *Suppose the same assumptions of Theorem (3.4.1) hold. Suppose also that F is $\mathcal{C}^2$ as function of $(t,x,\dot{x})$. Then $x^*(t)$ satisfies the Legendre's condition:*

$$\frac{\partial^2 F(t,x,\dot{x})}{\partial \dot{x}^2} \leq 0 \quad \forall\, t \in [t_0,t_1].$$

In a setup like this, where there are no theorems assuring the existence of a solution, it is even more important to have a general statement about the sufficient conditions for optimality. This is the purpose of the following theorem.

Theorem 3.4.3 *Let $F(t,x,\dot{x})$ be a concave (convex) function on $(x,\dot{x})$ for every $t \in [t_0,t_1]$. If $x^*(t)$ satisfies the Euler equation and the boundary conditions then x^* is a global maximum (minimum) solution for the problem in (3.58).*

Remark 3.4.2 *The concavity (convexity) requirement for the functional F is with respect to a subset of its arguments, namely $(x,\dot{x})$ and must hold only in a subset of all the possible values of the variable t. In practice, one may check this condition by looking at the definiteness of Hessian matrix formed by taking the second order partial derivatives of the functional F with respect to x and $\dot{x}$, keeping in mind that $t \in [t_0,t_1]$.*

We can now turn to the analysis of some particular cases, like the isoperimetric problem. In this framework we make the assumption that both the functionals F and G are of class $\mathcal{C}^2$ and that the coefficient expressing the value of the constraint is a real scalar (i.e. $b \in \mathbb{R}$). The necessary conditions are given in the next theorem; it is evident that the procedure to find out a candidate solution is equivalent to the previous one (standard problem), using a different functional (L instead of F).

Theorem 3.4.4 *let $x^*(t)$ be a continuously differentiable solution of the isoperimetric problem:*

$$\begin{cases} \max \displaystyle\int_{t_0}^{t_1} F(t,x,\dot{x})\,\mathrm{d}t \\ s.t. \\ \quad \displaystyle\int_{t_0}^{t_1} G(t,x,\dot{x})\,\mathrm{d}t = b \\ \quad x(t_0) = k_1 \quad k_1 \in \mathbb{R} \\ \quad x(t_1) = k_2 \quad k_2 \in \mathbb{R} \end{cases}$$

If it is not a stationary point of the constraint, then there exists a constant λ^* such that $(x^*(t),\lambda^*)$ satisfy the constraints and the Euler equation:

$$\frac{\partial L(t,x,\dot{x},\lambda)}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L(t,x,\dot{x},\lambda)}{\partial \dot{x}}\right) = 0 \quad \forall\, t \in [t_0,t_1]$$

where $L = F - \lambda G$

Finally, we state the necessary conditions to be met in case of alternative terminal conditions.

Theorem 3.4.5 *Let $x^*(t)$ be a continuously differentiable solution of the problem:*

$$\begin{cases} \max\limits_{x(t)} \displaystyle\int_{t_0}^{t_1} F(t,x,\dot{x})\,\mathrm{d}t \\ s.t. \\ \quad x(t_0) = x_0 \end{cases}$$

with terminal conditions (one of the two):

1. $x(t_1) \geq x_1$
2. $x(t_1)$ *free*

Then $x^*(t)$ solves the Euler equation and the initial condition, moreover it solves, respectively:

1. $\dfrac{\partial F(t,x,\dot{x})}{\partial \dot{x}} \leq 0$ *at* $t = t_1$, *with equality if* $x^*(t_1) > x_1$
2. $\dfrac{\partial F(t,x,\dot{x})}{\partial \dot{x}} = 0$ *at* $t = t_1$

3.4.2 Dynamic Programming

In this section we turn to the discrete time environment. The type of problems we are going to address is like the following: consider the purpose of maximizing the sum over a finite number (T) of periods, of a reward function ($f_t(t,x_t,u_t)$). In this context x_t represent the state variable, which cannot be influenced by external actions, while u_t is the control variable (or action), which by definition is the variable that is chosen in order to carry out the optimization problem. We assume that an initial value of the state variable is given (x_1^*). Finally, the link between each period is provided by the dynamic

constraint $x_{t+1} = g_t t,x_t,u_t$, which specifies the influence that the control exerts on the state (in the same period) and gives the resulting value of the state in the next period. In formal terms, we have:

$$\begin{cases} \max \sum_{t=1}^{T} f_t(t,x_t,u_t) \\ s.t. \\ \quad x_{t+1} = g_t(t,x_t,u_t) \quad \forall t = 1,\dots,T-1 \\ \quad u_t \in \Phi_t \quad \forall t = 1,\dots,T-1 \\ \quad x_1^* = k \quad k \in \mathbb{R} \end{cases} \tag{3.61}$$

It is important to stress that the control variable takes values in a given set $u_t \in \Phi_t$ and that the maximization is carried out with respect to the control variable only: the state is affected indirectly, via the dynamic constraint.

The next theorem, known as Bellman principle, is one of theoretical foundations of the backwards induction method for solving deterministic finite horizon Markov problems[10].

Theorem 3.4.6 (Bellman principle) *Assume that, for all t the reward function f_t is continuous and bounded on $S \times A$; the objective function g_t is continuous on $S \times A$ and the correspondence Φ_t is continuous and compact-valued on S. Then the dynamic programming problem admits an optimal solution. Moreover, the value function $V_t(\cdot)$ of the $(T-t+1)$-period continuation problem satisfies the following equation:*

$$V_t(x_t) = \max_{u_t \in \Phi_t(s)} \{f_t(x_t,u_t) + V_{t+1}(g_t(x_t,u_t))\} \quad \forall\, t \in \{1,\dots,T\},\ \forall\, s \in S$$

The **backward induction method** (see Algorithm (4)) is an iterative procedure which starts from the last period and maximizes the reward function with respect to the control variable (u_T), then the optimal control is plugged in f_T in order to obtain the value function at the last period ($V_T(x_T)$). Next, we exploit the dynamic constraint in the opposite direction, that is we substitute $x_{t+1} = g_t(x_t,u_t)$, obtaining $V_{t+1}(x_t,u_t)$. Then we add this "backward updated" value function to the reward function and we maximise again with respect to the control. This procedure is repeated until the first period, obtaining a sequence of optimal controls $\{u_t^*\}_{t=1}^T$. Finally, we exploit the (given) initial value of the state x_1^* together with the dynamic constraint and the sequence of optimal controls to obtain the sequence of optimal states $\{x_t^*\}_{t=1}^T$.

Algorithm 4 Basic Backward Induction Method

procedure OPTIMIZATION(f_t,g_t,Φ_t,x_1^*,T)
  $u_T^* \leftarrow \arg\max_{u_t \in \Phi_t} \{f_T(x_T) : x \in \mathcal{D}\}$
  $V_T(x_T) \leftarrow f_T(x_T,u_T^*)$
  for $t = T-1,\dots,1$ **do** ▷ obtain sequence of optimal controls
    $V_t \leftarrow f_t(x_t,u_t) + V_{t+1}(g_t(x_t,u_t))$ ▷ exploit $x_{t+1} = g_t(x_t,u_t)$
    $u_t^* \leftarrow \arg\max_{u_t \in \Phi_t} \{V_t\}$
    $V_t \leftarrow f_t(x_t,u_t^*)$
  end for
  for $t = 2,\dots,T$ **do** ▷ obtain sequence of optimal states
    $x_t^* \leftarrow g_{t-1}(x_{t-1}^*,u_{t-1}^*)$
  end for
  return Solution: $(\{x_t^*\}_{t=1}^T, \{u_t^*\}_{t=2}^T)$
end procedure

[10]Other fundamental results prove that a solution to this kind of problems actually exists and that the backward induction method actually converges to that solution. These theorems are not reported here.

This method highlights that in any given period we need to solve the static optimization problem:

$$\max_{u_t \in \Phi_t} \{f_t(t,x_t,u_t) + V_{t+1}(t,x_t,u_t)\} \tag{3.62}$$

which may be unconstrained or constrained according to the set (Φ_t) in which the control variable takes values; therefore, we can undertake this problem using all the tools described in Section 3.1.

Remark 3.4.3 *The problem is Markovian since when optimizing Eq. (3.62) all the information about the past history (that is, the future, since it is a backward induction procedure starting from the last period) is carried on by the function V_{t+1}, which is a one-step-ahead value function. In other words, there is no need to consider all future reward functions f_s $s = t+1,\ldots,T$, but is suffices to consider V_{t+1}, which "encompasses them all".*

Chapter 4

Exercises with Solutions

In this Chapter we present, analyse and carry out step by step some exercises following the analytical approach outlined in Chapter 3.

4.1 Static Optimization: Unconstrained Optimization

Exercise 4.1.1 *Find out whether the following function admits global/local maxima/minima:*

$$f(x,y,z) = e^{x+y+z} - x + y^2 + z^2 \tag{4.1}$$

SOLUTION: *First of all, the function is defined on an unrestricted domain equal to the space* $\mathbb{R}^3$*: since this space is open and unbounded, the Weierstrass Theorem (sufficient conditions for having maximum and minimum points) does not apply. In order to prove that it does NOT admit* ***global*** *maxima (minima) it is sufficient to demonstrate that taking the limit of the function as one (or more) variable(s) approaches infinity (or a suitable other point), while the remaining variables are taken fixed, then the objective function approaches* $+\infty$ $(-\infty)$*. In this case it holds:*

$$\lim_{x\to+\infty} f(x,y,z) = -\infty \qquad \lim_{y\to+\infty} = +\infty \tag{4.2}$$

hence the objective function is unbounded and does not admit neither global maxima nor global minima.

We now look for local optimum points: according to FOCs they must be searched among the critical points, hence we compute the gradient:

$$\nabla f(x,y,z) = \begin{bmatrix} e^{x+y+z} - 1 \\ e^{x+y+z} + 2y \\ e^{x+y+z} + 2z \end{bmatrix} \tag{4.3}$$

from which we can find the unique critical point by simple computation: $(x^*,y^*,z^*) = \big(1, -1/2, -1/2\big)$*. We now compute the Hessian matrix and study its properties (definitess/semidefiniteness), in order to apply the sufficient SOC (or to get some additional information from the necessary SOC, at least).*

$$\mathcal{H}(x,y,z) = \begin{bmatrix} e^{x+y+z} & e^{x+y+z} & e^{x+y+z} \\ e^{x+y+z} & e^{x+y+z} + 2 & e^{x+y+z} \\ e^{x+y+z} & e^{x+y+z} & e^{x+y+z} + 2 \end{bmatrix} \qquad \mathcal{H}(1,-1/2,-1/2) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}. \tag{4.4}$$

We check the definiteness of the Hessian evaluated in the critical point by means of the leading principal minors method[1]*. Since all the leading principal minors are strictly positive:*

$$\det(1) > 0 \quad \det\left(\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}\right) = 2 > 0 \quad \det(\mathcal{H}) = 4 > 0$$

[1] This is not the only method; for example one might have computed the eigenvalues

then the matrix is positive definite and, applying the sufficient SOC we conclude that the critical point $(x^*,y^*,z^*) = \left(1,-1/2,-1/2\right)$ *is a local minimum point.*

□

Exercise 4.1.2 *Compute the stationary points of the following function and characterize them using the SOCs:*

$$f(x,y) = (y-2)^2 \ln^2(x+1) \tag{4.5}$$

SOLUTION: *start by noticing that the Weirstrass Theorem does not apply since the domain of this function:*

$$\mathcal{D} = \{(x,y) \in \mathbb{R}^2 \ : \ x > -1\}$$

is open and unbounded.

Notice that the objective function in eq. (4.5) is unbounded: we can see this by computing its limit as letting one variable at time go to $+\infty$ *and taking the other fixed*[2]*:*

$$\lim_{x\to+\infty} f(x,y) = +\infty = \lim_{y\to+\infty} f(x,y) \tag{4.6}$$

$$\lim_{x\to-1^+} f(x,y) = -\infty \tag{4.7}$$

from which we can conclude that the function does not admit neither global maxima nor global minima.

Now, continue by computing the gradient:

$$\nabla f(x,y) = \begin{bmatrix} 2(y-2)^2 \frac{\ln(x+1)}{x+1} \\ 2(y-2)\ln^2(x+1) \end{bmatrix} \tag{4.8}$$

Notice that the second entry (but we may work alternatively with the first one) is null either for $x = 0$ *or* $y = 2$*, so plug in the first to get the two sets of stationary points*[3]*:*

$$(x_1^*,y_1^*) = \begin{cases} x = 0 \\ y \in \mathbb{R} \end{cases} \qquad (x_2^*,y_2^*) = \begin{cases} x \in \mathbb{R} \\ y = 2 \end{cases} \ . \tag{4.9}$$

We can represent graphically these two sets of stationary points on the plane as two lines, as drawn in Figure (4.1).

[2]Notice that this strategy is a general one for checking the existence of global maxima/minima when the Weirstrass Theorem does not apply. It may be useful to take limits letting just one variable at time approach either $\pm\infty$ or "particular" points outside the domain. It is the same procedure that one may use when studying one variable functions.

[3]Notice that in general terms we do have sets of stationary points. The case of single-valued sets, that is a precise value for (x^*,y^*), is just a particular case in which the set of stationary points consists only of a single point.

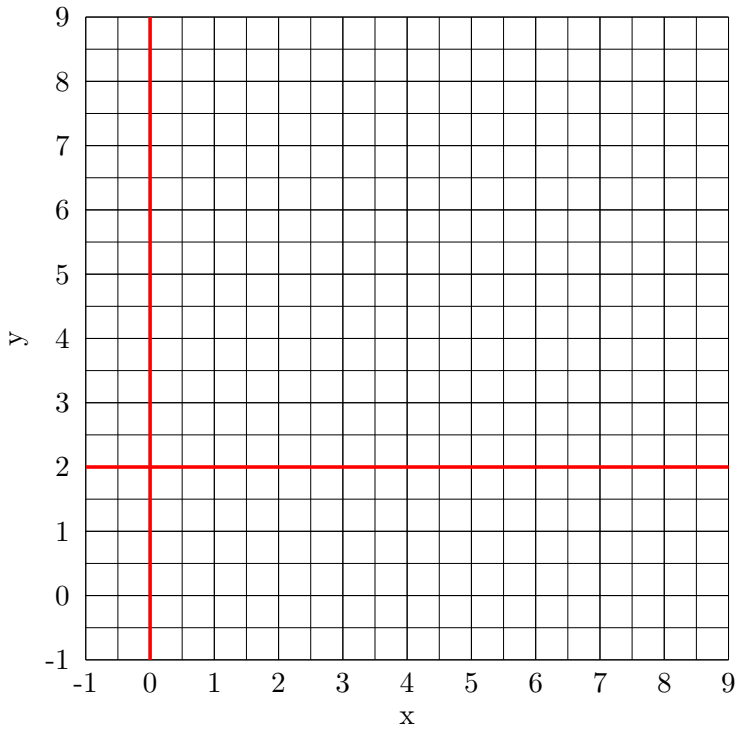


Figure 4.1: Stationary loci (i.e. sets) of problem (4.5).

The Hessian matrix is:

$$\mathcal{H}(x,y) = \begin{bmatrix} 2\,(y-2)^2\,\frac{1-\ln(x+1)}{(x+1)^2} & 4\,(y-2)\,\frac{\ln(x+1)}{x+1} \\ 4\,(y-2)\,\frac{\ln(x+1)}{x+1} & 2\ln^2(x+1) \end{bmatrix} \tag{4.10}$$

Now, we plug in the stationary points found in eq. (4.9) to obtain:

$$\mathcal{H}(0,y) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \mathcal{H}(x,2) = \begin{bmatrix} 0 & 0 \\ 0 & 2\ln^2(x+1) \end{bmatrix} \tag{4.11}$$

both of which have determinant equal to zero. As a consequence neither of them is positive definite nor negative definite and the sufficient SOCs do not apply in this case. Hence we conclude that, in order to assess the nature of the stationary points, it is necessary a "local study" of the function in the neighbourhood of each stationary locus.

First, consider the locus (x_1^*,y_1^*): at this point the objective function is equal to zero, but if we slightly change the value of x we obtain (formally, we add/subtract a small positive scalar ϵ):

$$f(x_1^*+\epsilon,y_1^*) > f(x_1^*,y_1^*) = 0 \tag{4.12}$$
$$f(x_1^*-\epsilon,y_1^*) > f(x_1^*,y_1^*) = 0. \tag{4.13}$$

In a similar way, one could check the second stationary locus (x_2^*,y_2^*):

$$f(x_2^*,y_2^*+\epsilon) > f(x_2^*,y_2^*) = 0 \tag{4.14}$$
$$f(x_2^*,y_2^*-\epsilon) > f(x_2^*,y_2^*) = 0. \tag{4.15}$$

As a consequence we conclude that both stationary loci are sets of local maxima.

□

4.2 Static Optimization: Equality Constrained Optimization

Exercise 4.2.1 *Consider the following maximization problem and check whether, and in which set, the constraint qualification conditions are satisfied.*

$$\begin{cases} \max\limits_{x,y,z} \quad xyz \\ s.t. \\ \qquad x^2+y^2=1 \\ \qquad x+z=1 \end{cases} \tag{4.16}$$

SOLUTION: *the constraint qualification conditions require that the rank of the Jacobian matrix, evaluated at the stationary point of the Lagrangian, is equal to the number of constraints, in other words, given m constraints and a vector of variables $\mathbf{x}$, with stationary point $\mathbf{x}^*$:*

$$\rho\left[\mathbf{J}\left(\mathbf{x}^*\right)\right]=m$$

First of all, compute the Jacobian associated to the problem (4.16)[4]*. Note that, for the purposes of computing its rank the transposition of $\mathbf{J}$ has no effects, that is:*

$$\rho\left[\mathbf{J}\left(\mathbf{x}\right)\right]=\rho\left[\mathbf{J}\left(\mathbf{x}\right)'\right]$$

As a consequence, without loss of generality, choose:

$$\mathbf{J}\left(x,y,z\right)=\begin{bmatrix} \frac{\partial g_1}{\partial x} & \frac{\partial g_2}{\partial x} \\ \frac{\partial g_1}{\partial y} & \frac{\partial g_2}{\partial y} \\ \frac{\partial g_1}{\partial z} & \frac{\partial g_2}{\partial z} \end{bmatrix}=\begin{bmatrix} 2x & 1 \\ 2y & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.17}$$

Definition 4.2.1 (Rank) *The rank of matrix is maximum order of the (NOT "all"!) square submatrix with determinant different from zero. Alternatively, it is the maximum number of linearly independent rows (or columns).*

Since the Jacobian in (4.17) is a (3x2) matrix, it has exactly 3 square submatrices of order 2 (clearly, none of order 3), we MUST check all the determinant of all of them in order to assess the rank of J. According to the definition, it is sufficient that one of these three submatrices has determinant different from zero to say that the rank of the Jacobian is two (which is equal to the number of equations m). The three submatrices are:

$$\mathbf{M_1}=\begin{bmatrix} 2x & 1 \\ 2y & 0 \end{bmatrix} \qquad \mathbf{M_2}=\begin{bmatrix} 2y & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{M_3}=\begin{bmatrix} 2x & 1 \\ 0 & 1 \end{bmatrix} \tag{4.18}$$

whose determinants are, respectively:

$$|\mathbf{M_1}|=-2y \qquad |\mathbf{M_2}|=2y \qquad |\mathbf{M_3}|=2x. \tag{4.19}$$

The first two determinants are zero if $y=0$, while the third when $x=0$. Now, we MUST consider all the possible combinations that may arise in order to have a clear picture of the region when the constraint qualification conditions are met. As a consequence, exploiting the Definition above:

a) $x=0 \wedge y=0 \quad \Rightarrow \quad |\mathbf{M_1}|=0 \quad |\mathbf{M_2}|=0 \quad |\mathbf{M_3}|=0$
$\Rightarrow \quad \rho\left[\mathbf{J}(0,0,z)\right]=1<m \quad \forall z \in \mathbb{R}$

[4]Recall that the Jacobian is a matrix of first order partial derivatives, with dimension (nxm), where m is the number of constraints and n is the number of unknowns of the problem.

b) $x = 0 \wedge y \neq 0 \quad \Rightarrow \quad |\mathbf{M_1}| \neq 0 \quad |\mathbf{M_2}| \neq 0 \quad |\mathbf{M_3}| = 0$
$\Rightarrow \quad \rho\left[\mathbf{J}(0,y,z)\right] = 2 = m \quad \forall y \in \mathbb{R} \setminus \{0\},\ \forall z \in \mathbb{R}$

c) $x \neq 0 \wedge y = 0 \quad \Rightarrow \quad |\mathbf{M_1}| = 0 \quad |\mathbf{M_2}| = 0 \quad |\mathbf{M_3}| \neq 0$
$\Rightarrow \quad \rho\left[\mathbf{J}(x,0,z)\right] = 2 = m \quad \forall x \in \mathbb{R} \setminus \{0\},\ \forall z \in \mathbb{R}$

d) $x \neq 0 \wedge y \neq 0 \quad \Rightarrow \quad |\mathbf{M_1}| \neq 0 \quad |\mathbf{M_2}| \neq 0 \quad |\mathbf{M_3}| \neq 0$
$\Rightarrow \quad \rho\left[\mathbf{J}(x,y,z)\right] = 2 = m \quad \forall x \in \mathbb{R} \setminus \{0\},\ \forall y \in \mathbb{R} \setminus \{0\}, \forall z \in \mathbb{R}$

We can now define the region in which the constraint qualification conditions hold as follows:

$$\mathcal{D} = \{(x,y,z) \in \mathbb{R}^3 \ : \ x \neq 0 \vee y \neq 0\} \tag{4.20}$$

Now that we have defined the "acceptance" region, once we have computed the stationary points of the Lagrangian we must check whether they lie inside $\mathcal{D}$: if $(x^*,y^*,z^*) \in \mathcal{D}$, the First Order Necessary Conditions (Lagrange Theorem) are satisfied and we can proceed, in the other case, any stationary point $(x^*,y^*,z^*) \notin \mathcal{D}$ does NOT satisfy the necessary conditions for max/min, hence we should keep in mind that it cannot be neither a maximum nor a minimum for the problem in (4.16).

□

Exercise 4.2.1 (Cont'd) *Given the preliminary results from exercise 3a), find the stationary points of the problem in (4.16).*
SOLUTION: *since we have already determined the region* D *in which the constraint qualification conditions hold, we need to find out the stationary points of the Lagrangian. Notice that, since $\frac{\partial g_2(x,y,z)}{\partial z} \neq 0$ we can apply the implicit function theorem and recover from the second constraint:*

$$x + z = 1 \quad \Rightarrow \quad z = 1 - x \tag{4.21}$$

which we may substitute both in the objective function and in the other constraints, obtaining the "new" problem[5]

$$\begin{cases} \max\limits_{x,y} & xy(1-x) \\ s.t. & \\ & x^2 + y^2 = 1 \end{cases} \tag{4.22}$$

In this way we "reduce" the dimensionality of the optimization problem, allowing in general to get easier computations. In any case, the solution of the original problem involves <u>all</u> the variables, therefore, given a solution to the "new" problem (4.22) (x^*, y^*) we need to compute back z^* from (4.21), obtaining the solution to the original problem (4.16) as (x^*, y^*, z^*).

The Lagrangian associated to (4.22) is[6]*:*

$$\mathcal{L}(x,y,\lambda) = xy - x^2y - \lambda\left[x^2 + y^2 - 1\right] \tag{4.23}$$

with gradient:

$$\nabla\mathcal{L}(x,y,\lambda) = \begin{bmatrix} y - 2xy - 2\lambda x \\ x - x^2 - 2\lambda y \\ x^2 + y^2 - 1 \end{bmatrix} \tag{4.24}$$

Now, you may solve it by setting the gradient equal to zero.

[5] Notice that this is just a suggestion, it is correct also to work directly on (4.16), without any substitution.

[6] Notice that, since we are dealing with equality constraints, nothing will change if we multiply the Lagrange multipliers by $+1$ instead of -1. It is <u>different</u> in the case of inequality constraints (see Karush-Khun-Tucker Theorem).

A different (but equally correct) way to proceed consists in deriving the Lagrangian directly from eq. (4.16):

$$\mathcal{L}(x,y,z,\lambda_1,\lambda_2) = xyz - \lambda_1\left[x^2+y^2-1\right] - \lambda_2\left[x+z-1\right] \tag{4.25}$$

then the FOC are derived by solving the system:

$$\begin{cases} yz - 2\lambda_1 x - \lambda_2 = 0 \\ xz - 2\lambda_1 y = 0 \\ xy - \lambda_2 = 0 \\ x^2+y^2 = 1 \\ x+z = 1 \end{cases} \Rightarrow \begin{cases} yz - 2\lambda_1(1-z) - (1-z)y = 0 \\ (1-z)z = 2\lambda_1 y \\ \lambda_2 = (1-z)y \\ (1-z)^2+y^2 = 1 \\ x = 1-z \end{cases} \tag{4.26}$$

In this case, in order to recover λ_1 from the seond equation, it is necessary to assume that $y \neq 0$. ***Remember this and all the assumptions*** *we make, because later on we will need to check what happens when they are not met.*

Then, ***HP1:*** $\mathbf{y \neq 0}$ *yields:*

$$\begin{cases} yz - \frac{(1-z)^2 z}{y} - (1-z)y = 0 \\ \lambda_1 = \frac{(1-z)z}{2y} \\ \lambda_2 = (1-z)y \\ (1-z)^2+y^2 = 1 \\ x = 1-z \end{cases} \Rightarrow \begin{cases} y^2 z - (1-z)^2 z - (1-z)y^2 = 0 \\ \lambda_1 = \frac{(1-z)z}{2y} \\ \lambda_2 = (1-z)y \\ (1-z)^2+y^2 = 1 \\ x = 1-z \end{cases} \Rightarrow \begin{cases} 2y^2 z - (1-z)^2 z - y^2 = 0 \\ \lambda_1 = \frac{(1-z)z}{2y} \\ \lambda_2 = (1-z)y \\ y^2 = 1-(1-z)^2 \\ x = 1-z \end{cases} \tag{4.27}$$

$$\begin{cases} y^2(2z-1) = (1-z)^2 z \\ \lambda_1 = \frac{(1-z)z}{2y} \\ \lambda_2 = (1-z)y \\ (1-z)^2+y^2 = 1 \\ x = 1-z \end{cases} \tag{4.28}$$

For obtaining y^2 from the first equation it is required to assume: ***HP2:*** $\mathbf{z \neq \frac{1}{2}}$ *(so up to now we have made two assumptions). Then:*

$$\begin{cases} y^2 = \frac{(1-z)^2 z}{2z-1} \\ \lambda_1 = \frac{(1-z)z}{2y} \\ \lambda_2 = (1-z)y \\ (1-z)^2 + \frac{(1-z)^2 z}{2z-1} = 1 \\ x = 1-z \end{cases} \Rightarrow \begin{cases} y^2 = \frac{(1-z)^2 z}{2z-1} \\ \lambda_1 = \frac{(1-z)z}{2y} \\ \lambda_2 = (1-z)y \\ (2z-1)(1-2z+z^2) + (1-2z+z^2)z = 2z-1 \\ x = 1-z \end{cases} \tag{4.29}$$

$$\begin{cases} y^2 = \frac{(1-z)^2 z}{2z-1} \\ \lambda_1 = \frac{(1-z)z}{2y} \\ \lambda_2 = (1-z)y \\ 2z - 4z^2 + 2z^3 - 1 + 2z - z^2 + z - 2z^2 + z^3 = 2z-1 \\ x = 1-z \end{cases} \Rightarrow \begin{cases} y^2 = \frac{(1-z)^2 z}{2z-1} \\ \lambda_1 = \frac{(1-z)z}{2y} \\ \lambda_2 = (1-z)y \\ 3z^3 - 7z^2 + 3z = 0 \\ x = 1-z \end{cases} \tag{4.30}$$

The third order polynomial of z has the following roots:

$$z_1 = 0 \; z_2 = \frac{7-\sqrt{13}}{6} \; z_3 = \frac{7+\sqrt{13}}{6}$$

which have to be plugged into eq. (4.30) to get the following solutions:

$$\begin{bmatrix} x_1^* \\ y_1^* \\ z_1^* \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} x_2^* \\ y_2^* \\ z_2^* \end{bmatrix} = \begin{bmatrix} \frac{-1+\sqrt{13}}{6} \\ \frac{11+\sqrt{13}}{18} \\ \frac{7-\sqrt{13}}{6} \end{bmatrix} \qquad \begin{bmatrix} x_3^* \\ y_3^* \\ z_3^* \end{bmatrix} = \begin{bmatrix} \frac{-1-\sqrt{13}}{6} \\ \frac{11-\sqrt{13}}{18} \\ \frac{7+\sqrt{13}}{6} \end{bmatrix} \tag{4.31}$$

*Notice that these are the solutions only if the two assumptions **HP1** and **HP2** hold. Now we should release one of them per time and both and solve the resulting systems, so we have to consider other three cases. Begin with the case* $\mathbf{y} = \mathbf{0}, \mathbf{z} \neq \frac{\mathbf{1}}{\mathbf{2}}$*, plugging in eq. (4.26) one gets:*

$$\begin{cases} -2\lambda_1 x = 0 \\ xz = 0 \\ \lambda_2 = 0 \\ x^2 = 1 \\ y = 0 \end{cases} \Rightarrow \begin{cases} \lambda_1 = 0 \\ z = 0 \\ \lambda_2 = 0 \\ x = 1 \\ y = 0 \end{cases} \quad \text{OR} \quad \begin{cases} \lambda_1 = 0 \\ z = 0 \\ \lambda_2 = 0 \\ x = -1 \\ y = 0 \end{cases} \tag{4.32}$$

Notice that the second solution in this case is not acceptable because it violates the second constraint of the problem in (4.16).

Consider the case $\mathbf{y} \neq \mathbf{0}, \mathbf{z} = \frac{\mathbf{1}}{\mathbf{2}}$*, plugging in eq. (4.26) one gets:*

$$\begin{cases} z = \frac{1}{2} \\ \frac{1}{4} - 2\lambda_1 y = 0 \\ \lambda_2 = \frac{y}{2} \\ \frac{1}{4} + y^2 = 1 \\ x = 1 - z = \frac{1}{2} \end{cases} \Rightarrow \begin{cases} z = \frac{1}{2} \\ \lambda_1 = \frac{1}{8y} \\ \lambda_2 = \frac{y}{2} \\ y^2 = \frac{3}{4} \\ x = \frac{1}{2} \end{cases} \Rightarrow \begin{cases} y = \sqrt{\frac{3}{4}} \\ x = \frac{1}{2} \\ z = \frac{1}{2} \end{cases} \quad \text{OR} \quad \begin{cases} y = -\sqrt{\frac{3}{4}} \\ x = \frac{1}{2} \\ z = \frac{1}{2} \end{cases} \tag{4.33}$$

Last, consider the case $\mathbf{y} = \mathbf{0}, \mathbf{z} = \frac{\mathbf{1}}{\mathbf{2}}$*:*

$$\begin{cases} z = \frac{1}{2} \\ y = 0 \\ \frac{1}{4} - 2\lambda_1 y = 0 \\ \lambda_2 = \frac{y}{2} \\ x^2 = 1 \\ x = 1 - z = \frac{1}{2} \end{cases} \tag{4.34}$$

which is an evident contradiction.

Summarizing, all the solutions to the problem are given by:

$$\begin{bmatrix} x_1^* \\ y_1^* \\ z_1^* \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} x_2^* \\ y_2^* \\ z_2^* \end{bmatrix} = \begin{bmatrix} \frac{-1+\sqrt{13}}{6} \\ \frac{11+\sqrt{13}}{18} \\ \frac{7-\sqrt{13}}{6} \end{bmatrix} \qquad \begin{bmatrix} x_3^* \\ y_3^* \\ z_3^* \end{bmatrix} = \begin{bmatrix} \frac{-1-\sqrt{13}}{6} \\ \frac{11-\sqrt{13}}{18} \\ \frac{7+\sqrt{13}}{6} \end{bmatrix} \tag{4.35}$$

$$\begin{bmatrix} x_4^* \\ y_4^* \\ z_4^* \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ -\sqrt{\frac{3}{4}} \\ \frac{1}{2} \end{bmatrix} \qquad \begin{bmatrix} x_5^* \\ y_5^* \\ z_5^* \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \sqrt{\frac{3}{4}} \\ \frac{1}{2} \end{bmatrix} \tag{4.36}$$

which is just the union of the solutions obtained in all the four cases discussed above.

We may look at the values that the objective function assumes at each of these points to get a "negative test" for detecting the nature of these stationary points:

$$f(x_1^*,y_1^*,z_1^*)=0 \qquad f(x_2^*,y_2^*,z_2^*)=\frac{-29+17\sqrt{13}}{72} \quad f(x_3^*,y_3^*,z_3^*)=\frac{-29-17\sqrt{13}}{72} \tag{4.37}$$

$$f(x_4^*,y_4^*,z_4^*)=\frac{\sqrt{3}}{8} \qquad f(x_5^*,y_5^*,z_5^*)=-\frac{\sqrt{3}}{8} \tag{4.38}$$

□

4.3 Static Optimization: Inequality Constrained Optimization

Exercise 4.3.1 *Given the following optimization problem, draw the admissible region and compute the necessary KKT FOC:*

$$\begin{cases} \max\limits_{x,y} \quad x^2 y e^{-x-y} \\ s.t. \\ \qquad x \geq 1 \\ \qquad y \geq 1 \\ \qquad x+y \geq 4 \end{cases} \tag{4.39}$$

SOLUTION: *The admissible region is given by the intersection of three planes on* $\mathbb{R}^2$*:*

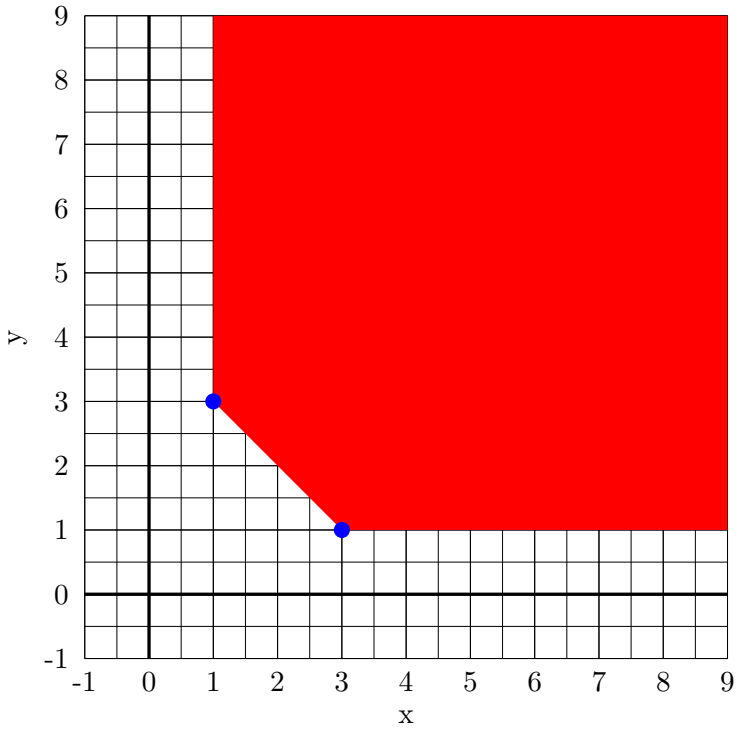


Figure 4.2: Admissible region of problem (4.39).

Notice that on the vertical line $y=1$ *only the second constraint is binding, on the horizontal line* $x=1$ *only the first one is binding, on the downward sloping segment only the third one is binding, while at the two blue points two constraints are simultaneously binding (either first and third or second and third), finally, in the middle of the red region there is no binding constraint.*

For the moment, ignore the constraint qualification conditions since their computation follows the same line of exercise 3a) in the previous exercise session, however it is necessary to check whether and where they are satisfied. Now compute the Lagrangian function:

$$\mathcal{L}(x,y,\lambda_1,\lambda_2,\lambda_3) = x^2ye^{-x-y} + \lambda_1[x-1] + \lambda_2[y-1] + \lambda_3[x+y-4] \tag{4.40}$$

where all the Lagrange multipliers are nonnegative. Notice that the sign which multiplies the Lagrange multipliers is "+" and their value is nonnegative: this is a consequence of:

- *the problem is a "Maximization"*
- *the constraints are expressed as "$\geq$"*
- *in the Lagrangian function we express the constraints as "$[g_i(x,y) - b_i]$"*

If one of the previous conditions is reversed, also the sign in front of the Lagrange multipliers (or their value) must be changed to its opposite. This is the result of the fact that we are simply multiplying by -1 one of the original equations/inequalities and then applying the original Kuhn-Tucker Theorem[7]*.*

Recall that in a inequality constrained optimization problem, the optimum (maximum or minimum) can be located either in the interior of the admissible region, where no constraint is binding (so it is equal to the optimum we would get in case of unconstrained optimization of the same objective function) or on the boundary, where one or two constraints are contemporary binding (basically what happens in the case of equality constraints, but here it not know a priori which of the inequality constraints, if any, is binding).

Then, the FOC yield the following system:

$$\begin{cases} 2xye^{-x-y} - x^2ye^{-x-y} + \lambda_1 + \lambda_3 = 0 \\ x^2e^{-x-y} - x^2ye^{-x-y} + \lambda_2 + \lambda_3 = 0 \\ x \geq 1 \\ y \geq 1 \\ x + y \geq 4 \\ \lambda_1 \geq 0 \\ \lambda_2 \geq 0 \\ \lambda_3 \geq 0 \\ \lambda_1 [x-1] = 0 \\ \lambda_2 [y-1] = 0 \\ \lambda_3 [x+y-4] = 0 \end{cases} \tag{4.41}$$

Since the complementary slackness conditions (the last three equations) are satisfied when either $\lambda = 0$ or the constraint is binding or both, we need to consider all the possible combination of the λ, which are $2^m = 2^3 = 8$ in this case.

CASE 1: $\boldsymbol{\lambda_1 = 0, \lambda_2 = 0, \lambda_3 = 0}$ *$\Rightarrow$ all the constraints may be binding or not, we have no information at all. Just plug in the values of the multipliers to get:*

$$\begin{cases} 2xye^{-x-y} - x^2ye^{-x-y} = 0 \\ x^2e^{-x-y} - x^2ye^{-x-y} = 0 \lambda_1 = \lambda_2 = \lambda_3 = 0 \\ \dots \end{cases} \quad \Rightarrow \quad \begin{cases} xy(2-x) = 0 \\ x^2(1-y) = 0 \\ \lambda_1 = \lambda_2 = \lambda_3 = 0 \\ \dots \end{cases} \tag{4.42}$$

[7] Recall that this Theorem is formulated in terms of **maximization** with constraints written with "$\leq$" and Lagrangian composed with "$[b_i - g_i(x,y)]$" and yield "$-\lambda_i$" in the Lagrangian, with $\lambda_i \geq 0$.

which is impossible, since the second is satisfied only for $x = 2$,$y = 1 \notin \mathcal{D}$ (third constraint not satisfied).

CASE 2: $\boldsymbol{\lambda_1 = 0, \lambda_2 > 0, \lambda_3 = 0} \Rightarrow$ *the second constraint is surely binding, but no information about the others:*

$$\begin{cases} 2xe^{-x-1} - x^2e^{-x-1} = 0 \\ x^2e^{-x-1} - x^2e^{-x-1} + \lambda_2 = 0 \\ y = 1 \\ \dots \end{cases} \Rightarrow \begin{cases} x(2-x) = 0 \\ x^2e^{-x-1} - x^2e^{-x-1} + \lambda_2 = 0 \\ y = 1 \\ \dots \end{cases} \tag{4.43}$$

again, we obtain $x = 2$,$y = 1 \notin \mathcal{D}$, hence no solution.

CASE 3: $\boldsymbol{\lambda_1 = 0, \lambda_2 > 0, \lambda_3 > 0} \Rightarrow$ *the second and third constraints are surely binding, but no information about the first:*

$$\begin{cases} 2xe^{-x-1} - x^2e^{-x-1} + \lambda_3 = 0 \\ x^2e^{-x-1} - x^2e^{-x-1} + \lambda_2 + \lambda_3 = 0 \\ y = 1 \\ x + y = 4 \\ \dots \end{cases} \Rightarrow \begin{cases} \lambda_3 = 3e^{-4} > 0 \\ \lambda_2 = -\lambda_3 < 0 \\ y = 1 \\ x = 3 \\ \dots \end{cases} \tag{4.44}$$

now, the conclusions contradict the assumption $\lambda_2 > 0$, hence no solution is obtained.

CASE 4: $\boldsymbol{\lambda_1 = 0, \lambda_2 = 0, \lambda_3 > 0} \Rightarrow$ *the third constraint is surely binding, but no information about the others:*

$$\begin{cases} 2xe^{-x-1} - x^2e^{-x-1} + \lambda_3 = 0 \\ x^2e^{-x-1} - x^2e^{-x-1} + \lambda_3 = 0 \\ x + y = 4 \\ \dots \end{cases} \Rightarrow \begin{cases} e^4(2x - x^2)(4-x) + \lambda_3 = 0 \\ e^4x^2(1 - 4 + x) + \lambda_3 = 0 \\ y = 4 - x \\ \dots \end{cases} \Rightarrow \begin{cases} \lambda_3 = -e^4x(8 - 6x + x^2) \\ e^4x^2(x-3) - e^4x(8 - 6x + x^2) = 0 \\ y = 4 - x \\ \dots \end{cases} \tag{4.45}$$

$$\begin{cases} \lambda_3 = -e^4x(8 - 6x + x^2) \\ -3x^2 + x^3 - 8x + 6x^2 - x^3 = 0 \\ y = 4 - x \\ \dots \end{cases} \Rightarrow \begin{cases} \lambda_3 = -e^4\frac{8}{3}\left(-\frac{8}{9}\right) > 0 \\ x = \frac{8}{3} > 1 \\ y = \frac{4}{3} > 1 \\ \dots \end{cases} \tag{4.46}$$

so in this case we find one admissible solution.

CASE 5: $\boldsymbol{\lambda_1 > 0, \lambda_2 = 0, \lambda_3 = 0} \Rightarrow$ *the first constraint is surely binding, but no information about the others:*

$$\begin{cases} 2ye^{-1-y} - ye^{-1-y} + \lambda_1 = 0 \\ e^{-1-y} - ye^{-1-y} = 0 \\ x = 1 \\ \dots \end{cases} \Rightarrow \begin{cases} e^{-2} + \lambda_1 = 0 \\ y = 1 \\ x = 1 \\ \dots \end{cases} \tag{4.47}$$

which leads to a contradiction of the assumption and a violation of the third constraint.

CASE 6: $\boldsymbol{\lambda_1 > 0, \lambda_2 > 0, \lambda_3 = 0} \Rightarrow$ *the first and second constraints are surely binding, but no information about the third:*

$$\begin{cases} 2e^{-2} - e^{-2} + \lambda_1 = 0 \\ e^{-2} - e^{-2} + \lambda_2 = 0 \\ x = 1 \\ y = 1 \\ \dots \end{cases} \tag{4.48}$$

which contradicts the assumption that $\lambda_2 > 0$.

CASE 7: $\lambda_1 > 0, \lambda_2 = 0, \lambda_3 > 0 \Rightarrow$ *the first and third constraints are surely binding, but no information about the second:*

$$\begin{cases} 2ye^{-1-y} - ye^{-1-y} + \lambda_1 + \lambda_3 = 0 \\ e^{-1-y} - ye^{-1-y} + \lambda_3 = 0 \\ x = 1 \\ x + y = 4 \\ \dots \end{cases} \Rightarrow \begin{cases} 3e^{-4} + \lambda_1 + 2e^{-4} = 0 \\ \lambda_3 = 2e^{-4} > 0 \\ x = 1 \\ y = 3 \\ \dots \end{cases} \tag{4.49}$$

which contradicts the assumption that $\lambda_1 > 0$.

CASE 8: $\lambda_1 > 0, \lambda_2 > 0, \lambda_3 > 0 \Rightarrow$ *all the constraints are surely binding:*

$$\begin{cases} 2ye^{-1-y} - ye^{-1-y} + \lambda_1 + \lambda_3 = 0 \\ e^{-1-y} - ye^{-1-y} + \lambda_3 = 0 \\ x = 1 \\ y = 1 \\ x + y = 4 \\ \dots \end{cases} \tag{4.50}$$

which is clearly impossible, as can be seen from Figure (4.2), in which there are no points of intersection between all the three constraints.

In the end, the only stationary point we obtained is:

$$\begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} \frac{8}{3} \\ \frac{4}{3} \end{bmatrix}$$

In order to assess whether this point satisfies the necessary KKT conditions for the problem (4.39) the constraint qualification conditions are met at this point. The Jacobian is given by:

$$\mathbf{J}(x,y) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \tag{4.51}$$

which has rank equal to 2 $\forall (x,y) \in \mathbb{R}^2$*, hence we can conclude that the point* (x^*,y^*) *satisfied the first order KKT necessary conditions for a maximum.*

Graphically, this point is located on the downward sloping segment in Figure (4.2), where only the third constraint is binding.

□

4.4 Differential Equations

Exercise 4.4.1 *Given the following differential equation, specify its type then find a general solution:*

$$t^5 \dot{x} - x^{-5} = 0 \tag{4.52}$$

SOLUTION: *This equation is nonlinear in* x*, but after a little bit of manipulation we obtain:*

$$t^5 \dot{x} = x^{-5} \Rightarrow \dot{x} = x^{-5} t^{-5} \tag{4.53}$$

which can be recognized as a separable differential equation. Starting from here, it is necessary to split the derivative on the left hand side and compute the two integrals that emerge, as follows:

$$\begin{aligned}\dot{x} = \frac{\mathrm{d}x}{\mathrm{d}t} &= x^{-5}t^{-5} \\ x^5\,\mathrm{d}x &= t^{-5}\,\mathrm{d}t \\ \int x^5\,\mathrm{d}x &= \int t^{-5}\,\mathrm{d}t \\ \frac{x^6}{6} &= -\frac{t^{-4}}{4} + c \quad c \in \mathbb{R}.\end{aligned}$$

So the general solution is:

$$x^*(t) = \sqrt[6]{-\frac{3}{2}t^{-4} + 6c}, \quad c \in \mathbb{R}. \tag{4.54}$$

□

Exercise 4.4.2 *Given the following differential equation, specify its type then find a general solution:*

$$e^x + (te^x + 2x)\dot{x} = 0 \tag{4.55}$$

SOLUTION: *This equation is nonlinear in x, and it is an exact differential equation. In order to check whether we can apply the method for exact equations, we need to put the equation in the suitable form:*

$$P(t,x) + Q(t,x)\dot{x} = 0 \tag{4.56}$$

and then to check that:

$$\frac{\partial^2 f}{\partial t \partial x} = \frac{\mathrm{d}P(t,x)}{\mathrm{d}x} = \frac{\mathrm{d}Q(t,x)}{\mathrm{d}t} = \frac{\partial^2 f}{\partial x \partial t}. \tag{4.57}$$

In this case, we need not to rearrange terms, hence we can directly compute the derivatives:

$$\frac{\mathrm{d}P(t,x)}{\mathrm{d}x} = e^x = \frac{\mathrm{d}Q(t,x)}{\mathrm{d}t} \tag{4.58}$$

therefore we can apply the method for solving exact equations, which consists in taking the integral of the equation written in the form as eq. (4.56) with respect to t. In this case, focusing on the first part $P(t,x) = e^x$ it yields:

$$\int e^x\,\mathrm{d}t = te^x + \phi(x) = f(t,x) \tag{4.59}$$

where the $\phi(x)$ emerges as a constant of integration. This constant is indeed a function of x, hence one may think that it is not a "proper constant". Indeed, since we have integrated with respect to t, we have kept all other variables as constants for the sake of the computation of the integral[8]. Hence, in order to be the most general possible, we put a constant of integration which is in reality function of all the other variables, different from that according to which we have integrated.

In order to find out what is the true form of $\phi(x)$ we recall the necessary and sufficient conditions in eq. (4.57): since we know that the derivative of the original function with respect to x corresponds other part of the equation (that is, $\frac{\mathrm{d}f}{\mathrm{d}x} = Q(t,x)$), we can compute it again taking into account this time also the $\phi(x)$, as follows:

$$\frac{\partial f(t,x)}{\partial x} = te^x + \phi'(x) = te^x + 2x = Q(t,x) \tag{4.60}$$

[8]Notice that this process applies exactly the same reasoning as that for computing partial derivatives of a vectorial function, where at each step just one variable is considered, while the others are treated "as if" they were constants.

for the second equality to hold, it is necessary that $\phi'(x) = 2x$, hence $\phi(x) = x^2 + d$. We can now put together all the parts we have computed, to get the general solution:

$$f^*(t,x) = te^x + x^2 + \tilde{c} \qquad \tilde{c} = d - c, \quad \tilde{c} \in \mathbb{R}. \tag{4.61}$$

The final constant $\tilde{c}$ has been computed by the difference of the constant emerged in the previous steps: d emerged as we integrated the $\phi'(x)$, while c from the fact that the solution of (4.55) is of the form $f(t,x) = c$.

□

Exercise 4.4.3 *Given the following differential equation, specify its type then find a general solution:*

$$\dot{x} = \left(\frac{2t-2}{t^2-2t+1}\right) x + t - 1 \tag{4.62}$$

SOLUTION: *This first order differential equation is clearly linear in x, in addition it is not separable and not exact. Since the coefficient of the x is not constant (it is a function of the variable t), we cannot apply the solution method for linear equations with constant coefficients, but we must use the general formula for linear differential equations.*

The first step consists in computing the (indefinite) integral of the coefficient of the x[9]*:*

$$A(t) = \int a(t)\,\mathrm{d}t = \int \frac{2t-2}{t^2-2t+1}\,\mathrm{d}t = \ln(t^2-2t+1). \tag{4.63}$$

Now, we apply the general formula:

$$\begin{aligned}
x^*(t) &= e^{A(t)}\left(c + \int e^{-A(t)} b(t)\,\mathrm{d}t\right) \\
&= e^{\ln(t^2-2t+1)}\left(c + \int e^{-\ln(t^2-2t+1)}(t-1)\,\mathrm{d}t\right) \\
&= (t^2-2t+1)\left(c + \int \frac{1}{t^2-2t+1}(t-1)\,\mathrm{d}t\right) \\
&= (t-1)^2\left(c + \int \frac{t-1}{(t-1)^2}\,\mathrm{d}t\right) \\
&= (t-1)^2\left(c + \ln(t-1)\right) \\
&= c(t-1)^2 + (t^2-2t+1)\ln(t-1) \quad c \in \mathbb{R}
\end{aligned}$$

□

Exercise 4.4.4 *Give a general solution of the following linear differential equation:*

$$4\ddot{x} + 4\dot{x} + 2x = 3e^{2t} \tag{4.64}$$

SOLUTION: *Start by looking for the solutions of the characteristic polynomial associated to the equation:*

$$4\lambda^2 + 4\lambda + 2 = 0, \tag{4.65}$$

[9] Notice that, if the x was on the left hand side of the equation, we would have to compute the integral of the opposite of its coefficient.

whose $\Delta = 16 - 32 = -16 < 0$. *As a consequence, the solutions are complex and conjugate:*

$$\lambda_1 = \frac{-4 + \iota 4}{8} = \frac{-1 + \iota}{2} \quad \lambda_2 = \frac{-4 - \iota 4}{8} = \frac{-1 - \iota}{2}. \tag{4.66}$$

As a consequence, the general solution of the homogeneous equation is[10]*:*

$$x_h^*(t) = e^{-\frac{1}{2}t}\left(c_1 \cos\left(\frac{1}{2}\right) + c_2 \sin\left(\frac{1}{2}t\right)\right) \qquad (c_1,c_2) \in \mathbb{R}^2. \tag{4.67}$$

Next, we need to look a particular solution of the whole equation (4.64)*, according to the family of* $b(t) = 3e^{2t}$*: we guess the solution* $x = ke^{2t}$*, since* 2 *is not a solution of the characteristic polynomial. Then, consider the first and second derivative:* $\dot{x} = 2kte^{2t}$ *and* $\ddot{x} = 4kte^{2t}$ *and plug all into* (4.64)*, then solve. One gets:*

$$\begin{aligned} &4(4ke^{2t}) + 4(2ke^{2t}) + 2ke^{2t} = 3e^{2t} \\ &16ke^{2t}) + 8ke^{2t}) + 2ke^{2t} = 3e^{2t} \\ &26k = 3 \qquad k = \frac{3}{26}. \end{aligned}$$

Hence the particular solution is:

$$x_p^*(t) = \frac{3}{26}e^{2t} \tag{4.68}$$

Putting together the two solutions (4.67) *and* (4.68)*, we get the general solution of the original equation* (4.64)*:*

$$x^*(t) = e^{-\frac{1}{2}t}\left(c_1 \cos\left(\frac{1}{2}\right) + c_2 \sin\left(\frac{1}{2}\right)\right) + \frac{3}{26}e^{2t} \qquad (c_1,c_2) \in \mathbb{R}^2. \tag{4.69}$$

□

Exercise 4.4.5 *Given the following autonomous differential equation, draw its phase diagram, find its equilibrium points and study their stability properties:*

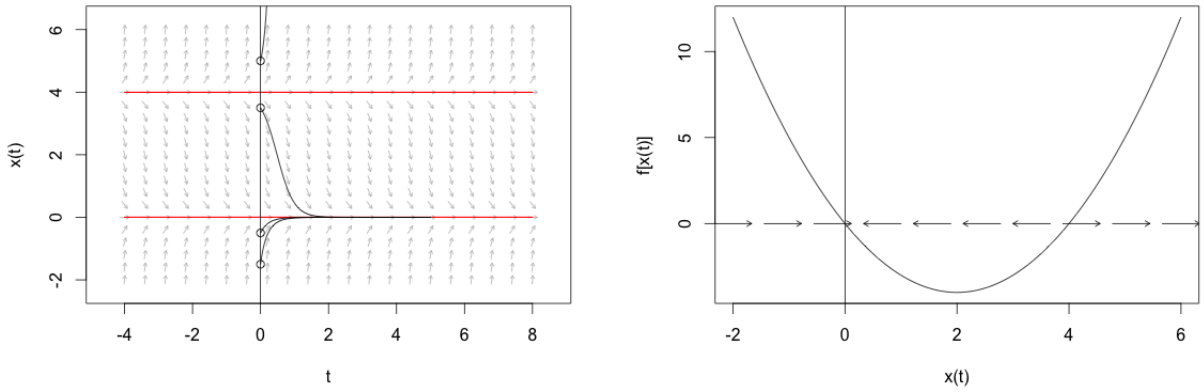
$$\dot{x} = x^2 - 4x \tag{4.70}$$

SOLUTION: *the differential equation* (4.70) *does not depend explicitly on* t*, hence it is a first order autonomous differential equation. We can study it from a qualitative point o view by means of phase diagram. Start by drawing the graph of the function* $f(x) = x^2 - 4x$*, as results in Figure* (4.3(b))*. The equilibrium points for this type of equation are all those functions* $x(t)$ *such that* $\dot{x} = 0$*, therefore, graphically, all the points in which the function* $f(x)$ *intersects the* $x-$*axis. In this case we have two such points:* (0,0) *and* (2,0)*.*

[10] Recall that the solution of the form:

$$e^{\alpha}(c_1 \cos(\beta) + c_2 \sin(\beta))$$

where $\lambda = \alpha \pm \beta$.



(a) *Flow field* (b) *Phase Diagram*

Figure 4.3: Flow field and Phase diagram of the equation (4.70).

We can study their stability properties in two different but equivalent ways: from the flow field (Fig. (4.3(a))) or from the phase diagram (Fig. (4.3(b))). Consider the first: the flow field reports the solution $x(t)$ as a function of t: the two red lines are the constant solutions of the original problem (4.70), in fact, since they are the constant solutions such that $\dot{x} = 0$, they are represented in the flow field as horizontal lines (that is, the derivative of $x(t)$ with respect to t is null). By looking at the grey arrows, which depict the directions of movement for any given point in the plane, it is easy to see that there are no initial values of $x(0)$ such that there exists a trajectory in the plane leading to $x(t) = 2$ (corresponding to the equilibrium $(2,0)$), therefore this equilibrium is unstable. By contrast, there exists an interval of values of $x(0)$ sufficiently close to 0 for which there exists a trajectory leading to $x(t) = 0$ (corresponding to the equilibrium $(0,0)$), hence this point is locally asymptotically stable. Notice that it is not globally asymptotically stable, since for all initial values $x(0) > 2$ there does not exist trajectories leading to it.

Now, consider the phase diagram in Fig. (4.3(b)): the arrows state that, for values of $f(x) > 0$ it holds $f(x) = \dot{x} > 0$, hence x is increasing, therefore the arrows points to right, and the opposite holds for all x such that $f(x) < 0$. As a consequence we can see that in a neighbourhood of the point $(2,0)$ the arrows are pointing away from the equilibrium, which is unstable. On the other hand, in a sufficiently small neighbourhood of the origin, all the arrows are pointing towards $(0,0)$, which is locally asymptotically stable.

□

Exercise 4.4.6 *Given the following system of linear differential equations, draw the associated phase diagram, find all the equilibrium points and study their stability properties:*

$$\begin{cases} \dot{x} = -3x + y \\ \dot{y} = -2x - y \end{cases} \tag{4.71}$$

SOLUTION: *Start by drawing the nullcline of the first equation by setting $\dot{x} = -x + y = 0$: this is the straight line $y = 3x$ plot in red in Figure (4.4). Next, fix a value for x, for example $x = 1$: the corresponding value of y is 3 too. As a consequence, if we take any value of $y > 3$ for the same value of x the previous relation, $y = 3x$, is no more satisfied with equality, instead it would be $y > 3x$ and, as a consequence from the first equation, $\dot{x} > 0$. Instead, if after fixing $x = 1$ we take a $y < 3$, we would get $y < 3x$ hence $\dot{x} < 0$. From these two results we are able to draw the grey arrows above*

and below the red nullcline: above the line $\dot{x} > 0$, hence x is increasing, therefore we draw an arrow pointing towards right; by contrast below $\dot{x} < 0$ so we draw arrows pointing towards left.

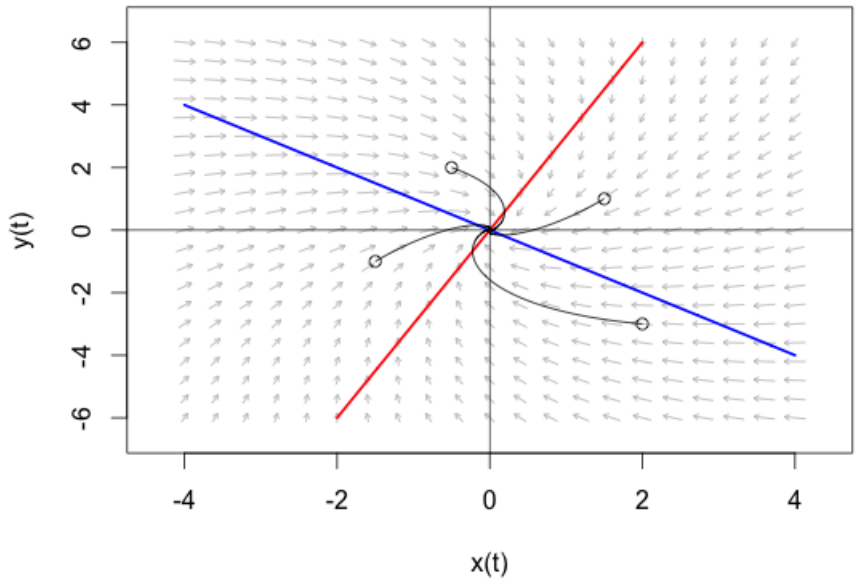


Figure 4.4: Phase diagram of the system (4.71).

We repeat the same procedure and logic steps for the second equation, whose nullcline has equation $y = -2x$ corresponding to the blue line in Fig. (4.71). Fix $x = 0$: the "correct y" equal 0. Below it ($y < 0$), hence $\dot{y} > 0$, while above it ($y > 0$) leading to $\dot{y} < 0$.

Some trajectories can be drawn (black curves in Fig. (4.71)) to have a clearer idea of the stability of the unique equilibrium point, which is the intersection of the two nullclines: (0,0). This point is clearly locally asymptotically stable, but in this case it is also globally asymptotically stable, since for all initial conditions (i.e. "starting points") the trajectory leads towards it. This can be seen directly from the direction of the grey arrows, without any need of the black trajectories.

□

Exercise 4.4.7 *Given the following system of linear differential equations, draw the associated phase diagram, find all the equilibrium points and study their stability properties:*

$$\begin{cases} \dot{x} = ax - bx^2 - y \\ \dot{y} = w(a - 2bx)y \end{cases} \tag{4.72}$$

where $(a,b,w) = (6,1,3)$ and assuming $x \geq 0$, $y \geq 0$.

SOLUTION: *The nullcline corresponding to the first equation is a concave parabola (red curve in Fig. (4.5)). Above it we have $y > 6x - x^2$, hence $\dot{x} < 0$ therefore the arrows point to left; the opposite result holds for points below the curve. On the other hand, the second equation has two nullclines, respectively $y = 0$ and $x = 3$ (blue lines in Fig. (4.5)). Notice that we are interested only in the first quadrant, where $x \geq 0$ and $y \geq 0$, hence ignore the part of the graph below the horizontal blue nullcline. Consider the region above this nullcline, there are two possibilities: above the horizontal and to the left of the vertical: $y > 0$ and $x < 3$, therefore from the second equation it emerges that $\dot{y} > 0$ hence we*

draw arrows pointing upwards; instead to the right of the vertical nullcline it holds $y > 0$ and $x > 3$, therefore from the second equation it emerges that $\dot{y} < 0$ hence we draw arrows pointing downwards.

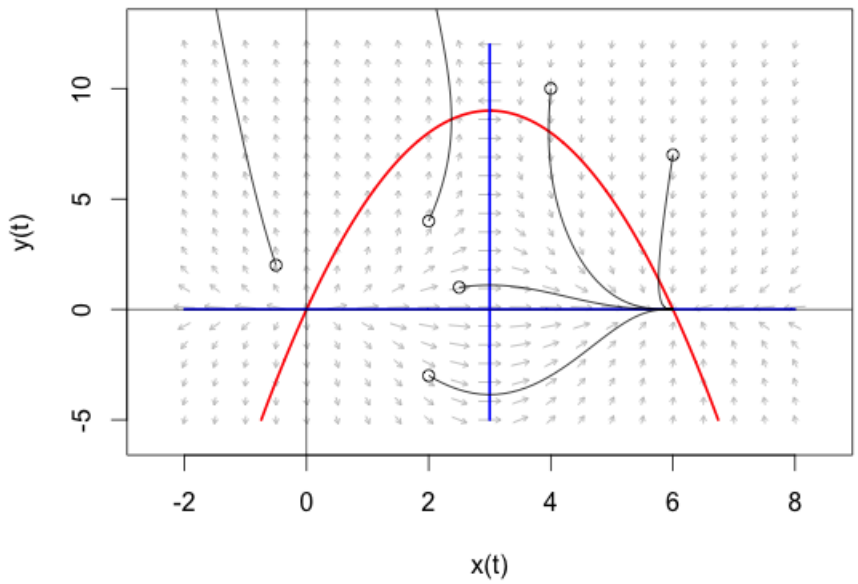


Figure 4.5: Phase diagram of the system (4.72).

The final graph is drawn in Figure (4.5), together with some trajectories. There are three equilibria in this case: (0,0), (3,9) *and* (6,0), *which are the intersection points with the blue and red curves, that is, the nullclines of the two equations of the system. In fact, notice that the point* (3,0) *where the two (blue) nullclines intersect is not an equilibrium since these two nullclines are associated to the same equation.*

In order to check the stability of these points, notice that the trajectories drawn in black and the grey arrows show clearly that for any starting point in a neighbourhood of the first two points the dynamics drives far away from the point, hence we conclude that they are unstable. By contrast, the last point is locally asymptotically stable, since in a neighbourhood (but not for all initial points, that is why it is not globally asymptotically stable) all trajectories converge to it.

□

4.5 Difference Equations

Exercise 4.5.1 *Give a general solution of the following linear difference equation:*

$$x_{t+1} + 3x_t = 4^t \tag{4.73}$$

SOLUTION: *Before applying the method for solving first order linear difference equations with constant coefficients, recall that this formula requires that the term with x_t is on the right hand side of the equation. If one would like to apply the same method directly on eq. (4.73) it is necessary to consider the coefficient $a = 3$ with opposite sign (in this case, -3 should be considered).*

In any case, one gets as solution of the associated homogeneous equation:

$$x_t^h = c(-3)^t \quad c \in \mathbb{R}. \tag{4.74}$$

Then one must look for a particular solution of the same form of $b(t) = 4^t$: since 4 is not a solution for the homogeneous case, we guess $x_t = k4^t$. Taking the value one period ahead yields: $x_{t+1} = k4^{t+1} = 4k4^t$. Then, plugging all into eq. (4.73) gives:

$$4k4^t + 3k4^t = 4^t \tag{4.75}$$

from which, by the principle of identity of polynomials, one gets: $k = \frac{1}{7}$. Therefore, the particular solution is:

$$x_t^p = \frac{1}{7}4^t. \tag{4.76}$$

Putting all together one gets the general solution of eq. (4.73):

$$x_t^* = c(-3)^t + \frac{1}{7}4^t \quad c \in \mathbb{R}. \tag{4.77}$$

□

Exercise 4.5.2 *Give a general solution of the following linear difference equation:*

$$x_{t+2} - 3x_{t+1} + 2x_t = t^2 + 2^t \tag{4.78}$$

SOLUTION: *The starting point, as for differential equations, is to compute the roots of the characteristic polynomial associated to the equation:*

$$\lambda^2 - 3\lambda + 2 = 0 \tag{4.79}$$

from which $\Delta = 9 - 8 = 1 > 0$, hence the real and distinct solutions are: $\lambda_1 = 1$ and $\lambda_2 = 2$. As a consequence, the solution of the homogeneous equation is:

$$x_t^h = c_1(1)^t + c_2(2)^t \qquad (c_1,c_2) \in \mathbb{R}^2. \tag{4.80}$$

We are left with the computation of a particular solution of eq. (4.78). In order to do this, consider separately each "type" of function composing $b(t)$, that is consider separately the exponential and the polynomial. Concerning the exponential, since 2 is a single root of the characteristic polynomial, we should not consider $k2^t$ as a guess, instead we need to multiply it by t: guess $x_t = kt2^t$. Compute the values one and two periods ahead: $x_{t+1} = k(t+1)2^{t+1} = 2kt2^t + 2k2^t$ and $x_{t+2} = k(t+2)2^{t+2} = 4kt2^t + 8k2^t$. Plug them into the original equation (4.78), ignoring again all the other parts of $b(t)$ except the one we are currently studying, and solve for k:

$$4kt2^t + 8k2^t - 3(2kt2^t + 2k2^t) + 2kt2^t = 2^t$$

$$4kt - 6kt + 2kt8k - 6k = 1 \quad \Rightarrow \quad k = \frac{1}{2}$$

hence this part has solution: $x_t^{p1} = \frac{1}{2}t2^t$. Now turn to the other part of $b(t)$, which is a second order polynomial t^2. Notice that, even though this polynomial is not complete, we must guess as a solution the complete polynomial of (at least) the same degree. However, since 1 is a single root of the characteristic polynomial, we cannot guess $at^2 + bt + c$, but, as in previous case, we need to multiply it by t: guess $x_t = t(at^2 + bt + c)$. One and two steps ahead values are: $x_{t+1} = (t+1)(a(t+1)^2 + b(t+1) + c) = (t+1)(at^2 + 2at + a + bt + b + c) = at^3 + 2at^2 + at + bt^2 + bt + ct + at^2 + 2at + a + bt + b + c$ and $x_{t+2} = (t+2)(a(t+2)^2 + b(t+2) + c) = (t+2)(at^2 + 4at + 4a + bt + 2b + c) = at^3 + 4at^2 + 4at + bt^2 + 2bt + ct + 2at^2 + 8at + 8a + 2bt + 4b + 2c$. Plug them into eq. (4.78), ignoring all other components of $b(t)$:

$$\begin{aligned}
&at^3 + 4at^2 + 4at + bt^2 + 2bt + ct + 2at^2 + 8at + 8a + 2bt + 4b + 2c - 3at^3 - 6at^2 - 3at \\
&- 3bt^2 - 3bt - 3ct - 3at^2 - 6at - 3a - 3bt - 3b - 3c + 2t(at^2 + bt + c) = t^2 \\
&4at - 2bt + ct - at^2 + 8at + 8a + 4b + 2c - 3at - 3ct - 6at - 3a - 3b - 3c + 2ct = t^2 \\
&- at^2 + (3a - 2b)t + (5a + b - c) = t^2.
\end{aligned}$$

According to the principle of equivalence between polynomials, we must equalize the coefficients of the variable (of the same degree) of the two sides, leading to:

$$\begin{cases} -a = 1 \\ 3a - 2b = 0 \\ 5a + b - c = 0 \end{cases} \quad \Rightarrow \quad \begin{cases} a = -1 \\ b = -\frac{3}{2} \\ c = -\frac{13}{2} \end{cases} \tag{4.81}$$

leading to the part of solution: $x_t^{p2} = -t^3 - \frac{3}{2}t^2 - \frac{13}{2}t$. *Putting all the parts together delivers the general solution of eq. (4.78):*

$$x_t^* = c_1(1)^t + c_2(2)^t - t^3 - \frac{3}{2}t^2 - \frac{13}{2}t + \frac{1}{2}t2^t \qquad (c_1,c_2) \in \mathbb{R}^2. \tag{4.82}$$

□

Exercise 4.5.3 *Draw the phase diagram of the following first order autonomous difference equation, then find out all its equilibrium points and study their stability:*

$$x_{t+1} = -(x_t)^2 + 4 \tag{4.83}$$

SOLUTION: *This equation is a first order nonlinear autonomous difference equation, which can be easily represented on the* $x_{t+1} - x_t$ *plane. The function* $f(x_t) = -(x_t)^2 + 4$ *is a concave parabola with maximum at* (0,4).

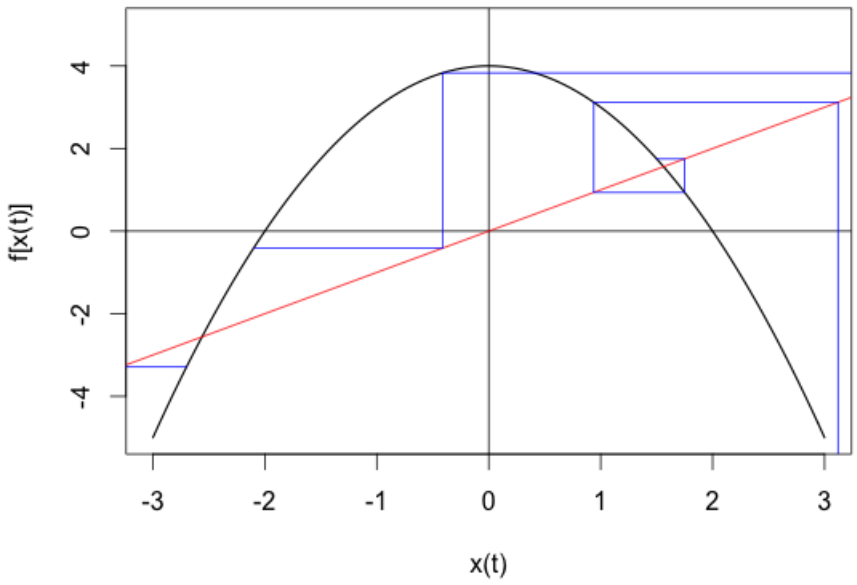


Figure 4.6: Phase diagram of the equation (4.83).

In Figure (4.6) is reported the phase diagram of this difference equation: the black curve is $f(x_t)$, *while the red line is the bisector of the first quadrant. The equilibrium points are all the* x_t *such that* $x_{t+1} = f(x_t) = x_t$ *hence are those points located at the intersection between the curve and the bisector. In this case there exist two equilibrium points: one is located in the third quadrant and the other is in*

the first quadrant. We can study their stability analytically by computing the values of the derivative of the function $f(x_t)$ at each equilibrium point and then compare it with the slope of the bisector (which is always 1): if the latter is smaller, then the point is locally asymptotically stable, otherwise if it is greater then the equilibrium is unstable[11]*. Another way consists in drawing the cobweb starting from a chosen point x_0 and checking whether the trajectory converges or not to the equilibrium. This is what has been done in Figure (4.6) for three points: $x_0 = -2.8$, $x_0 = -2.1$ and $x_0 = 1.5$. In all these cases it is possible to see that the trajectories diverge from both equilibria, which are in fact unstable.*

□

4.6 Calculus of Variations

Exercise 4.6.1 *Find the extremals of the following calculus of variations problem:*

$$\begin{cases} \int_0^1 \dot{x}^2 + 10tx \, \mathrm{d}t \\ s.t. \\ \quad x(0) = 1 \\ \quad x(1) = 2 \end{cases} \tag{4.84}$$

SOLUTION: *The necessary condition to be satisfied both for having a maximum and a minimum is that the solution satisfies the Euler equation. Hence proceed by defining this differential equation, starting from its building blocks:*

$$\begin{aligned} F_2'(t,x,\dot{x}) &= 10t \\ F_3'(t,x,\dot{x}) &= 2\dot{x} \\ \frac{\mathrm{d}}{\mathrm{d}t} F_3'(t,x,\dot{x}) &= 2\ddot{x}. \end{aligned}$$

Now, form the Euler equation:

$$10t - 2\ddot{x} = 0 \quad \Rightarrow \quad \ddot{x} = 5t \tag{4.85}$$

and solve it simply by integrating it twice with respect to t:

$$x^*(t) = \int \left(\int 5t \, \mathrm{d}t \right) \mathrm{d}t = \int \frac{5}{2}t^2 + c_1 \, \mathrm{d}t \tag{4.86}$$

$$= \frac{5}{6}t^3 + c_1 t + c_2 \quad (c_1,c_2) \in \mathbb{R}^2. \tag{4.87}$$

In order to recover the value of the two constants, plug in the boundary conditions and solve the system:

$$\begin{cases} x^*(0) = c_2 = 1 \\ x^*(1) = \frac{5}{6} + c_1 + c_2 = 2 \end{cases} \quad \Rightarrow \quad \begin{cases} c_2 = 1 \\ c_1 = \frac{1}{6} \end{cases} \tag{4.88}$$

Concluding, in this case the unique extremal for the problem in (4.84) is:

$$x^*(t) = \frac{5}{6}t^3 + \frac{t}{6} + 1 \tag{4.89}$$

□

[11] In case the derivative at the equilibrium point is exactly 1, a local study of the function in the neighbourhood of the equilibrium is necessary to establish its stability.

Exercise 4.6.2 *Solve the following isoperimetric problem:*

$$\begin{cases} \max \displaystyle\int_0^T x \, \mathrm{d}t \\ s.t. \\ \quad \displaystyle\int_0^T \left(1+\dot{x}^2\right)^{\frac{1}{2}} \, \mathrm{d}t = k \quad k \in \mathbb{R} \\ \quad x(0) = 0 \\ \quad x(T) = 0 \quad T \in \mathbb{R} \end{cases} \tag{4.90}$$

SOLUTION: *The first step in order to solve an isoperimetric problem is to form the augmented integrand:*

$$L(t,x,\dot{x}) = F(t,x,\dot{x}) - \lambda G(t,x,\dot{x}) = x - \lambda \left(1+\dot{x}^2\right)^{\frac{1}{2}}. \tag{4.91}$$

Then, the problem is exactly the same as an ordinary calculus of variations one, where the functional to be maximized is $L(t,x,\dot{x})$. As a consequence, it is necessary to compute the solution of the Euler equation:

$$0 = 1 - \frac{\mathrm{d}}{\mathrm{d}t}\left[-\lambda \frac{1}{2}\left(1+\dot{x}^2\right)^{-\frac{1}{2}} 2\dot{x}\right] \tag{4.92}$$

$$1 = -\frac{\mathrm{d}}{\mathrm{d}t}\left[\lambda \dot{x}\left(1+\dot{x}^2\right)^{-\frac{1}{2}}\right]. \tag{4.93}$$

Now we can proceed and compute the derivative, but notice that, since the goal is to solve the differential equation for $x(t)$ and equation (4.93) contains its derivative $\dot{x}$, by proceeding in this way we will end up with a second order differential equation to solve. Alternatively it is possible to integrate equation (4.93) with respect to t, obtaining a first order differential equation: in fact, the integration with respect to t and the derivative $\frac{\mathrm{d}}{\mathrm{d}t}$ simplify (nonetheless a constant as an outcome of the integration operation arises) since both of them are taken with respect to the same variable. We proceed in this way and get:

$$t = -\lambda \dot{x}\left(1+\dot{x}^2\right)^{-\frac{1}{2}} + c_1 \tag{4.94}$$

$$-\frac{t-c_1}{\lambda} = \dot{x}\left(1+\dot{x}^2\right)^{-\frac{1}{2}} \tag{4.95}$$

$$\dot{x} = -\frac{t-c_1}{\lambda}\left(1+\dot{x}^2\right)^{\frac{1}{2}} \tag{4.96}$$

$$\dot{x}^2 = \frac{(t-c_1)^2}{\lambda^2}\left(1+\dot{x}^2\right) \tag{4.97}$$

$$\dot{x}^2\left(1 - \frac{(t-c_1)^2}{\lambda^2}\right) = \frac{(t-c_1)^2}{\lambda^2} \tag{4.98}$$

$$\dot{x} = \frac{t-c_1}{\lambda}\left(1 - \frac{(t-c_1)^2}{\lambda^2}\right)^{-\frac{1}{2}} = \frac{t-c_1}{\left(\lambda^2 - (t-c_1)^2\right)^{\frac{1}{2}}}. \tag{4.99}$$

It is necessary to integrate with respect to t in order to solve this first order differential equation, but prior to do this notice that the fraction on the right hand side is structured as follows: the numerator is "similar" to the first order derivative (with respect to t) of the basis of the power at the denominator. That is:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\lambda^2 - (t-c_1)^2\right) = -2(t-c_1).$$

It is possible to exploit this fact and use a change of variable for solving the integral. IN particular, use:

$$z = \lambda^2 - (t - c_1)^2 \tag{4.100}$$

$$\mathrm{d}z = -2(t - c_1)\mathrm{d}t \tag{4.101}$$

$$-\frac{1}{2(t - c_1)}\mathrm{d}z = \mathrm{d}t \tag{4.102}$$

and substitute equation (4.101) and (4.102) into equation (4.99) to get:

$$x^*(t) = \int \dot{x}\,\mathrm{d}t = \int -\frac{t - c_1}{z^{\frac{1}{2}}}\frac{1}{2(t - c_1)}\,\mathrm{d}z = \int -\frac{1}{2z^{\frac{1}{2}}}\,\mathrm{d}z = \int -\frac{1}{2}z^{-\frac{1}{2}}\,\mathrm{d}z = -z^{\frac{1}{2}} + c_2. \tag{4.103}$$

Now, reverse the substitution according to equation (4.101) for obtaining an explicit function of t; hence the general solution of the problem in (4.90) is:

$$x^*(t) = -\left(\lambda^2 - (t - c_1)^2\right)^{\frac{1}{2}} + c_2 \quad (\lambda,c_1,c_2) \in \mathbb{R}^3. \tag{4.104}$$

Notice that in this case λ appears as one of the constants whose value can be computed (together with that of the other constants c_1,c_2) by using the constraint and the boundary conditions:

$$\begin{cases} x(0) = -\left(\lambda^2 - c_1^2\right)^{\frac{1}{2}} + c_2 = 0 \\ x(T) = -\left(\lambda^2 - (T - c_1)^2\right)^{\frac{1}{2}} + c_2 = 0 \\ \displaystyle\int_0^T \left(1 + \dot{x}^2\right)^{\frac{1}{2}}\,\mathrm{d}t = \int_0^T \left(1 + (t - c_1)^2\left(\lambda^2 - (t - c_1)^2\right)\right)^{\frac{1}{2}}\,\mathrm{d}t = k \end{cases} \tag{4.105}$$

Recalling that $T \in \mathbb{R}$ and $k \in \mathbb{R}$ are constant values, by solving the system (4.105) we get the values of λ,c_1,c_2.

Remark 4.6.1 (Interpretation note) *Recall that λ in the static optimization case with equality constraints was interpreted as the (marginal) effect on the value function of releasing the constraint (also called the "shadow price" in some particular economic applications). A very similar reasoning can be applied here: notice that we have a constrained maximization problem, with one equality constraint. The only difference is that here we are facing a dynamic optimization problem, hence the solution is not a point but a function which depends on the value of λ (see equation (4.104)). Nonetheless, the interpretation of λ is quite similar: it is the marginal value of the parameter k, that is, the marginal effect that the release of the equality constraint (which here is given by an integral equation, instead of a "simple" equation) is has on the optimum. The latter is given (as in the static case) simply by plugging in the "objective integral":*

$$V(k) = \int_0^T x^*(t;k)\,\mathrm{d}t \tag{4.106}$$

where $x^*(t;k)$ has been used to stress the fact that the values of λ,c_1,c_2 obtained from solving the system (4.105) will depend on the value of the parameter k, as a consequence also the solution function (after plugging into λ,c_1,c_2) will depend in k. Analogously to the static optimization case, it is possible to prove that the first derivative of the optimum with respect to the parameter equals the value of the multiplier λ, that is:

$$\frac{\mathrm{d}V(k)}{\mathrm{d}k} = \lambda. \tag{4.107}$$

□

Exercise 4.6.3 *Solve the following calculus of variations problem:*

$$\begin{cases} \max \int_0^1 1 - x^2 - \dot{x}^2 \, \mathrm{d}t \\ s.t. \\ \quad x(0) = 1 \\ \quad x(1) \textit{ free} \end{cases} \tag{4.108}$$

where, in case of free terminal value of the solution, the boundary condition is replaced by the requirement (transversality condition):

$$\left(\frac{\mathrm{d}F(t,x^*,\dot{x}^*)}{\mathrm{d}\dot{x}} \right)_{t=t_1} = 0. \tag{4.109}$$

SOLUTION: *This problem has one of the boundary conditions different free, which intuitively means that we do not case about the value that the solution function takes at one boundary extreme (in this case at $t = 1$). In any case, the procedure for finding out a candidate solution does not involve the boundary conditions at the first step, since it requires to determine the general solution of the Euler equation. As a consequence we construct and solve it:*

$$F_2^{'}(t,x,\dot{x}) = -2x \tag{4.110}$$

$$F_3^{'}(t,x,\dot{x}) = -2\dot{x} \tag{4.111}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} F_3^{'}(t,x,\dot{x}) = -2\ddot{x} \tag{4.112}$$

Hence the Euler equation is:

$$-2x + 2\ddot{x} = 0 \quad \Rightarrow \quad \ddot{x} - x = 0 \tag{4.113}$$

which can be solved by recognizing that it is a second order liner, homogeneous differential equation with constant coefficients and applying the corresponding solving method, which requires to compute the roots of the associated characteristic polynomial:

$$\lambda^2 - 1 = 0 \quad \Rightarrow \quad \begin{cases} \lambda_1 = 1 \\ \lambda_2 = -1 \end{cases} \tag{4.114}$$

hence the general solution of the problem is:

$$x^*(t) = c_1 e^t + c_2 e^{-t} \quad (c_1,c_2) \in \mathbb{R}^2. \tag{4.115}$$

The values of the coefficients (c_1,c_2) can be computed using the boundary condition $x(0) = 1$ and by imposing a transversality condition in substitution to the "free" value of the solution function at the final extreme $t = 1$. In case of "free" terminal condition the corresponding transversality condition is:

$$\left(\frac{\mathrm{d}F(t,x^*,\dot{x}^*)}{\mathrm{d}\dot{x}} \right)_{t=t_1} = 0. \tag{4.116}$$

In this case this is:

$$\left(\frac{\mathrm{d}F(t,x^*,\dot{x}^*)}{\mathrm{d}\dot{x}} \right)_{t=t_1} = \left(\frac{\mathrm{d}(1 - x^* - \dot{x}^*)}{\mathrm{d}\dot{x}} \right)_{t=1} = 0. \tag{4.117}$$

For computing (4.117) first of all compute the first order derivative of the general solution:

$$\dot{x}^*(t) = c_1 e^t - c_2 e^{-t}. \tag{4.118}$$

Finally, the values of the coefficients (c_1,c_2) *can be obtained as the solution of the system:*

$$\begin{cases} x(0) = c_1 + c_2 = 1 \\ -2\dot{x}^*(1) = c_1 e - c_2 e^{-1} = 0 \end{cases} \Rightarrow \begin{cases} c_1 = 1 - c_2 \\ (1-c_2)e - c_2 e^{-1} = 0 \end{cases} \tag{4.119}$$

$$\begin{cases} c_1 = 1 - c_2 \\ e - c_2(e + \frac{1}{e}) = 0 \end{cases} \Rightarrow \begin{cases} c_1 = 1 - c_2 = 1 - \frac{e^2}{e^2+1} = \frac{1}{e^2+1} \\ c_2 = e\left(\frac{e^2+1}{e}\right)^{-1} = \frac{e^2}{e^2+1} \end{cases} . \tag{4.120}$$

Therefore the final solution of problem (4.108) *is:*

$$x^*(t) = \frac{1}{e^2+1}e^t + \frac{e^2}{e^2+1}e^{-t} \tag{4.121}$$

□

4.7 Dynamic Programming

Exercise 4.7.1 *Solve the following dynamic programming problem*[12] *using the backward induction method:*

$$\begin{cases} \max \sum_{t=0}^{3}(1 + x_t - u_t^2) \\ s.t. \\ \quad x_{t+1} = x_t + u_t \;\; t = 0,1,2,3 \\ \quad u_t \in \mathbb{R} \;\; t = 0,1,2,3 \\ \quad x_0 = 0 \end{cases} \tag{4.122}$$

SOLUTION: *The backward induction method consists of a set of static optimization problems, starting from the last period of the original dynamic optimization problem. Therefore it is important to (i) understand which is the final time. As a general eyeball rule, it is possible to look the greatest* t *for which the dynamic constraint* $g(t,x_t,u_t)$*, which describes the evolution of the state variable* x_t *in response to the control variable (action)* u_t*, is defined and add one period. The reason for this is that, if* $g(t,x_t,u_t)$ *is defined for* $t \in [0,k]$*, then at the final time* $t = k$ *it will be:*

$$x_{k+1} = g(k,x_k,u_k) \tag{4.123}$$

hence a value x_{k+1} *for the state variable at time* $k+1$ *is defined. As a consequence the final time is* $k+1$ *and the whole time span is* $[0,k+1]$*. Next, (ii) understand the reward at the final time. Three cases may occur:*

- *it is specified in the objective function separately (*$\psi(\cdot)$ *is called scrap value function):*

$$\max \sum_{t=0}^{k} f(t,x_t,u_t) + \psi(x_{k+1}) \tag{a}$$

  then use $V_{k+1}(x_{k+1}) = \psi(x_{k+1})$*;*

- *it is specified in the objective function not separately, since the horizon of the sum includes* $k+1$*:*

$$\max \sum_{t=0}^{k+1} f(t,x_t,u_t) \tag{b}$$

  then use $V_{k+1}(x_{k+1}) = f(k+1,x_{k+1},u_{k+1})$*;*

[12] Here it used the notation (alternative in brackets): x_t (s_t) is the state variable and u_t (a_t) is the action or control variable; $f(\cdot)$ ($r(\cdot)$) is the reward function and $g(\cdot)$ ($f(\cdot)$) is the transition function from a period to the next one.

- *it is not specified at all:*

$$\max \sum_{t=0}^{k} f(t,x_t,u_t) \tag{c}$$

then use $V_{k+1}(x_{k+1}) = 0$.

In problem (4.122) *the constraint is defined up to* $t = 3$, *hence the final time is* $t = 4$. *We begin the backward method from here, as follows.*

- *FIRST STEP t=4*
  Since we are in case (c), *it holds:*

$$V_4(x_4) = 0. \tag{4.124}$$

This is evident since at time $t = 4$ *the objective function (or reward function) is not defined, hence it takes value zero. Since* $V_4(x_4)$ *is a scalar, it does not depend on the action* u_t. *We can go directly to the previous period by using the constraint:*

$$x_{t+1} = x_t + u_t \quad \Rightarrow \quad x_4 = x_3 + u_3 \tag{4.125}$$

- *SECOND STEP t=3*
  Construct the Bellman functional equation, exploiting (4.125)*:*

$$V_3(x_3) = \max_{u_3 \in \mathbb{R}} \Big\{ (1 + x_3 - u_3^2) + V_4(x_4) \Big\} \tag{4.126}$$

$$= \max_{u_3 \in \mathbb{R}} \Big\{ (1 + x_3 - u_3^2) + V_4(x_3 + u_3) \Big\} \tag{4.127}$$

$$= \max_{u_3 \in \mathbb{R}} \Big\{ (1 + x_3 - u_3^2) \Big\} \tag{4.128}$$

which is a static optimization problem in u_3. *Notice that the optimization is carried out with respect to the control (action) variable* u_t, *not the state variable* x_t. *This is logic, given that* u_t *is the action, that is the variable that can be directly modified, while the state is influenced indirectly be the action via the dynamic constraint.*

Remark 4.7.1 *In this case this problem is unconstrained since* $u_3 \in \mathbb{R}$, *that is* u_t *is allowed to take any value. In other cases when* $u_t \in \mathbb{D}$ *(for example the closed interval* $\mathbb{D} = [5,10]$*) the problem would become an inequality constrained one (continuing the example, the constraints will be* $u_t \geq 5$ *and* $u_t \leq 10$*). One can solve it by using the Khun-Tucker conditions or, more simply, by looking for stationary points (as in the unconstrained case), then comparing the value of the function* $V_t(x_t)$ *to be maximized at the stationary point* u_t^* *and and the boundaries (in the example at* $u_t = 5$ *and* $u_t = 10$*), and choosing the value yielding the highest value of* $V_t(x_t)$.

The unique stationary point is $u_3^* = 0$, *and, since the function is concave in* u_t, *it is the unique maximum point; therefore substituting in the value function we obtain:*

$$V_3(x_3) = 1 + x_3 \tag{4.129}$$

and from the dynamic constraint: $x_3 = x_2 + u_2$.

- *THIRD STEP t=2*
  The Bellman functional equation becomes:

$$V_2(x_2) = \max_{u_2 \in \mathbb{R}} \Big\{ (1 + x_2 - u_2^2) + V_3(x_3) \Big\} \tag{4.130}$$

$$= \max_{u_2 \in \mathbb{R}} \Big\{ (1 + x_2 - u_2^2) + 1 + x_3 \Big\} \tag{4.131}$$

$$= \max_{u_2 \in \mathbb{R}} \Big\{ (1 + x_2 - u_2^2) + 1 + x_2 + u_2 \Big\} \tag{4.132}$$

$$= \max_{u_2 \in \mathbb{R}} \Big\{ 2 + 2x_2 - u_2^2 + u_2 \Big\} \tag{4.133}$$

  where the first equality comes from plugging in the value function $V_3(x_3)$ previously computed, while the second from the application of the dynamic constraint. The unique stationary point is $u_2^* = \frac{1}{2}$, which must be plugged into the value function for obtaining:

$$V_2(x_2) = 2x_2 + \frac{9}{4}. \tag{4.134}$$

  From the dynamic constraint: $x_2 = x_1 + u_1$.

- *FOURTH STEP t=1*
  The Bellman functional equation becomes:

$$V_1(x_1) = \max_{u_1 \in \mathbb{R}} \Big\{ (1 + x_1 - u_1^2) + V_2(x_2) \Big\} \tag{4.135}$$

$$= \max_{u_1 \in \mathbb{R}} \left\{ (1 + x_1 - u_1^2) + 2x_2 + \frac{9}{4} \right\} \tag{4.136}$$

$$= \max_{u_1 \in \mathbb{R}} \left\{ (1 + x_1 - u_1^2) + 2(x_1 + u_1) + \frac{9}{4} \right\} \tag{4.137}$$

$$= \max_{u_1 \in \mathbb{R}} \left\{ \frac{13}{4} + 3x_1 - u_1^2 + 2u_1 \right\} \tag{4.138}$$

  whose unique stationary point is $u_1^* = 1$. Then following the same procedure as in the previous steps:

$$V_1(x_1) = 3x_1 + \frac{17}{4} \tag{4.139}$$

  and the dynamic constraint: $x_1 = x_0 + u_0$.

- *FIFTH STEP t=0*
  The Bellman functional equation becomes:

$$V_0(x_0) = \max_{u_0 \in \mathbb{R}} \Big\{ (1 + x_0 - u_0^2) + V_1(x_1) \Big\} \tag{4.140}$$

$$= \max_{u_0 \in \mathbb{R}} \left\{ (1 + x_0 - u_0^2) + 3x_1 + \frac{17}{4} \right\} \tag{4.141}$$

$$= \max_{u_0 \in \mathbb{R}} \left\{ (1 + x_0 - u_0^2) + 3(x_0 + u_0) + \frac{17}{4} \right\} \tag{4.142}$$

$$= \max_{u_0 \in \mathbb{R}} \left\{ \frac{21}{4} + 4x_0 - u_0^2 + 3u_0 \right\} \tag{4.143}$$

  whose unique stationary point is $u_0^* = \frac{3}{2}$.

Now that all the optimal values of the control variable have been computed and it is know the initial value of the state variable, given in (4.122), we can exploit the dynamic constraint and recover the sequence of values of the state variable as follows:

$$\begin{cases} x_0^* = 0 \\ x_1^* = x_0^* + u_0^* \\ x_2^* = x_1^* + u_1^* \\ x_3^* = x_2^* + u_2^* \\ x_4^* = x_3^* + u_3^* \end{cases} \quad \Rightarrow \quad \begin{cases} x_0^* = 0 \\ x_1^* = 0 + \frac{3}{2} = \frac{3}{2} \\ x_2^* = \frac{3}{2} + 1 = \frac{5}{2} \\ x_3^* = \frac{5}{2} + \frac{1}{2} = 3 \\ x_4^* = 3 + 0 = 3 \end{cases} \tag{4.144}$$

Therefore the solution of the problem (4.122) is the sequence of pairs:

$$\left\{(x_t^*, u_t^*)\right\}_{t=0}^{4} = \left\{ \left(0, \frac{3}{2}\right), \left(\frac{3}{2}, 1\right), \left(\frac{5}{2}, \frac{1}{2}\right), (3,0), (3,\cdot) \right\}. \tag{4.145}$$

Notice that, since the optimal controls u_t^* depends only on time (and not on the state variable), they are called open-loop controls.

□

Exercise 4.7.2 *Solve the following dynamic programming problem using the backward induction method:*

$$\begin{cases} \max \sum_{t=0}^{2} \ln(u_t) \\ s.t. \\ \quad x_{t+1} = \frac{11}{10} x_t - u_t \quad t = 0,1,2 \\ \quad u_t > 0 \quad t = 0,1,2 \\ \quad x_0 = 1 \\ \quad x_3 = \frac{121}{100} \end{cases} \tag{4.146}$$

SOLUTION: *Start with the determination of the final time of this problem. Clearly it is not 2, since there is a condition for the state variable at time $t = 3$, however, by applying the same way of reasoning described in the previous exercise, that is, by checking the highest t for which the dynamic constraint is defined and adding one, the same result is obtained.*

Given that the final time is 3, it now necessary to define the value function to be maximized at the final time: since the sum ends at $t = 2$ and there is no scrap value function, this is an example of case (c).

It is now possible to start the backward induction algorithm from the final time as follows:

- *FIRST STEP t=3*
  The value function is:

$$V_3(x_3) = 0 \tag{4.147}$$

  since the objective function (reward function) depends only on the control variable u_t, but at $t = 3$ there is no control defined (it is defined for $t = 0,1,2$). The dynamic constraint gives: $x_3 = \frac{11}{10} x_2 - u_2$. In ordinary problems where the state variable appears in the revenue function it will be necessary to use the dynamic constraint for substituting for the state variable and "come back" by one period in this way. However this is not the case, since the problem (4.146) contains

only the control variable. Therefore it is possible to exploit the dynamic constraint in order to recover the control:

$$x_3 = \frac{11}{10}x_2 - u_2 \quad \Rightarrow \quad u_2 = \frac{11}{10}x_2 - x_3 = \frac{11}{10}x_2 - \frac{121}{100} \tag{4.148}$$

where the last equality exploits the knowledge of the terminal value of the state variable as given in the problem (4.146).

- *SECOND STEP t=2*
  The Bellman functional equation is:

$$V_2(x_2) = \max_{u_2=\frac{11}{10}x_2-\frac{121}{100}} \left\{\ln(u_2) + V_3(x_3)\right\} \tag{4.149}$$

$$= \ln\left(\frac{11}{10}x_2 - \frac{121}{100}\right) \tag{4.150}$$

  NOTE: remind that the control variable at time $t = 2$ is constrained by the dynamic constraint: $u_2 = \frac{11}{10}x_2 - \frac{121}{100}$. As a consequence we are forced to choose the control satisfying this constraint. Of course, the precise value of the control is still unknown since it depends on the value of the state at the same time (x_2), but we will recover it later.

  Now, it is possible to use the dynamic constraint in the ordinary way to find out the state variable the previous time period, obtaining:

$$x_2 = \frac{11}{10}x_1 - u_1 \tag{4.151}$$

- *THIRD STEP t=1*
  The Bellman function equation is:

$$V_1(x_1) = \max_{u_1>0} \left\{\ln(u_1) + V_2(x_2)\right\} \tag{4.152}$$

$$= \max_{u_1>0} \left\{\ln(u_1) + \ln(u_2)\right\} \tag{4.153}$$

$$= \max_{u_1>0} \left\{\ln(u_1) + \ln\left(\frac{11}{10}x_2 - \frac{121}{100}\right)\right\} \tag{4.154}$$

$$= \max_{u_1>0} \left\{\ln(u_1) + \ln\left(\frac{11}{10}\left(\frac{11}{10}x_1 - u_1\right) - \frac{121}{100}\right)\right\} \tag{4.155}$$

$$= \max_{u_1>0} \left\{\ln(u_1) + \ln\left(\frac{121}{100}(x_1 - 1) - \frac{11}{10}u_1\right)\right\} \tag{4.156}$$

  whose unique stationary point is $u_1^* = \frac{55}{100}(x_1 - 1)$, so it is not an open-loop control, as it depends on the value of the state variable. It should be reminded, when the solution x_1 becomes available, to check that the control variable u_1 satisfies $u_1 > 0$.

  The dynamic constraint yields: $x_1 = \frac{11}{10}x_0 - u_0$.

- *FOURTH STEP t=0*

The Bellman function equation is:

$$V_0(x_0) = \max_{u_0>0} \left\{ \ln(u_0) + V_1(x_1) \right\} \tag{4.157}$$

$$= \max_{u_0>0} \left\{ \ln(u_0) + \left(\ln(u_1) + \ln\left(\frac{121}{100}(x_1 - 1) - \frac{11}{10}u_1 \right) \right) \right\} \tag{4.158}$$

$$= \max_{u_0>0} \left\{ \ln(u_0) + \left(\ln\left(\frac{55}{100}(x_1 - 1) \right) + \ln\left(\frac{121}{100}(x_1 - 1) - \frac{11}{10}\left(\frac{55}{100}(x_1 - 1) \right) \right) \right) \right\} \tag{4.159}$$

$$= \max_{u_0>0} \left\{ \ln(u_0) + \left(\ln\left(\frac{55}{100}x_1 - \frac{55}{100} \right) + \ln\left(\frac{121}{100}x_1 - \frac{121}{100} - \frac{605}{1000}x_1 + \frac{11}{10} \right) \right) \right\} \tag{4.160}$$

$$= \max_{u_0>0} \left\{ \ln(u_0) + \left(\ln\left(\frac{55}{100}x_1 - \frac{55}{100} \right) + \ln\left(\frac{121}{200}x_1 - \frac{1}{10} \right) \right) \right\} \tag{4.161}$$

$$= \max_{u_0>0} \left\{ \ln(u_0) + \left(\ln\left(\frac{55}{100}\left(\frac{11}{10}x_0 - u_0 \right) - \frac{55}{100} \right) + \ln\left(\frac{121}{200}\left(\frac{11}{10}x_0 - u_0 \right) - \frac{1}{10} \right) \right) \right\} \tag{4.162}$$

$$= \max_{u_0>0} \left\{ \ln(u_0) + \left(\ln\left(-\frac{55}{100}u_0 - \frac{55}{100} \right) + \ln\left(-\frac{121}{200}u_0 - \frac{1}{10} \right) \right) \right\} \tag{4.163}$$

where it has been exploited the initial value of the state variable given in problem (4.146) as well as the dynamic constraint and the value function in the time $t = 1$. The unique stationary point is: $u_0^* = \frac{11}{30}x_0 - \frac{1}{3} = \frac{1}{30}$.

The general solution can now be reconstructed as follows: the known values are the initial and final values of the state and the initial value of the control. Exploiting the dynamic constraint it is possible to find out all the others.

$$\begin{cases} x_0^* = 1 \\ u_0^* = \frac{1}{30} \\ x_1^* = \frac{11}{10}x_0^* - u_0^* \\ u_1^* = \frac{55}{100}(x_1^* - 1) \\ x_2^* = \frac{11}{10}x_1^* - u_1^* \\ u_2^* = \frac{11}{10}x_2^* - x_3^* \\ x_3^* = \frac{121}{100} \end{cases} \quad \Rightarrow \quad \begin{cases} x_0^* = 1 \\ u_0^* = \frac{1}{30} > 0 \\ x_1^* = \frac{16}{15} \\ u_1^* = \frac{55}{100}(x_1^* - 1) = \frac{11}{300} > 0 \\ x_2^* = \frac{341}{300} \\ u_2^* = \frac{11}{10}x_2^* - x_3^* = \frac{121}{3000} > 0 \\ x_3^* = \frac{121}{100} \end{cases} . \tag{4.164}$$

All controls are strictly positive as required in problem (4.146), hence the system (4.164) yields the solution sequences of the state and control variables:

$$\left\{ (x_t^*, u_t^*) \right\}_{t=0}^{3} = \left\{ \left(1, \frac{1}{30} \right), \left(\frac{16}{15}, \frac{11}{300} \right), \left(\frac{341}{300}, \frac{121}{3000} \right), \left(\frac{121}{100}, \cdot \right) \right\}. \tag{4.165}$$

□

Chapter 5

Exercises without Solutions

This Chapter provides additional exercises on all the topics discussed so far. No solution is provided.

5.1 Static Optimization: Unconstrained Optimization

Find out whether the following functions admits global/local maxima/minima:

1)

$$f(x,y) = 2x^3 - 3x^2y + \frac{4}{3}y^3 - 4y$$

2)

$$f(x,y,z) = (x-2)^2 + (y+2)(z-1)^2 + y^2$$

3)

$$f(x,y) = x^2 - xy - 6y^2 + ky^4$$

4)

$$f(x,y) = -(x+1)^3 + \alpha(x+1)(y+1) - (y+1)^3 \qquad x \geq 0, y \geq 0$$

5.2 Static Optimization: Equality Constrained Optimization

1) Solve the following optimization problem, then check that the Lagrange multipliers are equal to the gradient of the value function (Envelope Theorem):

$$\begin{cases} \max\limits_{x,y,z} \quad x+y+z \\ s.t. \\ \qquad x^2+y^2 = h \\ \qquad x - z = 1 \end{cases}$$

2) Solve the following optimization problems:

$$\begin{cases} \max\limits_{x,y,z} \quad -x+y+2z \\ s.t. \\ \qquad x^2+y^2 = 10 \\ \qquad z+y-3 = 0 \end{cases} \qquad \begin{cases} \max\limits_{x,y,z} \quad x \\ s.t. \\ \qquad x^2+y^2+z^2 = \frac{5}{2} \\ \qquad z+y = 1 \end{cases} \qquad \begin{cases} \max\limits_{x,y,z} \quad xyz \\ s.t. \\ \qquad x^2+y^2 = 1 \\ \qquad x+z = 1 \end{cases}$$

3) Solve the following optimization problems:

$$\begin{cases} \max\limits_{x,y} & y - x^2 \\ s.t. & \\ & x + y = 4 \end{cases} \qquad \begin{cases} \max\limits_{x,y} & \ln(x) + y \\ s.t. & \\ & px + y = 4 \quad \text{with } p > 0 \end{cases} \qquad \begin{cases} \max\limits_{x,y,z} & x + y + z \\ s.t. & \\ & xyz = 1 \end{cases}$$

5.3 Static Optimization: Inequality Constrained Optimization

1) Solve the following optimization problems:

$$\begin{cases} \max\limits_{x,y} & \frac{2}{3}x - \frac{1}{2}x^2 + \frac{1}{12}y \\ s.t. & \\ & x \leq 5 \\ & y - x \leq 1 \\ & x \geq 0 \\ & y \geq 0 \end{cases} \qquad \begin{cases} \max\limits_{x,y,z} & -x^2 + 2y - z^2 \\ s.t. & \\ & x + y + z \leq 8 \\ & y \leq 0 \\ & x \leq 0 \end{cases}$$

2) Solve the following optimization problem (use both the necessary and sufficient KKT conditions):

$$\begin{cases} \max\limits_{x,y,z} & -\left(e^{-x} + e^{-y}\right)^2 - 2z \\ s.t. & \\ & z \geq 0 \\ & y \leq 0 \\ & x \geq 0 \end{cases} \qquad \begin{cases} \min\limits_{x,y} & -(x+y)^2 \\ s.t. & \\ & xy \geq 1 \\ & y \geq (x-2)^2 \end{cases} \qquad \begin{cases} \min\limits_{x,y} & x^2 y e^{-x-y} \\ s.t. & \\ & x \geq 1 \\ & y \geq 1 \\ & x + y \geq 4 \end{cases}$$

3) Solve the following optimization problems:

$$\begin{cases} \min\limits_{x,y} & 2x^2 - xy - y^2 \\ s.t. & \\ & x \leq 3 \\ & y \geq -1 \end{cases} \qquad \begin{cases} \min\limits_{x,y,z} & -\left(e^{-x} + e^{-y}\right)^2 - 2z \\ s.t. & \\ & x \leq 0 \\ & y \leq 4 \\ & z \geq 0 \end{cases} \qquad \begin{cases} \min\limits_{x,y,z} & -x^2 + 2y - z^2 \\ s.t. & \\ & x + y + z \leq 8 \\ & y \leq 4 \\ & x \leq 0 \end{cases}$$

4) Given the following optimization problems, determine whether the objective function is concave on the admissibility region, then solve the problem:

$$\begin{cases} \max\limits_{x,y} & x^2 y e^{-x-y} \\ s.t. & \\ & x \geq 1 \\ & y \geq 1 \\ & x + y \geq 4 \end{cases}$$

5.4 Concavity/Convexity

1) According to the value of $\alpha \in \mathbb{R}$, study the definitness of the matrix A (positive/negative definitness, positive/negative semidefinitness, indefinitness):

$$\mathbf{A} = \begin{bmatrix} 2\alpha & -8 & 0 \\ -8 & 2\alpha & 0 \\ 0 & 0 & 1-\alpha \end{bmatrix}$$

2) Determine whether and in which region the following functions are concave or convex *(Hint: draw the graph, then apply a known result)*:

$$f(x,y) = x^2e^{-x} + y^2 - 2y \qquad f(x,y) = -3e^y + \ln(x) - x^2 + x + \sqrt{y}$$
$$f(x) = x^3 - x \qquad f(x) = x^4 - x^2$$
$$f(x) = x^3 + x$$

3) Given the following problem:

$$\begin{cases} \max\limits_{x,y} & -(x+y)^2 \\ s.t. & \\ & y \geq (x-2)^2 \\ & xy \geq -1 \end{cases}$$

determine whether the fixed point $(x^*, y^*) = (1, 1)$ satisfies the KKT necessary conditions; in addition, check whether the KKT sufficient conditions can be applied. *(OPTIONAL: solve the problem from the start.)*

5.5 Differential Equations

1) Find a general solution of the following first order Ordinary Differential Equations (ODEs):

$$\dot{x} = 2x + t^2 - 4 \qquad \dot{x} = t^2 - 4$$
$$\dot{x} = 3x + 2e^{5t} \qquad \dot{x} = 3x + 2e^{3t}$$
$$x\dot{y} - 2y = x^2 \qquad \dot{y} + 2xy = 2x$$
$$\frac{dy}{dx} + \left(\frac{1}{x}\right) y = 3x + 4 \qquad (x-1)\dot{y} + y = x^2 - 1$$
$$\dot{x} = 2x + 8 \qquad \dot{x} = \alpha x + \beta\alpha^t$$

2) Find the unique solution of the following first order ODEs:

$$\dot{x} = 3x + e^{3t}, \;\; x(0) = 1 \qquad \dot{x} = -4x + 3e^{2t}, \;\; x(0) = 2$$

3) Find a general solution of the following first order ODEs, then find the steady-state value and comment on its stability:

$$\dot{x} = 5x + t + 2 \qquad \dot{x} = -3x + 18$$

4) Find a general solution (and the unique, when it is possible) to the following second order ODEs:

$\ddot{x} - 7\dot{x} + 6x = 12$ $\qquad$ $\ddot{x} - 4\dot{x} + 3x = 6$

$\ddot{x} + 6\dot{x} + 9x = 27, \;\; x(0) = 6, \, \dot{x}(0) = 2$ $\qquad$ $\ddot{x} - 4\dot{x} + 13x = 26, \;\; x(0) = 5 \, \dot{x}(0) = 2$

$4\ddot{x} - 4\dot{x} + x = 0, \;\; x(0) = 1, \, \dot{x}(0) = -\dfrac{1}{2}$ $\qquad$ $\ddot{x} - 2\dot{x} + x = 0$

$\ddot{x} - 4\dot{x} + 40x = 80 + 36t + e^{2t}$ $\qquad$ $\ddot{x} + 6\dot{x} + 9x = 75 + 30t + e^{-3t}$

$\ddot{x} + 4\dot{x} + 4x = 4t^2 + 8t + 18 + e^{-2t}$ $\qquad$ $\ddot{x} - 3\dot{x} = 9t^2$

$\ddot{x} = 12t + 6$ $\qquad$ $\ddot{x} - 2\dot{x} + x = 6e^t$

5) Find a general solution to the following systems of first order ODEs:

$$\begin{cases} \dot{x} = 2x \\ \dot{y} = x - 3y \end{cases} \qquad \begin{cases} \dot{x} = 5x - y \\ \dot{y} = x + 3y \end{cases}$$

6) Discuss both analytically and qualitatively (i.e. graphically) the stability of the equilibrium points of the following first order ODEs, after having drawn the graph:

$\dot{x} = -x^2 \ln(x), \quad x(0) > 0$ $\qquad$ $\dot{x} = \dfrac{4 - x}{3x + 2}$

$\dot{x} = e^{-x} - 1$

7) For each of the following systems of first order ODEs, check whether it is stable, unstable or a saddle:

$$\begin{cases} \dot{x} = 10x + 3y + 2 \\ \dot{y} = -3x + y + 1 \end{cases} \qquad \begin{cases} \dot{x} = x + 3y + 10 \\ \dot{y} = -2x + y - 5 \end{cases}$$

$$\begin{cases} \dot{x} = 2x - 6y - 1 \\ \dot{y} = -3x + 5y + 2 \end{cases} \qquad \begin{cases} \dot{x} = -2x - 4y + 5 \\ \dot{y} = -2x - 9y + 1 \end{cases}$$

8) Solve the following systems of first order ODEs and draw the corresponding phase diagrams:

$$\begin{cases} \dot{x} = -3y + 6 \\ \dot{y} = x - 4y \end{cases} \qquad \begin{cases} \dot{x} = y - 2 \\ \dot{y} = \frac{1}{4}x + y - 1 \end{cases}$$

9) Solve the following separable, exact or linear differential equations:

$\dot{x}(t + 1) = e^t - 2x$ $\qquad$ $2(2t^2 + x^2)\mathrm{d}t + 4(tx)\mathrm{d}x = 0$

$\dfrac{\mathrm{d}y}{\mathrm{d}x} = x - 2y$ $\qquad$ $(x + 1)^2 \dfrac{\mathrm{d}y}{\mathrm{d}x} + 2(x + 1)y = e^x$

$(2tx + t^2 + t^4)\mathrm{d}t + (1 + t^2)\mathrm{d}x = 0$ $\qquad$ $(2t^3 - tx^2 - 2x + 3)\mathrm{d}t - (t^2x + 2t)\mathrm{d}x = 0$

$e^x \dot{x} = t + 1$ $\qquad$ $t\dot{x} = x(1 - t)$

$1 + tx^2 + t^2 x\dot{x} = 0$ $\qquad$ $1 - (t + 2x)\dot{x} = 0, \quad t > 0, x > 0$

10) Solve the following differential equations:

$\dot{x} + 2tx = 4t$ $\qquad$ $t\dot{x} + 2x + t = 0, \quad t \neq 0$

$\dot{x} - \dfrac{2}{t}x + \dfrac{2a^2}{t^2} = 0, \quad t > 0$ $\qquad$ $\dot{x} - 2tx = t(1 + t^2)$

5.6 Difference Equations

1) Solve the following difference equations:

$$\begin{aligned} &x_{t+1} = 3x_t + t^2 - 4 && x_{t+1} = t^2 - 4 \\ &x_{t+1} = 2x_t + t^2 - t + 3 && x_{t+1} = x_t + t^2 \\ &x_{t+1} = 3x_t + 8 \cdot 2^t && x_{t+1} = 3x_t + 4 \cdot 3^t \end{aligned}$$

2) Find the equilibrium points of the following difference equations and study their stability both analytically and graphically:

$$\begin{aligned} &x_{t+1} = 4x_t(1 - x_t) && x_{t+1} = x_t^2 \\ &x_{t+1} = 3x_t + 2 && x_{t+1} = x_t^3 \end{aligned}$$

3) Solve the following difference equations:

$$\begin{aligned} &x_{t+2} - 4x_{t+1} + 3x_t = t + 2 && x_{t+2} - 5x_{t+1} + 6x_t = t^2 - 1 \\ &x_{t+2} - x_{t+1} - 2x_t = 2^t && x_{t+2} - 10x_{t+1} + 25x_t = 3 \cdot 5^t \end{aligned}$$

4) Solve the following difference equations and study the stability of their equilibrium points:

$$\begin{aligned} &x_{t+2} - \frac{5}{6}x_{t+1} + \frac{1}{6}x_t = 2 && x_{t+2} - \frac{3}{4}x_{t+1} + \frac{1}{8}x_t = 3 \\ &x_{t+2} + \frac{1}{4}x_{t+1} - \frac{1}{4}x_t = 18 \end{aligned}$$

5) Solve the following systems of linear difference equations, then draw the corresponding phase diagram (when possible) and study the stability of the equilibrium points:

$$\begin{cases} x_{t+1} = 2y_t - 3 \\ y_{t+1} = \frac{1}{2}x_t + 1 \end{cases} \qquad \begin{cases} x_{t+1} = ay_t + ck^t \\ y_{t+1} = bx_t + dk^t \end{cases} \qquad a > 0,\ b > 0,\ k \neq ab$$

5.7 Calculus of Variations

1) Solve the following problems:

$$\begin{cases} \min \displaystyle\int_0^2 \dot{x}^2 \, \mathrm{d}t \\ x(0) = 1 \\ x(2) = 4 \end{cases} \qquad \begin{cases} \min \displaystyle\int_1^2 (t\dot{x} + \dot{x}^2) \, \mathrm{d}t \\ x(1) = 0 \\ x(2) = 2 \end{cases} \qquad \begin{cases} \max \displaystyle\int_0^1 \dot{x}^2(1 + x^2) \, \mathrm{d}t \\ x(0) = 1 \\ x(1) = 1 \end{cases}$$

$$\begin{cases} \max \displaystyle\int_0^1 (\dot{x} + 2x(\dot{x} + t) + 4x^2) \, \mathrm{d}t \\ x(0) = 0 \\ x(1) = 0 \end{cases} \qquad \begin{cases} \max \displaystyle\int_0^2 (x + tx - \dot{x}^2) \, \mathrm{d}t \\ x(0) = 0 \\ x(2) = -1 \end{cases} \qquad \begin{cases} \min \displaystyle\int_0^2 (t + 2x)^4 \, \mathrm{d}t \\ x(0) = 0 \\ x(2) = -1 \end{cases}$$

$$\begin{cases} \max \displaystyle\int_0^1 \dot{x}^2 e^{-\dot{x}} \, \mathrm{d}t \\ x(0) = 0 \\ x(1) = 1 \end{cases} \qquad \begin{cases} \max \displaystyle\int_0^1 (1 + t)\dot{x}^2 \, \mathrm{d}t \\ x(0) = 0 \\ x(1) = 1 \end{cases} \qquad \begin{cases} \max \displaystyle\int_0^1 2t^2 + \dot{x}^2 \, \mathrm{d}t \\ x(0) = 0 \\ x(1) = 1 \end{cases}$$

$$\begin{cases} \max \int_0^1 (t - 3x^2 + \dot{x}^2)\, \mathrm{d}t \\ x(0) = 1 \\ x(1) = 1 \end{cases} \qquad \begin{cases} \min \int_0^T e^{-t}\dot{x}^2\, \mathrm{d}t \\ x(0) = 1 \\ x(T) = b \quad T > 0, b \in \mathbb{R} \end{cases} \qquad \begin{cases} \max \int_0^2 (4 + t^2)\dot{x}^2\, \mathrm{d}t \\ x(0) = 1 \\ x(2) = 2 \end{cases}$$

5.8 Dynamic Programming

1) Solve the following optimization problem with the dynamic programming method:

$$\begin{cases} \max \quad (-x_1^2 + 6x_1 - x_2^2 + 5x_2 + x_3^2) \\ s.t. \\ \quad x_1 + x_2 + x_3 = 3 \quad x_i \in \mathbb{N} \end{cases}$$

2) Solve the following dynamic optimization problem with the dynamic programming method:

$$\begin{cases} \max \quad \sum_{t=0}^{3} (1 + x_t - u_t^2) \\ s.t. \\ \quad x_{t+1} = x_t + u_t \quad t = 0,1,2,3 \\ \quad u_t \in \mathbb{R} \quad t = 0,1,2,3 \\ \quad x_0 = 1 \end{cases}$$

3) Solve the following dynamic optimization problem with the dynamic programming method:

$$\begin{cases} \max \quad \sum_{t=0}^{3} (10x_t - \frac{1}{10}u_t^2) \\ s.t. \\ \quad x_{t+1} = x_t + u_t \quad t = 0,1,2,3 \\ \quad u_t \geq 0 \quad t = 0,1,2,3 \\ \quad x_0 = 0 \end{cases}$$

4) Solve the following dynamic optimization problem with the dynamic programming method:

$$\begin{cases} \max \quad \sum_{t=0}^{3} (3 - u_t)x_t^2 \\ s.t. \\ \quad x_{t+1} = x_t u_t \quad t = 0,1,2 \\ \quad u_t \in [0,1] \quad t = 0,1,2,3 \\ \quad x_0 \in \mathbb{R} \end{cases}$$

5) Solve the following dynamic optimization problem with the dynamic programming method:

$$\begin{cases} \max \quad \displaystyle\sum_{t=0}^{2} \left(-\frac{2}{3} u_t \right) + \ln(x_3) \\ s.t. \\ \quad x_{t+1} = x_t(1+u_t) \quad t = 0,1,2 \\ \quad u_t \in [0,1] \quad t = 0,1,2 \\ \quad x_0 \in \mathbb{R}_+ \end{cases}$$

Chapter 6

The Computational Approach

In Chapter 3 we have defined some problems of optimization and discussed the theorems which allow us to find out an analytical solution (when it exists). In practice, however, it is often the case that such calculations become cumbersome or impossible to be undertaken by hand, because of their intrinsic complexity, the dimensionality of the problem or many other reasons. It has become necessary to find out new ways to carry out these operations as fast and as precise as possible: this is the goal of the computational approach to mathematics.

In this Chapter we will point out the main features of this approach, starting from a general introduction to programming, then we will perform a direct comparison with the previous one for highlighting the advantages and the drawbacks of each of them. Finally, a series of applications is provided, one for each of the problems we previously discussed in Chapter 3.

The softwares/languages presented are MATLAB®[1] and R[2], however the initial information provided hold true for any sofware/programming language one would use.

The main references for this Chapter are: [8] for a general introduction to the use of computational methods in economics; [2] and [6] for MATLAB applications; [4], [9] and [13] for what concerns the use of R. Finally [11, ch. 5] gives some useful insights on the logic of numerical optimization.

6.1 Foundations of Programming

First of all, when we approach to the use of whatever software for numerical or symbolic computing one should keep in mind that every software requires you to use a specific programming language, which has to be understood by both the user and the machine (computer). There is a well known trade-off between interpreted languages, which are user-friendly but require to be "translated" into a set of instructions more suitable to be understood by the computer, which has a cost in terms of execution time of the program and not interpreted languages, which are generally harder to understand by the user, but faster to be executed by the machine. Among the first group we can identify many famous softwares, like MATLAB, R and others; on the other hand, we can signal C/C++, FORTRAN and many others.

Apart from this first distinction concerning the type of programming language, there are a lot of differences between computing softwares, the main are:

- numerical vs symbolic calculus -oriented. The first class performs operations with variables that represent numbers, while the second one is able to treat variables as functions (therefore can perform, to a given extent, operations like differentiation). MATLAB and R are devoted to numerical calculus, with some toolboxes/packages allowing symbolic operations;
- different syntax, however the way of reasoning and constructing algorithms is independent from the syntax of the chosen language;

[1] MathWorks webpage: http://uk.mathworks.com/products/matlab/.
[2] CRAN webpage: https://cran.revolutionanalytics.com/index.html.

Priority	Method	Time required	Possibility of error
1.	look for a specific toolbox/package (collections of functions)	Low	Low*
2.	ask on a forum dedicated to the software we are using	Medium	Medium*
3.	search the web for user made solutions	Low	High*
4.	create the function on our own	High	High

*Need to check whether the proposed functions really solve the problem

Table 6.1: Methods for searching for functions.

- different libraries (collections of functions) already available for use. When no library is available, the user has to create by himself the functions needed.

According to these features, a software may be "more suited" to carry out a particular operation, with respect to its competitors, but theoretically we can do all that we need using any language, though with efficiency losses, since the bottom line difference is the time spent in programming and the running time of the algorithm. Next, we provide a wide definition of algorithm, a concept that we will use very frequently in this Chapter.

Definition 6.1.1 *An **algorithm** (or function, routine, procedure, program) is a detailed and organic set of instructions that, starting from a given set of inputs (even an empty set) is able to produce a set of outputs.*

Assume now that we have chosen a specific software or programming language to use for doing computational mathematics. In the problem solving process many functions may be needed, so one natural question that comes out is whether there exist or not an "already prepared" function which does exactly what we intend to do. There are various ways to address the issue of finding such best function, the most common ways are listed in Table (6.1).

Remark 6.1.1 *The official material may be standard and unable to solve the particular issues of the problem at hand. By contrast, the online material (forums in particular) may propose many solutions, but it is necessary to check that they are really dealing with the required task.*

Remark 6.1.2 *It is **fundamental to check** all the functions that are found on the web or in toolboxes/packages in order to be sure that they solve exactly the specific task they are required to address and that they do not contain bugs (i.e. hidden errors in the program).*

Given a task to solve, it is possible to proceed as described in Algorithm (5); however a non trivial attention should be given to the choice of the keywords when attempting a online research. In this case it is often necessary to use the appropriate lexicon: looking for specific terms may lead to very good results or to none[3].

Here are some examples of online searches[4] which may shed some light on the outcomes that may occur in practice: in some cases we are able to find out the right solution; in some others this is hampered by misleading answers, which after an accurate check reveal not to be apt to solve the problem at hand; further searches are shown to strongly depend on the keywords used.

[3]The output of any online research extremely depends on the input keywords and usually does not yield a "global maximum", that is the best solution among all possible, but only a "local" one, which may perform poorly.

[4]Last check: December 2015.

Algorithm 5 Search for functions

```
1: procedure SEARCH(task)
2:     use one of the methods in Table (6.1)
3:     if method solves the task then
4:         go to Step 11
5:     else
6:         back to Step 2
7:     end if
8:     if no existing methods then
9:         create new algorithm or use different programming language
10:    end if
11:    check that final algorithm actually solves the problem
12:    return Implement the method
13: end procedure
```

Example 6.1.1 (R: 3D plot) *Google "3D plot in R" (link). The output lists many solutions, hence it is necessary to check each one and choose that with the best fit, by checking which one (if any) actually solves the task.*

■

Example 6.1.2 (MATLAB: quiver colors) *Google "matlab color quiver" (link). The first link points to a forum; the second to official material; while the third promotes a user-created specific function (which does what, precisely?).*

■

Example 6.1.3 (MATLAB: cobweb) *Google "matlab difference equation phase plane" (link), there are no pertinent results; while if you Google "matlab cobweb diagram" (link), the second link points to a possible solution to the problem.*

■

Now that we have given a brief overview of how what to do in order to carry out a specific computational problem, possibly avoiding the creation of a new algorithm, it is necessary to pay attention also to the process of algorithm creation. This is a very delicate and complex procedure, since many times even the most expert programmer may make mistakes that remain hidden, so that an output is produced but it is wrong and there is no way to quickly find it out. A first choice we can make is between the analytical and the numerical (computational) approach. The first one is concerned with the use of the tools developed in Chapter 3, while the second refers to the direct use of softwares. Notice that in practice both of them are used, since complex tasks are decomposed in simpler ones which are then solved in most efficient way, by using the most suitable set of tools. Nonetheless, a comparison between the two approaches is useful to point out some critical differences and for being able to choose the most suitable way to carry out a particular operation. Table (6.2) summarizes the most striking points of strength and weakness of both approaches. It is worthwhile to remark that analytical methods are actually implemented by softwares, so programming does not necessarily mean using numerical methods.

Among the most well known practical advantages of the analytical approach, we would like to stress that it provides exact solutions, which has also the remarkable advantage of favouring fast implementations when programming. By contrast, it is not able to solve any kind of problem and in this respect numerical methods may perform better, being able to provide solutions in a wider range of cases. However, when dealing with a complex task numerical methods most often rely on the derivation of an exact solution to a simplified (approximated so that it become more easily tractable)

Approach	Advantages	Drawbacks
Analytical	(i) exact solution (ii) fast computation	(i) only few problems can be solved
Numerical	(i) can solve a high number of problems (ii) can solve some problems with no analytical solutions	(i) computationally (time) expensive (ii) approximate solution or solution of an approximate problem (iii) problems with local optima, convergence, ...

Table 6.2: Comparison between analytical and computational approach.

problem or they provide an approximate solution to the original problem (close to the true one, but we are unable to say how much close).

A fundamental problem of the numerical methods used in optimization is their reliance on the initial conditions from which they start the search for optima. The most basic numerical optimising procedure goes though the following steps: starting from the fact that the full exploration of the domain of a function is impossible (since it would require an infinite time), an initial point is required, than a **local search** is performed, in order to find out the direction of fastest (or highest) growth. Then a new point is chosen along that direction and the process is iterated until convergence, that is, until the change in the objective function (evaluated in each of these new points) is smaller than a given threshold. It is now clear the reason why such procedures are unable to guarantee that an optimum is found (even when it actually exists), nor to characterize a given optimum as global (even though it actually is).

Remark 6.1.3 *The practical implication of this drawback is the so called **local optima trap**, which consists in the fact that these class of procedures are unable to find out global optima, but only local ones, which in turn strongly depend on the user chosen starting value.*

Example 6.1.4 *Consider the simple problem of maximizing a bimodal function by means of numerical optimization. Suppose that for a given starting value the algorithm converges to the first optimum. The procedure does not detect that there exists also another optimum, nor is able to characterize the nature of the point in output.*

■

We now turn to tackle the task of creating a function (or algorithm or procedure or routine). It may be stressed that, though every software requires the use of a specific programming language, the logic underneath the coding process is approximately the same in all cases. In light of this remark, Algorithm (6) illustrates the principal steps to be followed for creating a function, whatever programming language (or software) is used.

A couple of definitions may be useful in practice:

Definition 6.1.2 *We call **efficiency** the ability of the algorithm to accomplish a given task with the highest saving of computational time and computer memory as possible.*

Definition 6.1.3 *A **bug** is any kind of error "hidden" in the code which has the effect of making the algorithm deliver wrong (or, in a wider concept, no output). Most of the times they occur in particular cases which the programmer has not foreseen.*
*The **debugging** process is the set of procedures aimed at discovering and fixing the bugs (not the just problems they cause).*

Algorithm 6 Algorithm creation process

```
1: procedure ALGORITHM(problem)
2:     clearly understand the problem to be solved
3:     decompose problem in simpler sub-problems to be solved sequentially (or in parallel)
4:     for each sub-problem do
5:         look for already available functions
6:         if functions exist then
7:             go to next sub-problem
8:         end if
9:         choose software and functions to be used          ▷ each solves some tasks, not all
10:    end for
11:    write the code, commenting main steps                 ▷ for future or external use
12:    check that final algorithm actually solves the problem
13:    return Implement the algorithm
14: end procedure
```

Type	Definition
Numeric	complex numbers (real is a subset of complex) or arrays of complex numbers. Letters correspond to variables which are considered as numbers
String	characters. Letters and numbers are treated as text
Symbolic	unspecified function (in mathematical meaning, not an algorithm) of another variable

Table 6.3: Most common types of variables.

Notice that sometimes there may not exists any guidance from theory about the "correct" way of writing a problem solving routine, which may result in a lot of bugs and long effort (an time) for removing them.

Bugs may represent very dangerous problems, as the following example aims to point out.

Example 6.1.5 *Consider the case of running an algorithm. If the program crashes, we are sure that something has gone wrong and there is an error either in the inputs or somewhere in the main part of the code. By contrast, if it doesn't and an output is actually produced, we cannot be sure that it is what the algorithm was supposed to deliver, since it is possible that the code contains an error that does not impede to produce an output. For example, if we mistakenly write the formula for the solution of second order equation by writing the wrong signs (or by multiplying for a constant) no error message is return and an answer is produced, though totally wrong.*

■

The concept of variable is the core of each programming language: they are denoted by single letters of groups of letters and can be considered the most basic building block of each algorithm. Any language classifies variables in many categories and attaches to each of them specific properties and usage; Table (6.3) provides an overview of the most commonly used classes of variables. Notice that both MATLAB and R treat any letter as a numeric (or string) variable, while additional toolboxes/packages are required to deal with symbolic calculus.

Example 6.1.6 *Consider the following cases.*
***Numeric**: $x = [-2\ 1\ y\ 5]$, where y is another number, is a numeric vector.*

***String**: x =['-2' '1' y '5'], where y is a string, is a string vector.*
Note that the content is "the same", but it is treated in different ways according to the type.
***Symbolic**: $x(t)$, where it has not been specified neither the range of values of t nor the explicit expression of the function. However differentiation in general yields: $\mathrm{d}x(t)/\mathrm{d}t = x'(t)$. Get a value after specifying the function and the t.*

■

6.2 Computational Issues

When dealing with computational softwares there are some non trivial issues to always keep in mind:

- uncertainty: we cannot be completely sure about the exactness of the results;
- unexpected cases: we should foresee (almost) all possible occurrences while coding, but there is (almost) always some specific event which is not foreseen that may invalidate the output of the algorithm;
- calibration: when exploring an unknown problem and some conditions are indeed required (i.e. initial values, time span, limits for the variable and so on), it may be hard to make a "good guess". The algorithm may perform bad accordingly and it may take a long time for the programmer to choose "suitable values";
- symbolic calculus: it is still a hard task both for MATLAB and R (especially for the latter). For this purpose, consider a different software, like Maple® (Maplesoft webpage) or Mathematica® (Wolfram webpage).

Framework	Goal	Problem	Tentative solution
Static Optimization	Find global optimum	Local optima trap: only local optima may be obtained, given initial value	Use multiple initial values
Plotting	Get interpretable plot	"strange" plot, potentially misleading conclusions. Due to absence of guidance in choosing graph sizes	Change sizes of the plot, initial values

Table 6.4: Summary of computational issues and potential solutions.

As an example, in Table (6.4) are listed a couple of computational problems whose cause may require some time and trials to be discovered, though the solution is in reality straightforward. Cases like these are quite frequent and likely to occur in practical work and reflect some unforeseen cases.

6.2.1 MATLAB and R references

In this subsection we give a list of useful online and paper references for the two softwares we will use later: MATLAB and R. This set of resources is far from being exhaustive, since the published material (offial as well as unofficial) in both cases is steadily growing and making a representative list of them is a hard task.

We start from MATLAB, which is a software released under licence by MathWorks (MathWorks website): if the required licence has been purchased, it is possible to download many official toolboxes (and the attached documentation) from the same website and to require the help of the staff in case of troubles with the correct functioning of MATLAB itself.

An important built-in function is the "`help`", which allows to get information about other built-in functions (usually, a description of the tasks it solves, its inputs and outputs and some examples).

Another website hosted by the same Company is the MATLAB Central, which contains: free user-defined functions; a forum where to ask other users for help; basic and intermediate tutorials on the use of MATLAB.

As far as books and organic technical texts are concerned, some noteworthy references include: MathWorks documentation for what regards manuals and tutorials; [2], [8] for applications in finance and economics; MathWorks Support for a webpage with a comprehensive list of further references.

Another very widespread software for statistical and general computing is R, which is an open-source software freely downloadable from the CRAN website. On the same webpage there are thousands of downloadable packages, and for each of them it is possible to download also the documentation, containing instructions of usage and practical examples, as well as the source code, for modifying or adapt the functions of the package to our personal interests. Again, the built-in "`help`" function allows to get information about other built-in functions.

Another quite useful resource is the Stackoverflow forum.

Additional aid is provided by a lot a references among which we point out the manuals available at CRAN documentation; the online learning tools on RStudio Online learning; the books by [9], [4] and [13] for applications in finance end economics; the webpage R books which contains many more references.

6.3 Static Optimization

In this section we show how to use MATLAB and R to solve some simple static optimization problems. Since the computer most often will perform numerical rather than analytical methods for solving a given problem, it is of fundamental importance to specify:

- method of numerical optimization to be used (optional, softwares have default settings)
- initial value

Remark 6.3.1 *The choice of the initial value is fundamental since any numerical method suffers (though with different extents) from the* ***initial conditions dependence****, that is it may give different output for different initial conditions. This is the case in presence of multiple optima, from which the alternative name of* ***local optima trap****.*

The main additional tools required in the routines described below for the static optimization case are the "`Optimization`" toolbox, as MATLAB is concerned and the "`nloptr`" package for R.

6.3.1 Unconstrained Optimization

A simple unconstrained optimization can be performed following the guidelines of Algorithm (7) below.

Algorithm 7 Naïve Numerical solution of Unconstrained Optimization problem

procedure OPTIMIZATION(f,$\mathbf{x_1}$,ϵ)
  $i \leftarrow 2$
  $v_i \leftarrow f(\mathbf{x_1})$
  while $|v_{i+1} - v_i| \geq \epsilon$ **do**
    propose $\mathbf{x}^*$ according to the chosen numerical method
    $\mathbf{x_{i+1}} \leftarrow \mathbf{x}^*$
    $v_{i+1} \leftarrow f(\mathbf{x_{i+1}})$
    $i \leftarrow i + 1$
  end while
  return Solution: $\mathbf{x_i}$
end procedure

The functions to be used are "`fminsearch`" in MATLAB and "`optim`" in R. The default optimization problem solved by both softwares is minimization, however recall the results in Table (**??**) in order to solve maximization problems by simply re-stating the problem in a different way. The output is a vector or a list with: the location of the optimum point and the corresponding optimal value of the function, plus other information depending on the options specified and the software used (for example, the numerical method applied, the number of iterations of the algorithm).

Remark 6.3.2 *As previously stressed, in any case these algorithms are able to yield only local optima (if any), while if there are many of them they are able to find them all. The output is crucially driven by the specified initial value.*

Example 6.3.1 *Solve the problem:*

$$\max_{(x_1,x_2)} \quad 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{6.1}$$

using alternatively MATLAB or R.

- *MATLAB code for unconstrained optimization:*

```
% objective function to MINIMIZE
fun = @(x) 100*(x(2)-x(1)^2)^2+(1-x(1))^2;

[x,fval,exitflag,output] = fminsearch(fun,[-1.2, 1])
```

- *R code for unconstrained optimization:*

```
# objective function to MINIMIZE
fun = function(x){
    x1 = x[1]
    x2 = x[2]
    100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}
# Gradient (can be omitted)
grad = function(x){
    x1 = x[1]
    x2 = x[2]
    c(-400 * x1 * (x2 - x1 * x1) - 2 * (1 - x1),
       200 *       (x2 - x1 * x1))
}

# optimization
res= optim(par= c(-1.2,1), fun, grad, method= "BFGS", hessian= TRUE)
print(res)

# only the Hessian
optimHess(res$par, fun, grad)
```

■

6.3.2 Constrained Optimization

A naïve version of a numerical procedure for finding a constrained optimum is obtained by a slight modification of Algorithm (7) and is described in Algorithm (8).

Algorithm 8 Naïve Numerical solution of Constrained Optimization problem

1: **procedure** OPTIMIZATION(f,g,$\mathbf{b}$,$\mathbf{x_1}$,ϵ)
2: $i \leftarrow 2$
3: $v_i \leftarrow f(\mathbf{x_1})$
4: **while** $|v_{i+1} - v_i| \geq \epsilon$ **do**
5: propose $\mathbf{x}^*$ according to the chosen numerical method
6: **if** $\mathbf{x}^*$ satisfies the constraints g **then**
7: $\mathbf{x_{i+1}} \leftarrow \mathbf{x}^*$
8: $v_{i+1} \leftarrow f(\mathbf{x_{i+1}})$
9: $i \leftarrow i + 1$
10: **else**
11: go back to Step 5
12: **end if**
13: **end while**
14: **return** Solution: $\mathbf{x_i}$
15: **end procedure**

The functions to be used in the equality constraints case are "`fmincon`" in MATLAB and "`nloptr`" in R, which, as in the previous case, by default solve a minimization problem.

The main inputs of "`fmincon`" are: the objective function, a matrix and a vector of coefficients (`Aeq` and `beq`, respectively) which are used to represent linear constraints (in vector form `Aeq*x=beq`), a function containing nonlinear constraints (`ceq`) and initial values (`x0`).

On the other hand, the main inputs of "`nloptr`" are: the objective function (`eval_f`), a function containing linear constraints, another one with nonlinear constraints (`eval_g_eq`) and initial values (`x0`).

The output is a vector or a list with: the location of the optimum point and the corresponding optimal value of the function, plus other information depending on the options specified and the software used (for example, the numerical method applied, the number of iterations of the algorithm).

Example 6.3.2 *Use MATLAB to solve the problem:*

$$\begin{cases} \min\limits_{(x_1,x_2)} & 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \\ s.t. & \\ & 2x_1 + x_2 = 1 \end{cases} \tag{6.2}$$

while use R for solving:

$$\begin{cases} \min\limits_{(x_1,x_2,x_3,x_4)} & x_1 x_4 (x_1 + x_2 + x_3) + x_3 \\ s.t. & \\ & x_1^2 + x_2^2 + x_3^2 + x_4^2 = 40 \end{cases} \tag{6.3}$$

- *MATLAB code for constrained optimization with equality constraints:*

```
% min 100*(x(2)-x(1)^2)^2 + (1-x(1))^2
% s.t.
% 2x(1) + x(2) = 1

% objective function
fun = @(x) 100*(x(2)-x(1)^2)^2 + (1-x(1))^2;
% initial value + other parameters for the solver
x0 = [0.5,0];
```

```
A=[];
b=[];
Aeq = [2,1];
beq = 1;

sol = fmincon(fun,x0,A,b,Aeq,beq)
```

- *R code for constrained optimization with equality constraints:*

```
# min x1*x4*(x1 + x2 + x3) + x3
# s.t.
# x1^2 + x2^2 + x3^2 + x4^2 = 40

require(nloptr)
# objective function
eval_f = function( x ){return( list(
        "objective"= x[1]*x[4]*(x[1]+x[2]+x[3])+x[3],
        "gradient"= c( x[1]*x[4] +x[4]*(x[1] +x[2] +x[3]),
        x[1]*x[4],
        x[1]*x[4] +1.0,
        x[1]*(x[1] +x[2] +x[3]) ) ) )}
# constraint functions
eval_g_eq = function( x ){constr= c(x[1]^2+x[2]^2+x[3]^2+x[4]^2-40 )
        grad= c( 2.0*x[1],2.0*x[2],2.0*x[3],2.0*x[4] )
        return( list("constraints"=constr,"jacobian"=grad ))}
# initial values
x0 = c(1, 5, 5, 1)
# options
local_opts= list( "algorithm"= "NLOPT_LD_MMA", "xtol_rel"= 1.0e-7 )
opts= list( "algorithm"= "NLOPT_LD_AUGLAG", "xtol_rel"= 1.0e-7,
        "maxeval"= 1000,"local_opts"= local_opts )

res = nloptr( x0=x0, eval_f=eval_f, eval_g_eq=eval_g_eq, opts=opts)
print(res)
```

■

The functions to be used in the inequality constraints case are "`fmincon`" in MATLAB and "`nloptr`" in R, which, as in the previous case, by default solve a minimization problem.

The main inputs of "`fmincon`" are: the objective function, a matrix and a vector of coefficients (`A` and `b`, respectively) which are used to represent linear constraints (in vector form `A*x`$\leq$`b`), a function containing nonlinear constraints (`c`) and initial values (`x0`).

On the other hand, the main inputs of "`nloptr`" are: the objective function (`eval_f`), a function containing linear constraints, another one with nonlinear constraints (`eval_g_ineq`) and initial values (`x0`).

The output is a vector or a list with: the location of the optimum point and the corresponding optimal value of the function, plus other information depending on the options specified and the software used (for example, the numerical method applied, the number of iterations of the algorithm).

Example 6.3.3 *Use MATLAB to solve:*

$$\begin{cases} \min\limits_{(x_1,x_2)} & 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \\ s.t. & \\ & (x_1 - \frac{1}{3})^2 + (x_2 - \frac{1}{3})^2 - (\frac{1}{3})^2 \leq 0 \end{cases} \tag{6.4}$$

and use R for solving:

$$\begin{cases} \min\limits_{(x_1,x_2)} & \sqrt{x_2} \\ s.t. & \\ & x_2 \geq (2x_1)^3 \\ & x_2 \geq (-x_1+1)^3 \end{cases} \tag{6.5}$$

- *MATLAB code for constrained optimization with inequality constraints:*

```
% objective function to MINIMIZE
fun = @(x) 100*(x(2)-x(1)^2)^2+(1-x(1))^2;
% function with nonlinear (inequality) constraints
function [c,ceq] = circlecon(x)
c = (x(1)-1/3)^2 + (x(2)-1/3)^2 - (1/3)^2;
ceq = [];

% initial value + no linear nor equality constraints
x0 = [1/4 1/4];
A = [];
b = [];
Aeq = [];
beq = [];
% no lower/upper bounds of variable: it is 'free'
lb=[]; ub=[];

nonlcon = @circlecon;

sol = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
```

- *R code for constrained optimization with inequality constraints:*

```
# min sqrt( x2 )
# s.t. x2 >= 0
# x2 >= ( a1*x1 + b1 )^3   # where a1 = 2, b1 = 0,
# x2 >= ( a2*x1 + b2 )^3   # where a2 = -1, b2 = 1

require(nloptr)
# objective function
eval_f0 = function( x, a, b ){return( sqrt(x[2]) )}
# constraint function
eval_g0 = function( x, a, b ){return( (a*x[1] + b)^3 - x[2] )}
# gradient of objective function
eval_grad_f0 = function( x, a, b ){return( c( 0, .5/sqrt(x[2]) ) )}
# jacobian of constraint
eval_jac_g0 = function( x, a, b ){
    return( rbind( c( 3*a[1]*(a[1]*x[1] + b[1])^2, -1.0 ),
    c( 3*a[2]*(a[2]*x[1] + b[2])^2, -1.0 ) ) )}
# define parameters
a = c(2,-1); b = c(0, 1)

res0 = nloptr( x0=c(1.234,5.678), a= a, b= b, eval_f= eval_f0,
        eval_grad_f= eval_grad_f0, lb= c(-Inf,0), ub= c(Inf,Inf),
        eval_g_ineq= eval_g0, eval_jac_g_ineq= eval_jac_g0,
        opts= list("algorithm"="NLOPT_LD_MMA") )
print(res0)
```

■

6.4 Differential Equations

In the class of differential equations and the calculus of variations problems it is necessary to undertake integration as well as differentiation of functions, the latter is the main reason for which the main additional tools required in the routines described below are the "`Symbolic Math`" toolbox, as MATLAB is concerned and the "`deSolve`" and "`PhaseR`" packages for R.

6.4.1 General Case

The functions to be used in the first order case are "`ode23`" (or "`ode45`") in MATLAB and "`ode`" in R.

The main inputs of "`ode23`" are: a function which contains the functional (`odefun`), a set of boundary conditions (`y0`) and a time span over which the solution will be evaluated and plotted (`tspan`). The output is a vector with the optimal values of the solution function at each point within the specified time span, together with a plot of the solution.

On the other hand, the main inputs of "`ode`" are: a function containing the functional (`func`), a set of boundary conditions (`y`), a time span over which the solution will be evaluated and plotted (`times`) and eventually a vector of parameters to be passed to lower level functions (i.e. the functional). The output is a vector with the optimal values of the solution function at each point within the specified time span, together with a plot of the solution.

Example 6.4.1 *Use MATLAB to obtain the solution of:*

$$\dot{x} = -(a^2 - a - 3)x + 3\sin\left(b - \frac{1}{4}\right) \tag{6.6}$$

with initial value $x(1) = 1$ *and a grid of values* $a \in [0,5]$, $b \in [1,6]$. *Then use R to solve the Lorenz system:*

$$\begin{cases} \dot{x} = -\frac{8}{3}x + yz \\ \dot{y} = -10(y - z) \\ \dot{z} = -xy - 28y - z \\ \textit{initial conditions:} \\ x(1) = 1 \\ y(1) = 1 \\ z(1) = 1 \end{cases} \tag{6.7}$$

- *MATLAB code for first order differential equation:*

```
at= @(a) a.^2 -a -3;
bt= @(b) 3*sin(b -0.25);
tspan=[1 5]; ic=1;

function [T,X]=Differential_first_M(at,bt,tspan,ic)
    % general linear ODE: x' +a(t)x = b(t)
    % Inputs: at,bt function handles with variables a,b
    %         tspan vector 2x1 of time limits for evaluating solution
    %         ic scalar of initial value for particular solution
    % Outputs: analytical solution + plot

    % generate sequence of a(t) and b(t)
    a = linspace(0,5, 25);
    at = feval(at,a);
    b = linspace(1,6, 25);
```

```
    bt = feval(bt,b);
    % 'compose' the ODE as function
    function xdot = firstode(t,x,a,at,b,bt)
        % create finer grid for a(t) and b(t) via interpolation at t
        at = interp1(a,at,t);
        bt = interp1(b,bt,t);
        % Evaluate ODE at time t
        xdot = -at.*x + bt;
    end
    % call fucntion 'firstode' and solve
    [T, X] = ode45(@(t,x) firstode(t,x,a,at,b,bt),tspan,ic);
    plot(T,X); xlabel('time','Interpreter','latex');
    ylabel('x(t)','Interpreter','latex');
    title(sprintf('Solution of linear differential equation, given...
            initial condition $x(%d)=%d$', tspan(1),ic),'Interpreter','latex');
end
```

- *R code for first order differential equation:*

```
require(deSolve)
param = c(a=-8/3, b=-10, c=28) # parameters
state = c(X=1, Y=1, Z=1) # state vars + initial values
# function describing the derivative of the state variables
Lorenz = function(t, state, parameters){
  # transform vectors into lists, so we can 'call' its elements
  # by their name instead of their position in the vector
  with( as.list( c(state, parameters) ), {
    # define the rate of change
    dX = a*X + Y*Z
    dY = b * (Y-Z)
    dZ = -X*Y + c*Y - Z
    list(c(dX, dY, dZ)) # NOTE: order must be the same as input
  }) # end with(as.list ...
}
time = seq(0, 100, by=0.01)   #time: 100 days, steps of 0.01 days
# SOLUTION and PLOTS - 4 graphs
out = ode(y= state, times= time, func= Lorenz, parms= param)
par(oma = c(0, 0, 3, 0))
plot(out, xlab = "time", ylab = "-")
plot(out[, "X"], out[, "Z"], pch = ".")
mtext(outer = TRUE, side = 3, "Lorenz model", cex =1.5)
```

■

The functions to be used in the second order case in MATLAB are the same ("`ode23`" or "`ode45`") if we tackle the problem by rewriting the second order equation as system of first order equations, while we need to use the functions "`dsolve`" and "`eval(vectorize(.))`" contained in the "`Symbolic Math`" toolbox if we wish to use symbolic calculus to find a solution.

The main inputs of the symbolic approach are: the second order equation and the initial values (both given in input as strings), furthermore a grid of values is required for plotting. The output is a vector with the optimal values of the solution function at each point within the specified grid, together with a plot of the solution.

By contrast, in R we can still use the previously described function "`ode`", with the same inputs and outputs.

Example 6.4.2 *Use MATLAB for obtaining the solution to:*

$$\ddot{y} - 5\dot{y} + 7y = \sin(x)x^2 \tag{6.8}$$

where $\dot{y} = \frac{\partial y(x)}{\partial x}$, *with initial conditions* $y(0) = 0$, $\dot{y}(0) = 1$. *Use R for getting the solution of:*

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0 \tag{6.9}$$

with $\mu = 1000$ *and initial conditions* $x(0) = 2$, $\dot{x}(0) = 0$.

- *MATLAB code for second order differential equations:*

```
eqn= 'D2y -5*Dy +7*y = sin(x).*(x.^2)';
ic= ['y(0)=0' 'Dy(0)=1'];

function [T,X]=Differential_second_M(eqn,ic)
    % linear ODE const coeff: x'' +a1x' +a2x = b(t)
    % Inputs: eqn ODE as string. Use D2y Dy y as variables
    %         ic initial values as string. Use y(t0)=y0, Dy(t0)=Dy0
    % Outputs: analytical solution + plot

    % NOTES: 1) y not suited for array operations --> vectorize()
    %           transforms into strings
    %        2) result is symbolic object --> eval() evaluates strings

    % grid of values of time for plotting the solution
    x = linspace(0,1, 20);
    % use Symbolic calculus
    syms y
    Dy = diff(y); D2y = diff(y,2);
    % dsolve requires to specify the variable, otherwise takes 't'
    [T,X] = dsolve(eqn,ic,'x');
    z = eval(vectorize(X));
    plot(x,z); xlabel('time (x)','Interpreter','latex');
    ylabel('$y(t)$','Interpreter','latex');
    title(strcat('Solution of [',eqn,'], given initial values [',...
        ic,']'),'Interpreter','latex');
end
```

- *R code for second order differential equations:*

```
# x'' -mu(1 - x^2)x' +x = 0
require(deSolve)
# split in system of first order ODEs
# x'=y2
# x=y1
# hence: y2=y1'

vdpol = function (t, y, mu){
  list(c(
    y[2],
    mu*(1 - y[1]^2) * y[2] - y[1]
  ))
}
yini = c(y1 = 2, y2 = 0)

out = ode(y= yini, func= vdpol, times= 0:3000, parms= 1000)
plot(out, type= "l", which= "y1", ylab="x(t)",xlab="time",
     main="Solution of [x''-mu(1-x^2)x'+x=0],
     given initial conditions")
```

■

6.4.2 Autonomous Case

For dealing with the one dimensional autonomous case in MATLAB there are no predefined routines, hence the author has developed a new function[5].

The main inputs are: the equation (in input as a function handle), an interval for the time and another one for the solution function (required for plotting) and a vector of initial conditions. The output consists of the optimal values of the solution function at each point within the time interval, together with a plot of the solution in the specified interval (phase plane). In addition the velocity plot is drawn together with an additional plot drawing the trajectory of the solution for each initial condition.

On the other hand, in R we use the functions in the packages "`deSolve`" and "`PhaseR`", which take as main inputs a function containing the derivatives (i.e. the original equation), a time interval and an interval for the solution function (required for plotting) and a vector of initial conditions. The output consists in the phase plane and a velocity plot containing also all the trajectories of the solution for each of the specified initial conditions.

Example 6.4.3 *Plot the phase plane of the following differential equation using both MATLAB and R:*

$$\dot{x} = x^3 - 3x^2 - x + 1. \tag{6.10}$$

- *MATLAB code for phase plane of first order autonomous differential equations:*

```
eq= @(t,x) x.^3 -3.*x.^2 -x +1;

function ind_phase(plim,tlim,xlim,points,eq,init)
  % Inputs: limits for function plot; limits for x and time;
  %         points for grid spacing; equation (as function handle);
  %         initial value for solution evaluation
  % Output: Function plot; Velocity plane + solutions;
  %         Separate solutions according to initial values

  % plot the function x'=f(x)
  figure(1)
  w= linspace(plim(1),plim(2),100);  y= feval(eq,0,w);
  plot(w,y,[0 0],[min(y) max(y)],'black',[min(w) max(w)],[0 0],'black');
  xlabel('$x(t)$','Interpreter','latex');
  ylabel('$\dot{x}(t)$','Interpreter','latex');
  title(strcat('Plot of the function:  $',func2str(eq),'$'),...
      'Interpreter','latex')

  % Velocity Plane
  figure(2)
  a= linspace(tlim(1),tlim(2),points); b= linspace(xlim(1),xlim(2),points);
  [t,x]=meshgrid(a,b);
  dx= feval(eq,0,x);
  dt= ones(size(dx));
  dxu= dx./sqrt(dt.^2+dx.^2);
  dtu= dt./sqrt(dt.^2+dx.^2);
  nplots= length(init)+1;  d= round(nplots/2);
  subplot(2,d,1);
  quiver(t,x,dtu,dxu);xlabel('time','interpreter','latex');
  ylabel('$x(t)$','Interpreter','latex');
  title('Velocity Plane for $\dot{x}(t)=f(x(t))$',...
      'Interpreter','latex');

 % add solutions paths for different initial values
```

[5]The script of this and all other functions are available upon request to the author.

```
 hold on
 for i=1:length(init)
    [T,Y] = ode45(eq,tlim,init(i));
    subplot(2,d,1);
    plot(T,Y,'r-');
    subplot(2,d,i+1);
    plot(T,Y); xlabel('time','interpreter','latex');
    ylabel('$x(t)$','Interpreter','latex');
    title(sprintf('Solution for initial value:  $x(%.2f)=%.2f$',...
    tlim(1),init(i)),'Interpreter','latex');
  end
  hold off
end
```

- *R code for phase plane of first order autonomous differential equations:*

```
require(deSolve)
require(phaseR)
ex1 = function(t, y, parameters){
  a=parameters[1]
  b=parameters[2]
  dy = a*y^3 +b*y^2 -y +1
  list(dy)
}
# Flow Field #
ex1.flowField = flowField(ex1, x.lim=c(0, 5), y.lim=c(-3, 5),
            parameters=c(1,-3), system = "one.dim", points=21,
              add=FALSE, xlab="t", ylab="y(t)")
ex1.nullclines = nullclines(ex1, x.lim=c(0, 5), y.lim=c(-3, 5),
            parameters=c(1,-3), system = "one.dim", lwd=1.5)
ex1.trajectory = trajectory(ex1, y0=c(-2.5,-0.5,1.5,4), t.end=5,
            parameters=c(1,-3), system="one.dim",
            colour=rep("black", 4))
# Phase Diagram #
ex1.PD = phasePortrait(ex1, y.lim=c(-1.5,3.5), parameters=c(1,-3),
            points=15, xlab="y(t)", ylab="f[y(t)]")
abline(v=0)
```

■

For dealing with the two dimensional (system) autonomous case in MATLAB there are no predefined routines, hence the author has crated a new function, which deals only with the 2×2 case.

The main inputs are: two strings containing the right hand side of the equations of the system, a vector of initial values and a time span for evaluating and plotting the solutions. The output consists of the optimal values of the solution function at each point within the time interval, together with the phase plane with the nullclines and another phase plane with the nullclines and the trajectories of the solution functions starting from the given initial conditions. In addition a plot is drawn reporting the trajectory of the solution for each initial condition.

On the other hand, in R we can still use the functions in the packages "`deSolve`" and "`PhaseR`", which in this case take as main inputs a function including the equations of the system, a time interval and a vector of initial conditions. The output consists in the phase plane in which are drawn all the nullclines and the trajectories of the solution functions for each initial condition.

Example 6.4.4 *Plot the phase plane for the following system of first order differential equations, using MATLAB:*

$$\begin{cases} \dot{x} = -3x - 2y \\ \dot{y} = 2x - 5y \end{cases} \tag{6.11}$$

and do the same in R for:

$$\begin{cases} \dot{x} = -3x + y \\ \dot{y} = -2x - y \end{cases} \tag{6.12}$$

- *MATLAB code for autonomous systems of two first order differential equations:*

```
xdot= '-3.*x 2.*y';
ydot= '2.*x -5.*y';

function sol=Differential_sys_M(xdot,ydot,veclim,sep,vp,x0,y0,tspan)
    % Inputs: xdot,ydot equations of the 2x2 system as strings
    %         veclim vector 4x1 of x,y limits for velocity plot
    %         sep=1 for separate plots in output
    %         x0,y0 cell arrays of initial values for solutions
    %               entries as strings with form: 'x(t0) == x0'
    %         tspan vector 2x1 of intial and final time for solution
    % Output: 1] Velocity Plot (optional)
    %         2] Phase Plane = Velocity Plot + nullclines;
    %         3] Phase Plane + trajectories
    %         4] Solutions plot
    %         5] Solutions as matrix of symbolic fucntions
    %************************************************************
    % IMPORTANT: xdot,ydot -> use 'x' and 'y' as variables
    %            x0,y0 -> maximum length=4
    %************************************************************
    % NOTE: function: string for 'vectfield', symbolic for 'ezplot'

    % check length of initial values
    if length(x0)>4 || length(y0)>4 || length(x0)~=length(y0)
        h=errordlg('Wrong input. Max number of initial conditions is
        four and length(x0) mut be equal to length(y0).','Wrong input');
        return
    end

    % symbolic solution
    syms sol
    % form the anonymous function from string
    f = str2func(strcat('@(t,x,y)[',xdot,',',ydot,']'));
    % 1) VELOCITY PLOT
    figure(1)
    if sep ~= 1
        if vp==1
            subplot(2,2,1)
            vectfield(f, veclim(1):.5:veclim(2), veclim(3):.5:veclim(4))
        end
        subplot(2,1,1)
    end
    % 2) PHASE PLANE
    % plot velocity plot (only arrows)
    if sep ~= 1
        if vp==1
            subplot(2,2,2)
        end
        subplot(1,2,1)
    else
        figure(2)
    end
    hold on
    vectfield(f, veclim(1):.5:veclim(2), veclim(3):.5:veclim(4));
    % symbolic for evaluation of function
    % NOTE: NOT use x(t) otherwise error from ezplot
    syms x y
```

```
    funcx = eval(xdot);
    funcy = eval(ydot);
    % ezplot for plotting functions (=curve in R)
    hx = ezplot(funcx,[veclim(1),veclim(2),veclim(3),veclim(4)]);
    set(hx,'color','red');
    hy = ezplot(funcy,[veclim(1),veclim(2),veclim(3),veclim(4)]);
    set(hy,'color','blue');
    xlabel('$x(t)$','Interpreter','latex'); ylabel('$y(t)$','Interpreter','latex');
    title('Phase plane of the system','Interpreter','latex');
    l = legend([hx hy],'$\dot{x}(t)=0$','$\dot{y}(t)=0$','best');
    set(l,'Interpreter','latex','fontsize', 8);
    hold off

    % 3) SOLUTIONS
    if length(x0)==1
        r=1; c=length(x0);
    elseif length(x0)==2
        r=1; c=length(x0);
    elseif length(x0)>2
        r=2; c=length(x0);
    end
    %hold on
    % need symbolic variabes as functions of t
    syms x(t) y(t)
    funcxt = eval(xdot); funcyt = eval(ydot);
    eqn1 = diff(x) == funcxt; eqn2 = diff(y) == funcyt;
    % solve the system for each initial value
    figure(2)
    for i=1:length(x0)
        [xs,ys] = dsolve(eqn1, eqn2, x0(i), y0(i));
        % store the solution
        sol(i,1)=simplify(xs); sol(i,2)=simplify(ys);
        % plots
        subplot(r,c,2*i-1)
        hx = ezplot(xs,[tspan(1),tspan(2)]); set(hx,'Color',[0.4 0.7 1]);
        xlabel('time','Interpreter','latex'); ylabel('$x(t)$','Interpreter','latex');
        title(sprintf('Solution of the system: case %s,%s',x0{i},y0{i}),...
                'Interpreter','latex');
        l = legend(hx,'$x(t)$','best'); set(l,'Interpreter','latex','fontsize', 8);
        subplot(r,c,2*i)
        hy = ezplot(ys,[tspan(1),tspan(2)]); set(hy,'Color',[1 0.5 0]);
        xlabel('time','Interpreter','latex'); ylabel('$y(t)$','Interpreter','latex');
        title(sprintf('Solution of the system: case %s,%s',x0{i},y0{i}),...
                'Interpreter','latex');
        l = legend(hy,'$y(t)$','best'); set(l,'Interpreter','latex','fontsize', 8);
    end
    hold off

    % 4) PHASE PLANE WITH SOLUTIONS' TRAJECTORIES
    if sep ~= 1
        figure(1)
        if vp==1
            subplot(2,2,3)
        end
        subplot(1,2,2)
    else
        figure(3)
    end
    hold on
    % redo the same for plottig Phase Plane
    vectfield(f, veclim(1):.5:veclim(2), veclim(3):.5:veclim(4));
    syms x y
    funcx = eval(xdot); funcy = eval(ydot);
```

```
    hx = ezplot(funcx,[veclim(1),veclim(2),veclim(3),veclim(4)]);
    set(hx,'color','red');
    hy = ezplot(funcy,[veclim(1),veclim(2),veclim(3),veclim(4)]);
    set(hy,'color','blue');
    xlabel('$x(t)$','Interpreter','latex');
    ylabel('$y(t)$','Interpreter','latex');
    title('Phase plane with trajectories','Interpreter','latex');
    % define time sequence for evaluating solutions
    t = tspan(1):(tspan(2)-tspan(1))/30:tspan(2);
    % create and plot trajectories for each initial condition
    for i=1:length(x0)
        [xs,ys] = dsolve(eqn1, eqn2, x0(i), y0(i));
        traj = [subs(xs); subs(ys)];
        plot(traj(1,:),traj(2,:),'m-<','MarkerFaceColor','m');
        axis([veclim(1) veclim(2) veclim(3) veclim(4)]);
    end
    % evaluate the solutions at the time
    traj = [subs(xs); subs(ys)];
    % plot trajectories of the solutions
    plot(traj(1,:),traj(2,:),'m-<','MarkerFaceColor','m');
    axis([veclim(1) veclim(2) veclim(3) veclim(4)]);
    hold off
end
```

- *R code for autonomous systems of two first order differential equations:*

```
require(deSolve)
require(phaseR)
sys2 = function(t,y,parameters){
  x=y[1]
  y=y[2]
  a=parameters[1]
  b=parameters[2]
  c=parameters[3]
  d=parameters[3]
  dy=numeric(2)
  dy[1]= a*x +b*y
  dy[2]= c*x +d*y
  list(dy)
}
sys2.flowField = flowField(sys2, x.lim=c(-4,4), y.lim=c(-6,6),
                 param=c(-3,1,-2,-1), points=21,add=F,
                 xlab="x(t)",ylab="y(t)")
sys2.nullclines = nullclines(sys2, x.lim=c(-4,4),y.lim=c(-6,6),
                 param=c(-3,1,-2,-1), points=500,lwd=2.2)
abline(v=0,h=0,lwd=0.8)
y0 = matrix(c(2,-3,-1.5,-1,-0.5,2,1.5,1), ncol=2, nrow=4, byrow=T)
sys2.trajectory = trajectory(sys2, y0 = y0, t.end=10,
                 param=c(-3,1,-2,-1), colour=rep("black", 3))
```

■

6.5 Difference Equations

In the context of difference equations we are going to present the computational part only for the first order autonomous case, namely the cobweb diagram. The procedure is quite simple and is described in Algorithm (9).

Algorithm 9 Cobweb diagram

procedure COBWEB(f,$\mathbf{x_1}$,n,$\mathcal{I}$)
  plot f over the interval $\mathcal{I}$
  add the bisector of the I-III quadrant
  for $i = 1,\ldots,k$ **do**
    $x_{i+1} \leftarrow f(x_i)$
    draw horizontal segment between $(x_i,f(x_i))$ and $(x_{i+1},f(x_i))$
    draw vertical segment between $(x_{i+1},f(x_i))$ and $(x_{i+1},f(x_{i+1}))$
  end for
  return Plot: cobweb diagram
end procedure

6.5.1 Autonomous Case

In MATLAB as well as in R there are no built-in functions nor routines in advanced toolboxes/packages that are devoted to the creation of a cobweb diagram, hence the author has developed two procedures for solving this problem[6].

In both cases the main inputs are: the original function, a vector of initial conditions, a scalar representing the maximum number of cobweb segments to be drawn and an interval of the independent variable over which the function will be drawn. The output consists the plot of function together with the cobweb for each initial point.

Example 6.5.1 *Use MATLAB for drawing the cobweb diagram of:*

$$x_{t+1} = x_t^3 - 2x^2 \tag{6.13}$$

for $x_0 = -0.21$*, then do the same in R for:*

$$x_{t+1} = x_t^2 \tag{6.14}$$

and starting points $x_0 = 0.5$ *and* $x_0 = 1.05$*.*

- *MATLAB code for phase plane of first order autonomous difference equations:*

```
f= @(x) x.^3 -2*x.^2;
x0= -0.21;

function Difference_M(f,x0,n,tsol)
    % Inputs: f (autonomous) function handle
    %         x0 initial value
    %         n high values = high precision of the plot
    %         tsol time length of solution evaluation
    % Outputs: Phase Plane + Solution for intial tsol times

    x=zeros(n+1,1); t=zeros(n+1,1);
    x(1)=x0; tt(1)=0;
    for i=1:n % create sequence of (t,xt)
        t(i)=i-1;
        x(i+1)=feval(f,x(i));
    end
    % 1) PLOT function and bisector
    t(n+1)=n; nn=100; del=1./nn; xstart=-0.5;
    yy=zeros(nn+1,1); xx=zeros(nn+1,1); lin=zeros(nn+1,1);
```

[6]The source code for the R function is given by professor Paolo Pellizzari from Ca' Foscari University of Venice.

```
    for i=1:nn+1
        xx(i)=xstart+(i-1)*del;
        lin(i)=xx(i);
        yy(i)=feval(f,xx(i));
    end
    figure(1)
    subplot(1,2,1)
    hold on
    plot(xx,lin,'r',[min(xx) max(xx)],[0 0],'k',...
        [0 0],[min(min(lin),min(yy)) max(max(lin),max(yy))],'k');
    plot(xx,yy,'k','linewidth',1.5);
    axis([min(xx) max(xx) min(min(lin),min(yy)) max(max(lin),max(yy))]);
    xlabel('$x_t$','Interpreter','latex');
    ylabel('$x_{t+1}$','Interpreter','latex');
    title(strcat('Phase diagram of  $',func2str(f),'$'),'Interpreter','latex');
    % 2) COBWEB
    xc=zeros(24,1); yc=zeros(24,1);
    xc(1)=x0; yc(1)=0; % starting value
    for j=3:tsol;
        jj=2*j-4;
        xc(jj)=xc(jj-1); %vertical line
        yc(jj)=feval(f,xc(jj)); %f(x)
        xc(jj+1)=yc(jj); %horizontal line
        yc(jj+1)=yc(jj);
    end
    plot(xc,yc,'b');
    h= legend('$x_{t+1}=x_t$','$x_{t+1}=f(x_t)$','cobweb','Location','best');
    set(h,'Interpreter','latex');
    hold off
    % 3) SOLUTION
    subplot(1,2,2)
    plot(1:1:tsol,x(1:tsol),'b',1:1:tsol,x(1:tsol),'ro');
    axis([1 tsol min(x)-0.2 max(x)+0.2]);
    xlabel('time','Interpreter','latex');
    ylabel('$x_t$','Interpreter','latex');
    title(strcat('Solution of  $',func2str(f),'$'),'Interpreter','latex');
end
```

- *R code for phase plane of first order autonomous difference equations:*

```
## Phase Diagram for first order autonomous difference equation ##
f=function(x){ x^2 }
#initial value
x0 = 0.5
#draw function in the interval
curve( f(x), from=-0.1, to=1.5, lwd=1.5,ylim=c(-0.1,1.6),
        xlab="x(t)",ylab="f[x(t)]")
#add axes and 1-3 quadrant bisector
abline(v=0,h=0)
abline(0,1,col=2)
#compute 50 iterates for creating the cobweb
for(i in 1:50){
  x1 = f(x0)
  #draw segments of the cobweb
  lines( c(x0,x1,x1), c(x1,x1,f(x1) ), col='blue')
  x0 = x1
}
legend("right", c("x(t+1)=f[x(t)]","x(t+1)=x(t)","Cobweb"),
        col=c("black","red","blue"), cex=0.75, lty=c(1,1,1) )
```

■

Algorithm 10 Calculus of Variations - MATLAB

procedure CALCVAR(F,t_0,t_1,x_{t_0},x_{t_1})
  use "`syms`" for creating symbolic variables x,$\dot{x}$
  $F' \leftarrow$ use "`diff`" for differentiating F
  $Eeq \leftarrow$ compose the Euler equation
  $Esy \leftarrow$ "`symfun`" for creating symbolic function from Eeq
  $Ess \leftarrow$ "`char`" for creating string from Esy
  $x^*(t) \leftarrow dsolve(Ess)$ ▷ use "`dsolve`" function
  return Solution: $x^*(t)$
end procedure

Algorithm 11 Calculus of Variations - R

procedure CALCVAR(f,$\mathbf{x_1}$,n,$\mathcal{I}$)
  ▷ Part 1
  $F' \leftarrow$ use "`Deriv`" for differentiating F wrt x and $\dot{x}$)
  compute dF/dt by hand and input it
  $Eeq \leftarrow$ compose the Euler equation
  ▷ Part 2
  $Ef \leftarrow$ create an R function based on Eeq as system of first order equations
  $x^*(t) \leftarrow ode(Ef)$ ▷ use "`ode`" function
  return Plot: cobweb diagram
end procedure

6.6 Dynamic Optimization

There is not a function neither in MATLAB nor in R for solving the calculus of variations problems, therefore the author has coded two routines for addressing this issue. On the other hand, there exists a user-made package (called "`MDPtoolbox`") available both for MATLAB and R that provide the routines necessary for solving finite horizon discrete time Markov problems via the backward induction method.

6.6.1 Calculus of Variations

Algorithm (10) and Algorithm (11) describe the main steps of the two routines used for solving the basic version of the calculus of variations problem.

Example 6.6.1 *Solve the following calculus of variations problem with both MATLAB and R:*

$$\begin{cases} \max\limits_{x(t)} & \int_1^{20} \ddot{x}t^2 + 10tx \, \mathrm{d}t \\ s.t. & \\ \quad x(0) = 1 & \\ \quad x(0) = 1 & \end{cases} \tag{6.15}$$

- *MATLAB code for solving calculus of variations problems:*

```
syms t x D2x Dx;
F= (Dx^2)*t^2 +10*t*x;
ic= 'x(0)=1, Dx(0)=1'; times=[1 20];

function [eq, sol] = Eulereq_M(F,ic,times)
    % Input: functional written with symbolic arguments
```

```
    %        initial conditions as string
    %        time of evaluation (vector=[initial, final])
    % Output: Euler equation (symbolic) and solution (symbolic)

    syms t x Dx D2x
    % F'2
    dF2= diff(F, x);
    % F'3
    dF3= diff(F,Dx);
    % d(F'3)/dt
    dF3dt= diff(dF3, x)*Dx + diff(dF3, Dx)*D2x + diff(dF3, t);
    % Euler equation
    eq= dF2 - dF3dt;

    % Convert symbolic into suitable string for dsolve
    eqn= symfun(eq==0,[D2x Dx x t]);
    eqn= char(eqn);
    % solve
    sol= dsolve(eqn,ic);
    % plot solution
    t= linspace(times(1),1,times(2));
    z= eval(vectorize(sol));
    plot(t,z)
end

%-----EXAMPLE-----
% create symbolic variables
syms t x D2x Dx;
% Functional
F= (Dx^2)*t^2 +10*t*x;
% initial values + time
ic= 'x(1)=1, Dx(1)=1'; times=[1 20];
[eq sol] = Eulereq(F,ic)
```

- *R code for solving calculus of variations problems:*

```
Eulereq = function(F){
  # Inputs: functional as string, with variables "t", "x", "Dx"
  # Output: Euler equation as string
  require(Deriv)

  dF2 = Deriv(F,"x")
  dF3 = Deriv(F,"Dx")

  dF3dt_t = Deriv(dF3,"t")
  dF3dt_x = Deriv(dF3,"x")
  dF3dt_Dx = Deriv(dF3,"Dx")

  Eulereq= paste(dF2,"-(", dF3dt_t, "+(", dF3dt_x, ")*Dx+(",
                 dF3dt_Dx,")*D2x )")
  # exp= parse(text=Eulereq)
}

#-----EXAMPLE-----
# F="Dx^2 * t^2 +10*t*x"
# Eeq= Eulereq(F)

require(deSolve)
# 10*t -(4*(Dx*t) +(0)*Dx +(2*t^2)*D2x) = 10t -4tDx -2t^2D2x = 0

# write as: x= x1; Dx= x2= dx1; D2x= dx2
# initial values: x=1; dx=1
```

```
state=c(x1=1, x2=1)
param=c(a=10, b=-4, c=-2)
time= seq(1, 20, 0.01)

eq= function(t, state, parameters){
  with( as.list( c(state, parameters) ), {
    dx1= x2
    dx2= -(a*t +b*t*x2)* ((c*t^2)^(-1))
    list(c(dx1, dx2))
  })
}

sol= ode(y=state, times=time, func=eq, parms=param, method="ode45")

plot(sol[,"x1"], type="l",col="red",xlab="time",ylab="x(t)")
```

■

6.6.2 Dynamic Programming

The function to be used in both softwares is "`mdp_finite_hrizon`" from the "`MDPtoolbox`" package, which takes as main inputs: the transition probability array of dimension $S \times S \times A$ (containing a $S \times S$ matrix for each action A), the reward matrix of dimension $S \times A$, the number of time steps and (optionally) a terminal reward vector. The output is twofold: the optimal value function matrix, containing in each cell the value of being in state S_i at each time t; the optimal action matrix, which gives the optimal action (control) to be taken at each time t, conditional on being in a given state S_i.

Remark 6.6.1 *This function takes as input a finite set of actions which must be specified before running the algorithm. This is a very strong limitation to the kind of problems that can be solved, since in a very few cases it is possible to identify a priori a discrete, finite set of possible actions to be undertaken at each time. One could argue that by increasing the number of possible actions (that is, creating a finer grid for the control variable) with the aim of solving more complex problems, but the other face of the coin is a significant increase in the computational time.*

Example 6.6.2 *Use MATLAB and R to solve the deterministic finite horizon Markov problem over time $t = 1,2,3$ characterized as follows:*

(i) final reward at $t = 3$: equal to 1 in all states

(ii) reward matrix $\mathbf{R_t} = \mathbf{R}$ $\forall\, t = 1,2$

$$\mathbf{R} = \begin{bmatrix} 5 & 5 \\ 9 & 2 \\ 7 & 4 \end{bmatrix} \tag{6.16}$$

(iii) transition matrices P_t at $t = 1,2$

$$\mathbf{P_1} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.6 & 0.2 & 0.2 \\ 0.3 & 0.1 & 0.6 \end{bmatrix} \quad \mathbf{P_2} = \begin{bmatrix} 0 & 0.9 & 0.1 \\ 0.1 & 0 & 0.9 \\ 0.4 & 0.4 & 0.3 \end{bmatrix} \tag{6.17}$$

- *MATLAB code for solving finite horizon Markov problems using the backwards induction method (Bellman's principle):*

```
% transition array
P1 = [0.5 0.5 0; 0.6 0.2 0.2; 0.3 0.1 0.6];
P2 = [0 0.9 0.1; 0.1 0 0.9; 0.4 0.4 0.3];
P = cat(3, P1, P2);
% reward matrix
R = [5 5; 9 2; 7 4];
% paramters
discount= 1;
time = 3;
final = [1 1 1];
% solve
[val, act] = mdp_finite_horizon(P, R, discount, time, final);

% plot optimal value function and action
figure(1)
hold on
plot(val(1,:),'-r'); plot(val(2,:),'-b'); plot(val(3,:),'-black')
legend('State 1','State 2','State 3', 'Location','northeast')
title('Optimal Value function');
xlabel('time'); ylabel('Value function')
hold off
figure(2)
hold on
plot(act(1,:),'-r'); plot(act(2,:),'-b'); plot(act(3,:),'-black')
legend('State 1','State 2','State 3', 'Location','northeast')
title('Optimal Action'); xlabel('time'); ylabel('Action')
axis([1 3 0.5 2.5])
hold off
```

- *R code for solving finite horizon Markov problems using the backwards induction method (Bellman's principle):*

```
library(MDPtoolbox)
# transition array
P = array(0, c(3,3,2))
P[,,1] = matrix(c(0.5,0.5,0, 0.6,0.2,0.2, 0.3,0.1,0.6),
        3, 3, byrow=TRUE)
P[,,2] = matrix(c(0,0.9,0.1, 0.1,0,0.9, 0.4,0.3,0.3),
        3, 3, byrow=TRUE)
# reward matrix
R = matrix(c(5,5, 9,2, 7,4), 3, 2, byrow=TRUE)
# parameters
discount= 1
time = 3
final = c(1,1,1)
# solve
sol = mdp_finite_horizon(P, R, discount, time, final)

# plot optimal value function and action
plot(sol$V[3,],pch=16,xlab="time",ylab="Value function",
        main="Optimal Value function")
points(sol$V[1,],pch=16,col="red"); lines(sol$V[1,],col="red")
points(sol$V[2,],pch=16,col="blue"); lines(sol$V[2,],col="blue")
lines(sol$V[3,],col="black")
legend("topright",legend=c("State 1","State 2","State 3"),
        col=c("red","blue","black"),pch=16,cex=0.8)

plot(sol$policy[1,],pch=16,xlab="time",ylab="Action",
        main="Optimal action",col="red")
points(sol$policy[2,],col="blue",pch=16)
points(sol$policy[3,],col="black",pch=16)
```

```
lines(sol$policy[2,],col="blue")
lines(sol$policy[3,],col="black")
lines(sol$policy[1,],col="red")
legend("topright",legend=c("State 1","State 2","State 3"),
        col=c("red","blue","black"),pch=16,cex=0.8)
```

■

Chapter 7

Conclusion

The modern approach to applied mathematics is twofold: on one hand, the analytical methods offer exact and fastly implementable solutions to a rather narrow set of problems. On the other the computational approach, which involves both the implementation of analytical results in high dimensions as well as the use of numerical methods, is becoming increasingly popular.

We reviewed some of the most important results in optimization theory (both static and dynamic), then we outlined the procedures that should be followed in order to solve these kind of problems. First of all, a short summary of some basic results in mathematics (including calculus and linear algebra) has been provided, then static unconstrained and constrained optimization problems have been discussed in detail. Next, the we presented the theory of ordinary differential equations and difference equations in order to provide a solid background for carrying out the basic dynamic optimization problems, explained below.

Finally, we introduced the meaning and fundamental logic behind the computational approach to applied mathematics. As far as the applications are concerned, we provided several examples of exercises solved using the analytical results as well as others carried out by using the computational tools, more specifically the softwares MATLAB® and R.

Bibliography

[1] Andrescu, T. and D. Andrica (2014): *Complex Numbers from A to...Z*, Springer. 7

[2] Brandimarte, P. (2006): *Numerical Methods in Finance and Economics - A MATLAB-Based Introduction*, John Wiley & Sons, second ed. 88, 94

[3] Carter, M. (2001): *Foundations of Mathematical Economics*, MIT Press. 7, 17

[4] Daróczi, G. and M. Puhle (2013): *Introduction to R for Quantitative Finance*, PACKT Books. 88, 94

[5] de la Fuente, A. (2000): *Mathematical Methods and Models for Economists*, Cambridge University Press. 7, 17

[6] Gilat, A. (2011): *MATLAB - An Introduction with Applications*, John Wiley & Sons, fourth ed. 88

[7] Kamien, M. I. and N. I. Schwartz (1991): *Dynamic Optiization, the Calculus of Variations and Optimal Control in Economics and Management*, North Holland, second ed. 17

[8] Kendrick, D., R. Mercado, and H. Amman (2006): *Computational Economics*, Princeton University Press. 88, 94

[9] Kleiber, C. and A. Zeileis (2008): *Applied Econometrics with R*, Springer. 88, 94

[10] Ok, E. (2005): *Real Analysis with Economic Applications*, New York University. 17

[11] Robert, C. P. and G. Casella (2004): *Monte Carlo Statistical Methods*, Springer, second ed. 88

[12] Simon, P. and L. Blume (1994): *Mathematics for Economists*, Norton & Company. 7, 17

[13] Sun, C. (2015): *Empirical Research in Economics: Growing up with R*, ERER Book. 88, 94

[14] Sundaram, R. (1996): *A First Course in Optimization Theory*, Cambridge University Press. 7, 17

[15] Sydsæter, K., P. Hammond, A. Seierstad, and A. Strøm (2008): *Further Mathematics for Economic Analysis*, Prentice Hall, second ed. 12, 15, 16, 17, 32, 33, 39, 42, 44, 48

[16] Sydsæter, K., P. Hammond, and A. Strøm (2012): *Essential Mathematics for Economic Analysis*, Pearson, fourth ed. 7