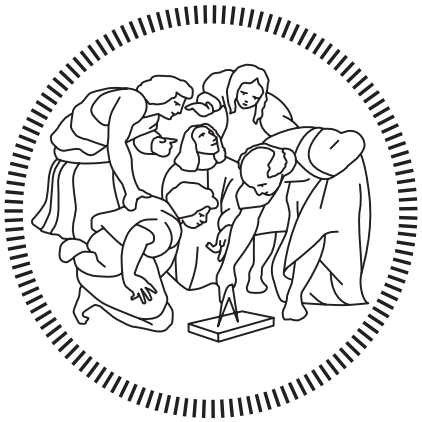
Politecnico di Milano

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE
Master of Science – Mathematical Engineering



Numerical modelling of optimal vaccination strategies for SARS-CoV-2

Advisor
Prof. Marco Verani

Co-advisor
Prof. Stefano Berrone

MSc Candidate:
Giovanni Ziarelli
940123

Academic Year 2020 – 2021

Sommario

Il 31 dicembre 2019 le autorità sanitarie cinesi comunicano all'Organizzazione Mondiale della Sanità la diffusione nella città di Wuhan di una malattia che causa crisi respiratorie acute dal carattere fortemente infettivo. Inizia così a diffondersi il virus SARS-CoV-2, artefice della pandemia da COVID-19, la più grande crisi sanitaria mondiale dell'ultimo secolo. Sul finire del 2020, la pandemia, che aveva già causato 1.8 milioni di morti nel mondo, subisce una svolta decisiva attraverso l'approvazione di vaccini sviluppati appositamente per il COVID-19.

In questa tesi viene presentato ed implementato uno strumento matematicamente sofisticato e computazionalmente efficiente in grado di risolvere problemi di Controllo Ottimo per la distribuzione di vaccini per il COVID-19. Dapprima viene formulato un nuovo modello epidemico adattato per catturare alcune delle peculiarità della diffusione progressiva della malattia e della campagna vaccinale: il modello `SEIHRDVW`, sviluppato congiuntamente con il gruppo di ricerca EpiMox del Laboratorio di Modellistica e Calcolo Scientifico (MOX) del Politecnico di Milano. Vengono definite due diverse tipologie di problemi di Controllo Ottimo governati da sistemi di Equazioni Differenziali Ordinarie in cui le variabili di controllo sono le somministrazioni quotidiane di prime e seconde dosi vaccinali. Pertanto è possibile proporre idealmente delle linee guida concrete per la pianificazione della campagna vaccinale ottimale per la minimizzazione del numero di morti o la quantità di persone infettate in un tempo fissato. Per la risoluzione numerica dei problemi di Controllo Ottimo viene introdotto un algoritmo innovativo, cioè il Metodo del Gradiente Proiettato Multiplo, in grado di restituire le distribuzioni ottimali di vaccini che soddisfino vincoli precedentemente definiti, imposti da fattori farmacologici e dalla disponibilità di dosi. I problemi di Controllo Ottimo vengono risolti inquadrando il problema sia in scenari artificiosi (i cui parametri rappresentano situazioni idealizzate), sia nello scenario quasi-realistico della Lombardia nel 2021.

Per la risoluzione dei problemi di controllo ottimo, è stato implementato un solutore in `python3`, sfruttando la differenziazione automatica e la compilazione `just-in-time` definite nella libreria `jax`.

Abstract

On the 31st December 2019, Chinese health authorities notified the World Health Organisation of the spread of a disease in the city of Wuhan that causes acute respiratory crises of a highly infectious nature. Thus, the SARS-CoV-2 virus, the causative agent of the COVID-19 pandemic, begins to spread resulting in the most devastating global health crisis of the last century. In December 2020, the pandemic, which already caused 1.8 million deaths worldwide, takes a decisive turn with the approval of vaccines developed specifically for COVID-19.

In this thesis, a mathematically sophisticated and computationally efficient tool for solving Optimal Control problems for the distribution of COVID-19 vaccines is presented and implemented. First, we formulate a new epidemic model adapted to capture some of the peculiarities of the progressive evolution of the disease and of the vaccination campaign: the `SEIHRDVW` model, developed jointly with the EpiMox research group of the Laboratory of Mathematical Modelling and Scientific Computing (MOX) at Politecnico di Milano. We introduce two different types of Optimal Control problems governed by systems of Ordinary Differential Equations where the control variables are the daily administrations of first and second vaccine doses. Therefore, it is possible to ideally propose concrete guidelines for planning the optimal vaccination campaign for minimising the number of deaths or the amount of people infected in a fixed time. For computing the numerical solution of Optimal Control problems, we develop an innovative algorithm, namely the Multi-Projected Gradient Method, which is able to return the optimal distributions of vaccines that satisfy previously defined constraints imposed by pharmacological factors and dose availabilities. Optimal control problems are solved by framing the problem both in artificial scenarios (where parameters represent idealised situations) and in the semi-realistic scenario of Lombardy in 2021.

To cope with Optimal Control problems, we implemented a solver in `python3` exploiting the automatic differentiation and the `just-in-time` compilation provided in the `jax` library.

Ringraziamenti

Ore 23:44 del 7 settembre 2021. Il silenzio, la notte e anche il fresco. Questo è il momento per dire dei grazie, perché imparando ho deciso che non avrei mai smesso.

Innanzitutto desidero ringraziare sinceramente il professor Marco Verani, non solamente per avermi accompagnato, seguito e guidato negli ultimi mesi, ma soprattutto per la profonda umanità, la disponibilità gratuita e la cortesia con cui lo ha fatto. Grazie perché questo lavoro non porta solamente alla chiusura di un percorso, ma ad una nuova strada che umilmente mi appresto ad intraprendere, e che certamente non avrei valutato senza i suoi preziosi consigli. Ringrazio con la stessa riconoscenza i professori Nicola Parolini, Luca Dede' e Alfio Quarteroni, e tutto il gruppo di ricerca di EpiMox. Grazie per i numerosissimi stimoli, per i confronti nelle *tavole rotonde* virtuali in cui questo lavoro è stato man mano costruito, per la disponibilità e per il tempo speso. Ringrazio inoltre il professor Stefano Berrone per la sua disponibilità.

Per la mia famiglia, fedele compagna in tutti i momenti vissuti, a volte più vicina a volte più lontana, un grazie sincero dal profondo del mio cuore. Ringrazio per mia madre Emanuela, che nelle parole ma soprattutto nei gesti trasmette quell'amore consistente e immeritato che razionalmente non puoi spiegare. Ringrazio per mio padre Emilio, per le spalle grandi e per il cuore tenero, per la leggerezza e per le risate, che (ogni tanto) servono. Ringrazio per mia sorella Eleonora, per farsi vicina, sempre, perché il suo posto nella mia vita è come il mio posto nella sua: accanto. E perché in fondo ci somigliamo più di quello che pensiamo. Ringrazio per i miei nonni, perché consapevoli del poco tempo che avremmo passato insieme, non si sono mai risparmiati, né lamentati. E un grazie di cuore per Leonardo, Luigi e Monica, che sono di famiglia.

Grazie per Francesca, Letizia, Margherita e Sara, per la costanza, la sincerità, e il bene che ci vogliamo. Porto con me i vostri abbracci e le vostre risate. Da voi ho imparato che in amicizia non ci si può risparmiare.
Grazie per Francesco e Giovanni, per il vostro cuore buono.
Grazie per Matteo, per il cammino iniziato e mai finito, e perché con me non lascia mai la presa.
Grazie per Emilia e Vincenza, per il pensiero costante, mai nostalgico, sempre nuovo.

Grazie per Anna (B.), per gli abbracci, per la vicinanza e per la distanza, per la diversità e la somiglianza.
Grazie per Anna (T.), per l'umiltà nelle scelte grandi.
Grazie per Dario ed Emanuele, per il ritrovarsi sempre come ci si è lasciati.
Grazie per Francesca, per gli occhi trasparenti, perché non finisce mai di stupirsi e di stupirmi.
Grazie per Gian Marco, perché sceglie di esserci. Sempre.
Grazie per Ilaria, perché mi insegna a scegliere con il cuore.
Grazie per Letizia, Daniele e il piccolo Nicola, per la testimonianza di amore.
Grazie per Luca, per la saggezza e la fratellanza.
Grazie per Marika, perché il suo sguardo mi indica sempre il Centro. Per la tenerezza e gli abbracci infiniti.
Grazie per Vittorio e Clara, per l'amicizia sincera e la presenza inaspettata.
Grazie per Paola e Leo, per i sogni, quanti sogni. Per i progetti, le speranze e le lunghe chiacchierate. Per la porta sempre aperta. Per la famiglia. Per come nella mia vita sono sempre Presenza e mai assenza.
Grazie per Gio e Speri, per la casetta troppo piccola, per gli urletti intorno al tavolo, per i fiori sul prato della Cons e per gli abbracci che sciolgono.

Grazie per la famiglia dell'AC, perché la comunione che la caratterizza la ritrovo sempre e perché dovunque io vada la ritrovo sempre nella mia strada.
Grazie per Spagliagrano. Per l'Amore che salva. Per i ragazzi incontrati e per quelli che incontreremo. Per i sogni che partono dallo stomaco, passano alla testa e rimangono nel cuore.

Grazie a Te, per tutte queste forme in cui ti sei manifestato. Grazie per la memoria e il desiderio. Grazie perché mi rendi capace proprio laddove meno lo merito. Grazie perché doni e porti via, donando a tutto il giusto tempo. Grazie, perché, dovunque, non Ti stanchi mai di venirmi a cercare.

"Strada facendo, predicate, dicendo che il regno dei cieli è vicino.
Guarite gli infermi, risuscitate i morti, purificate i lebbrosi, scacciate i demòni.
Gratuitamente avete ricevuto, gratuitamente date."
Mt 10, 7-8

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The term *Epidemiology* merges the greek prefix *Epi*, meaning *upon*, with *Demos*, *people*, and *Logos*, standing for *reason, study*. It is literally the study of infections that involve human population. Among the other epidemic events originated by different biological viruses or bacteria, pandemic events are those sanitarian emergencies which are caused by abnormal spreads of infectious viruses or bacteria strains [84]. The word pandemic has Greek roots too and literally means *all people*, meaning that the epidemic disease spreads worldwide. During the last centuries many epidemic events have ravaged the world, for instance the Athens plague (Typhoid fever) in 430 B.C., Smallpox Antonine plague 165-180 in Italy, the Black Death which spread worldwide during the 14th century, the Third Plague Pandemic started in China in the 19th century or the current pandemic due to SARS-CoV-2 virus.

The *Severe Acute Respiratory Syndrome Coronavirus 2* (SARS-CoV-2) epi-centered in Wuhan in the Chinese province of Hubei and then rapidly spread worldwide as a consequence of the extremely connected society we are living in. Coronaviruses are a group of enveloped, single-stranded RNA-viruses that have a wide-ranged tropism, and are able to cause devastating diseases [32]. The three most notable coronaviruses which affected human population are SARS-CoV-1 in 2001, MERS-CoV in 2012, both causing epidemic events, and SARS-CoV-2 in 2019, the causative agent of the current coronavirus disease (COVID-19) pandemic. According to the WHO dashboard[1], as of June 2021 $1.78e8$ people have been infected by the virus, and $3.9e6$ of them died. Structural components of the virus include the membrane, envelope, nucleocapsid and Spike proteins. The entry of SARS-CoV-2 into host cells depends on two different factors:

1. binding of the viral Spike proteins to cellular receptors;
2. Spike proteins priming by the host cell protease.

Figure 1.1 displays schematically SARS-CoV-2 structure and how it interacts with host cells. Patients affected by SARS-CoV-2 present different clinical symptomps including

[1] `https:// covid19.who.int/`

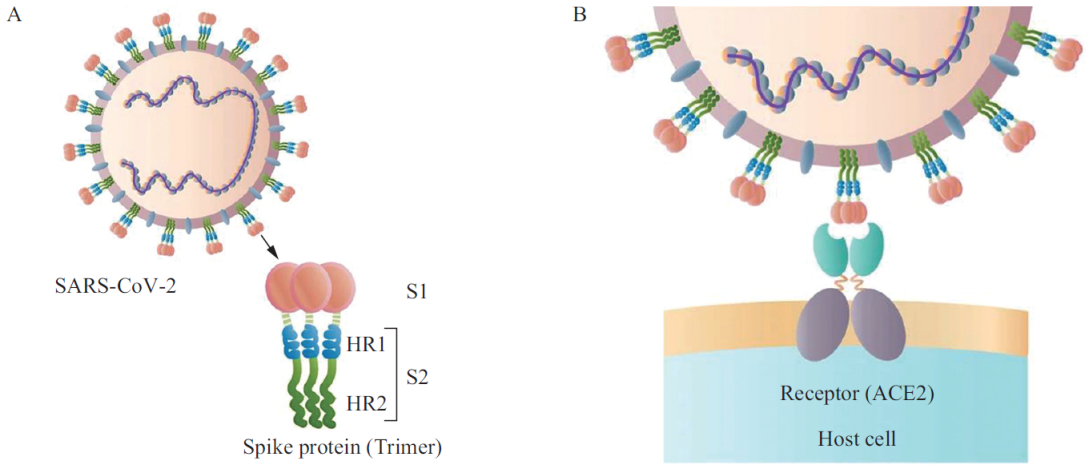


Figure 1.1: Schematic structure of SARS-CoV-2 virus. **A)** Spike protein structure; **B)** interaction between SARS-CoV-2 and host cells. Source: [41].

fatigue, fever, cough, loss of smell and tasting and headache. Almost 30% of them reported gastrointestinal symptomps such as diarrhea, nausea, and stomach pain [90]. A study based on 254 chinese patients [122] assesses that the most common complications were pneumonia (82.3%), arrhythmia (0.06%) and shock (0.03%). In patients recovered for COVID-19 infections the most prevalent comorbidities are age, obesity, hypertension, diabetes mellitus, heart diseases and lung diseases [23]. Multiple surveillance methods have been used to monitor the spread of SARS-CoV-2. The nasopharynx and nasal swabs PCR tests have been the dominant form of testing. Despite the prompt development of vaccines and the numerous ongoing clinical trials about other therapeutics, the SARS-CoV-2 infection is still a large public health concern.

In recent years, epidemiology has started integrating mathematics, sociology, management science, complexity science, and computer science. The resulting cross of multiple disciplines has led to the development of mathematical and computational approaches to epidemic modeling, such as mathematical modelling. Mathematical predictions through epidemic models can open the way to the optimal application of control strategies and prevention treatments which basically contribute in eradicating the epidemic, even though they cannot be overexploited. Indeed, since vaccinations supply and medical interventions are often limited due to *apriori* economical and political constraints, they could not be sufficient and effective enough for facing outbreaks on their own. Besides, politically imposed restrictions which help in preventing the spread of the disease can have negative repercussions on the social, economic and psychological point of view if they are overextended [105]. Some works are devoted to provide a mathematical response for optimal implementation of both medical and non-pharmaceutical interventions during epidemic events. For instance, [8] provides an analysis of optimal scenarios for treatments and educational campaign strategies with the objective of minimizing infectious individuals during a fixed timeframe. In [42]

vaccinations are used for controlling the total amount of infected individuals solving a mathematical Optimal Control problem based on a `SIR` model (see Section 2.2.1). An Optimal treatment strategy for the administration of antiretroviral drug (*Reverse Transcriptase Inhibitors*) in HIV-positive individuals is investigated in [69]. Effects and optimal strategies for medical treatments and quarantine have been investigating in [51], taking into account for multiple strains of the disease itself.

With this work, we aim at contributing to the Italian panorama of epidemic modelling for the SARS-CoV-2 by introducing an original *vaccination-oriented* model and proposing an innovative optimization algorithm specifically developed for coping with the complex problem of vaccine distribution. In particular:

- We introduced vaccination compartments in an already-estabilished epidemic model (private communication Luca Dede'), the `SEIHRD`, which has been developed by the EpiMox research group at MOX Laboratory of Politecnico di Milano. Particular attention has been devoted to the relationship between classes of vaccinated with first and second dose (indeed we consider two-shots vaccines) and to the role of vaccine administrations to recovered individuals. Thus, we set up a model completely conceived with the Italian vaccination program for SARS-CoV-2;

- We formulated Optimal Control Problems to achieve the eradication of the pandemic and the minimization of deceased due to the infection using vaccinations as control variables. For this purpose we had to express mathematically the constraints imposed on vaccine administrations due to temporal minimum and maximum elapsing time among consecutive doses and to their availability;

- We proposed and implemented a novel optimization algorithm for dealing with the formulated multi-constrained optimization problems: the Multi-Projected Gradient Descent (`MultiPGD`). We solved Optimal Control problems through the `MultiPGD` in artificial scenarios to ideally provide policy makers with qualitative guidelines for an optimal vaccination campaign, and in the semi-realistic scenario of Lombardy in 2021.

Outline

The thesis is organized in two parts:

Part 1. Epidemic Models, Optimal Control and Numerical Methods.

- Chapter 2 shortly reviews some of the Epidemic Models that are present in the state of the art;
- Chapter 3 is devoted to provide instrumental results of Optimal Control theory;
- Chapter 4 introduces a new epidemic model, named `SEIHRDVW` model, incorporating the vaccination process and formulates Optimal Control problems to deal with the optimal vaccination campaign;
- Chapter 5 illustrates numerical details about the implemented algorithm for solving Optimal Control problems;

Part 2. Numerical Results.

- Chapter 6 collects and discusses computational results related to artificial scenarios with the goal of underlining qualitative key-features of optimal vaccination campaigns;
- Chapter 7 collects and discusses numerical results related to a more realistic scenario where we employ some realistic parameters to frame the simulation in Lombardy and to account for realistic vaccines for fighting SARS-CoV-2.

The final chapter (Chapter 8) draws concluding remarks and includes possible future developments of this work.

Part I

Epidemic Models, Optimal Control and Numerical Methods

Chapter 2

Epidemic models

Mathematical modeling in epidemiology is concerned with describing the spread of diseases and their effect on people. In order to cope with infectious spread due to communicable diseases, a wide area of the mathematical community developed different mathematical models able to catch peculiar dynamics of the disease of concern. This chapter is entirely devoted to the overview of some of the most adopted models for describing different diseases.

We first recall some mathematical basics of Continuous Dynamical Systems and Bifurcation theory in Section 2.1. In Section 2.2 we discuss three fundamental archetypes of Compartmental Epidemic Models, pointing out how to model medical and social interventions. Section 2.3 is devoted to a short presentation of possible PDE-based epidemic models. Finally, in Section 2.4 we present a short review of the state of the art concerning mathematical epidemic models applied to SARS-CoV-2 infection, with a specific focus on the `SUIHTER` model [89].

2.1 Continuous Dynamical Systems

In this section some fundamental definitions and results of Dynamical Systems Theory are introduced in accordance with [53], [28] and [37].

Definition 1. *Let X be a set and $(T,+)$ an abelian group with identity element e. A* ***dynamical system*** *on X is a map $\eta : T \times X \to X$ such that*

$$\begin{aligned} \forall x \in X : \quad & \eta(e, x) = x, \\ \forall s, t \in T, \forall x \in X : \quad & \eta(s, \eta(t, x)) = \eta(s + t, x). \end{aligned} \tag{2.1}$$

A dynamical system on $T = \mathbb{Z}$ or $\mathbb{N}$ is called discrete; a dynamical system with $T = \mathbb{R}$ or $\mathbb{R}^+$ is called continuous.

The function $t \mapsto \eta(t, x)$ is called **trajectory** or **orbit** through x.

Definition 2. *An $\bar{x} \in X$ such that*

$$\forall t_1, t_2 \in T: \quad \eta(t_1, \bar{x}) = \eta(t_2, \bar{x}) \tag{2.2}$$

is called ***equilibrium point*** *for the dynamical system η.*

Taking $t_1 = t$, generic time, and $t_2 = e$, if $\bar{x}$ is an equilibrium point the following holds

$$\eta(t, \bar{x}) = \bar{x} \ \ \forall t \in T, \tag{2.3}$$

meaning that the trajectory starting from an equilibrium point is represented by the starting position itself. Assuming now an algebraic structure for the space X, namely assuming that X is a vector or a difference space (*i.e.* a space where the displacement between two distinct points can be defined), we can introduce differential calculus in the dynamical systems framework. Indeed,

$$\frac{d}{dt}\eta(t, x) = \dot{\eta}(t, x) = \lim_{\epsilon \to 0} \frac{\eta(t + \epsilon, x) - \eta(t, x)}{\epsilon} = \lim_{\epsilon \to 0} \frac{\eta(\epsilon, \eta(t, x)) - \eta(t, x)}{\epsilon} \tag{2.4}$$

is correctly defined. We notice that the above quotient is a function exclusively of $\eta(t, x)$ and therefore

$$\dot{\eta}(t, x) = F(\eta(t, x)). \tag{2.5}$$

Hence, the evolution of dynamical systems obeys to autonomous differential equations. In the case of vectorial variables, we verify that multidimensional dynamical systems are associated with autonomous systems of differential equations. At the same time, even the converse is true. Indeed, if a function $t \mapsto u(t)$ on a topological vector space X solves the following Cauchy problem

$$\begin{cases} \dot{u}(t) = F(u(t)), \\ u(0) = x, \end{cases} \tag{2.6}$$

with $t \in \mathbb{R}$, then the function $(t, x) \mapsto u(t) = \eta(t, x)$ is a dynamical system on T and X according to Definition 1. The first condition is trivial to verify, since $\eta(0, x) = u(0) = x$. For the second, fix $s \in \mathbb{R}$ and define $v_s : t \mapsto u(t + s)$. Since $v_s(t)$ evolves according to the following differential system,

$$\begin{cases} \dot{v_s}(t) = F(v_s(t)), \\ v_s(0) = u(s), \end{cases} \tag{2.7}$$

then $v_s(t) = u(t + s) = \eta(t + s, x)$, and contemporarily,

$$\eta(t + s, x) = v_s(t) = \eta(t, u(s)) = \eta(t, \eta(s, x)),$$

which is exactly the second condition.

Identifying equilibria means finding those points corresponding to stationary trajectories constituted by a single point. However, it is not always true that starting close to the equilibrium position, the trajectory will remain close to the point.

Definition 3. *Let X be a topological space and let $\mathcal{I}_x$ the set of all neighbourhoods of point $x \in X$. An equilibrium position $\bar{x}$ of a dynamical system is said to be* ***stable*** *if*

$$\forall V \in \mathcal{I}_{\bar{x}} \; \exists U \in \mathcal{I}_{\bar{x}} \;\; s.t. \;\; \forall x \in U \Rightarrow \eta(t,x) \in V \; \forall t \geq 0,$$

otherwise it is said to be ***unstable****.*

Stability is a very tough property to check if $\eta(t,x)$ is not explicitly defined, or if X is a generic topological space. In the following we restrict ourselves to $X = \mathbb{R}^n$ and $T = \mathbb{R}$, *i.e.* the case of ordinary differential equation.

Theorem 1. *Assume there exists a unique solution $t \mapsto u(t)$ of (2.6) $\forall t \in \mathbb{R}$. Let $\bar{x}$ be an equilibrium position of the system and set*

$$A = \left(\frac{\partial F}{\partial u} \right) \Big|_{\bar{x}},$$

the Jacobian of F computed at the equilibrium. If all eigenvalues associated to A have strictly negative real part, then $\bar{x}$ is stable. If at least one eigenvalue has strictly positive real part, then $\bar{x}$ is unstable.

This theorem is known in literature as the Linearization Method (for further details refer to [53]). In the neutral case where at least one eigenvalue is purely immaginary there may be stability as well as instability. For those cases where Linearization fails, a sufficient condition is provided by the concept of *Ljapunov function* [17].

Definition 4. *Let η be a dynamical system on $X = \mathbb{R}^n$ having $\bar{x}$ as equilibrium point and let $W : X \to \mathbb{R}$ a continuous function having a strict minimum in $\bar{x}$. W is said to be a* ***Ljapunov function*** *relative to $\bar{x}$ if there exists a neighbourhood Z of $\bar{x}$ such that*

$$\forall x \in Z, \forall t_1, t_2 \geq 0 : \; t_1 < t_2 \Rightarrow W(\eta(t_2, x)) \leq W(\eta(t_1, x)).$$

This is equivalent to say that W has a strict minimum in $\bar{x}$ and the function $t \mapsto W(\eta(t,x))$ is nonincreasing.

Theorem 2. *If a dynamical system on $X = \mathbb{R}^n$, continuous in time, has an equilibrium point $\bar{x}$ which admits a Ljapunov function relative to it, then it is stable.*

Ljapunov functions are in general not intuitive to find. If Theorem 2 is verified, Ljapunov functions often carry on important information on the system, especially when they are constant on trajectories. An example of Ljapunov function is the total mechanical energy in conservative holonomic systems.

The notion of stability does not include information about asymptotic behaviours of orbits, *i.e.* does not take into account any possible dissipative effects. For this purpose, we introduce the concept of *attractivity*.

Definition 5. *Let η be a dynamical system on X having $\bar{x}$ as equilibrium point. $\bar{x}$ is said to be* ***attractive*** *if there exists a neighbourhood U of $\bar{x}$ such that*

$$\forall x \in U : \lim_{t \to +\infty} \eta(t, x) = \bar{x}.$$

If X is simply a metric space, a set $Y \subseteq X$ is attractive if the same holds starting from a point x where $\operatorname{dist}(\eta(t, x), Y) \leq \epsilon$, with $\operatorname{dist}(x, Y) = \inf_{y \in Y} d(x, y)$.

Definition 6. *Let η be a dynamical system on X having $\bar{x}$ as equilibrium point. If $\bar{x}$ is stable and attractive, then it is said to be* ***asymptotically stable****.*

Attractivity does not imply stability. In Appendix A there is an example of a dynamical system which equilibrium is attractive but not stable. Asymptotic stability can be shown both via Linearization Method or through a more strict Ljapunov theorem.

Theorem 3. *Let $t \mapsto u(t)$ be a solution of (2.6) and suppose that it exists $\forall t \in \mathbb{R}$. Let $\bar{x}$ an equilibrium position of the system and set*

$$A = \left(\frac{\partial F}{\partial u} \right) \Big|_{\bar{x}}.$$

If all eigenvalues associated to A have strictly negative real part, then $\bar{x}$ is asymptotically stable.

Definition 7. *Let η be a dynamical system on $X = \mathbb{R}^n$ having $\bar{x}$ as equilibrium point and let $W : X \to \mathbb{R}$ a continuous function having a strict minimum in $\bar{x}$. W is said to be a* ***strict Ljapunov function*** *relative to $\bar{x}$ if there exists a neighbourhood Z of $\bar{x}$ such that*

$$\forall x \in Z,\ x \neq \bar{x} \Rightarrow \forall t_1, t_2 \geq 0 :\ t_1 < t_2 \Rightarrow W(\eta(t_2, x)) < W(\eta(t_1, x)),$$

$$\forall x \in Z : (\forall t_1, t_2 \geq 0 \quad W(\eta(t_1, x)) = W(\eta(t_2, x)) \Rightarrow x = \bar{x}.$$

This definition implies that W is always strictly decreasing on trajectories except for the equilibrium point.

Theorem 4. *If a dynamical system on $X = \mathbb{R}^n$ which is continuous in time has an equilibrium point $\bar{x}$ admitting a strict Ljapunov function relative to it, then it is asymptotically stable.*

2.1.1 Bifurcation theory

Definition 8. *Let X be a set and $\gamma : \mathbb{R} \to X$ a curve. Let $\mathcal{F}$ be a partition*[1] *of X. A value $k_0 \in \mathbb{R}$ for which $\gamma(k) \in G$ for $k > k_0$ and $\gamma(k) \in F$ for $k < k_0$ with $F \neq G$ is called* ***bifurcation value****, and the element $\gamma(k_0)$* ***bifurcation point****.*

The idea behind this definition is that crossing k_0 the function $\gamma(k)$ changes its behaviour. The parameter k is the quantity depending on which we want to investigate how the bifurcation function γ changes. The same definition can be extended to cases where the function γ depends on more than one parameter (*i.e.* $\gamma : \mathbb{R}^n \to X$).

Equilibria of dynamical systems can exhibit different stability properties (or can change in number) depending on different values of some parameters. An example of bifurcation function is the one counting the number of equilibria of a specified dynamical system. In this case, we draw the *bifurcation diagram* which is a visual qualitative representation of equilibria of the dynamical system by varying the parameter k on which γ depends. The convention we are going to use is to represent stable equilibria as continuous lines, and unstable ones as dashed lines.

Generally, the bifurcation diagram is constituted by union of curves in a space of the same dimension of the dimension of k. Assuming the evolutive function is sufficient regular on the bifurcation parameters, the curves intersect each other in potential bifurcation points. According to the purpose of this thesis, we categorize three main types of bifurcation points (see Figure 2.1):

- **Saddle-node bifurcation**. It is defined when a curve folds up on itself;
- **Transcritical bifurcation**. It is defined when two curves, which are locally graphs of functions, intersects each other;
- **Pitchfork bifurcation**. It is defined when two curves intersect each other and at least one folds up on itself.

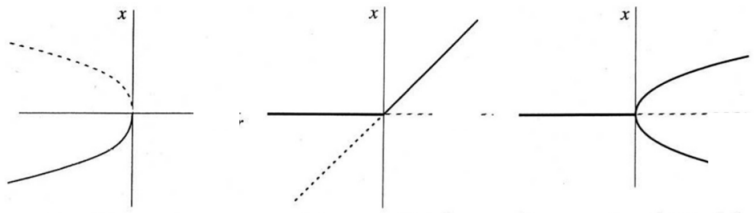


Figure 2.1: Bifurcation diagrams representing three different bifurcation points. Source: [53].

[1] A partition of X is a family $\mathcal{F}$ of subsets of X such that $\emptyset \notin \mathcal{F}, F \bigcap G = \emptyset$ for every $F, G \in \mathcal{F}$ and $\bigcup_{F \in \mathcal{F}} F = X$.

2.2 Compartmental Epidemic Models

In 1927 William O. Kermack and Anderson G. McKendrick published their first work about epidemic models from the Laboratory of Royal College of Physicians in Edinburgh [64]. They introduced the SIR model that is considered as the cornerstone of modern mathematical epidemiology since it is the first compartmental model describing the spread of infectious diseases among susceptibles for human-to-human contact. The authors themselves subsequently pusblished adjournments of their theory (in 1932 [65] and 1933 [66]) in order to model epidemic events persistent in time.

Three are the main archetypes of compartmental deterministic models for the spread of infectious diseases by direct person-to-person contact, the SIR, SIS and SEIR models (see Figure 2.2). They can describe different diseases with different biological characteristics, and moreover they constitute the bases for the development of more complex models. Those kinds of epidemic models have been applied to a wide range of diseases, for instance measles [7], rubella [5], whooping cough [1], polyiomelites [19], chickenpox [31], mumps [45], fever [43] and HIV (see, *e.g.*, [104], [72] and [50]).

In the following subsections we propose a short overview of some of the most common compartmental models for epidemic events, and we investigate qualitative behaviours of the solutions.

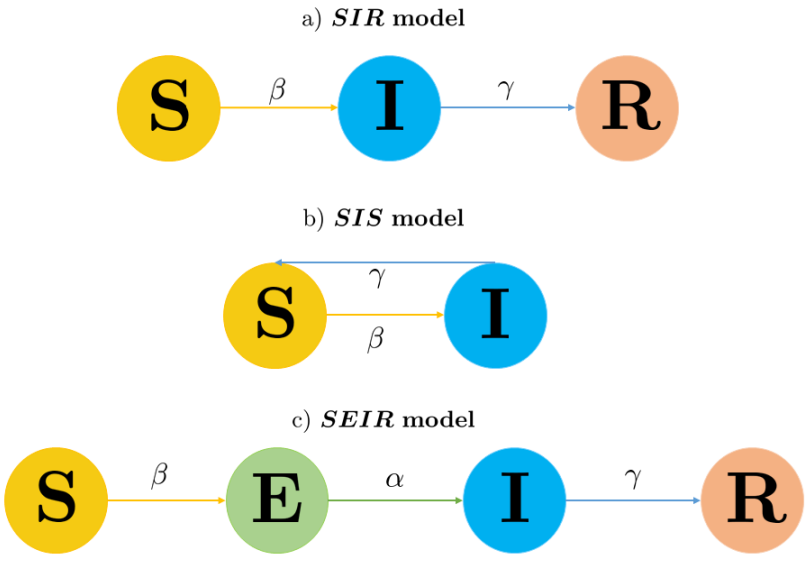


Figure 2.2: Schematic representation of SIR, SIS and SEIR models.

2.2.1 SIR model

The SIR model (Figure 2.2.a) is a compartmental model which assumes to subdivide population in three main disjoint average classes. The susceptible class ($\boldsymbol{S}$) is consti-

tuted by individuals who can contract the infection. Instead, the infectious class ($\boldsymbol{I}$) is composed by ill individuals who are responsible for the epidemic spread. When infected individuals have recovered from the illness, they end up in the removed class ($\boldsymbol{R}$). The `SIR` model is based on some restrictive assumptions which cannot be circumvented:

- infected individuals are also infectious;
- it assumes to describe properly epidemic outbreaks and not time-persistent epidemics;
- the population is homogeneously mixed, hence each involved parameter is an average quantity;
- it neglects migratory phenomena, births and deaths (also called *vital dynamics*) and assumes to be extremely localized in time and space. The total population per-time remains constant, *i.e*

$$S(t) + I(t) + R(t) = N; \tag{2.8}$$

- it assumes to deal with viruses that grant immunity after the illness stage, and so reinfection is not allowed;
- the disease is not fatal;
- when a susceptible individual is infected there is instantaneous manifestation of symptoms. Incubation time is not considered.

The `SIR` model can be formulated as a system of Ordinary Differential Equations (ODEs) as follows:

$$\begin{cases} \dot{S} = -\dfrac{\beta}{N} S\, I, & \forall t \in (0; T] \\ \dot{I} = \dfrac{\beta}{N} S\, I - \gamma I, & \forall t \in (0; T] \\ \dot{R} = \gamma I, & \forall t \in (0; T] \end{cases} \tag{2.9}$$

where β ($[\frac{1}{T}]$) is the *daily contact or transmission rate*, namely the parameter averaging the number of adequate contacts per infective per day. Adequate contacts are the interactions among infectives and susceptibles which successfully outcome in infections. The factor $\frac{\beta}{N} SI$ models the interaction between S and I and it is called the *incidence*, interpretable as a mass action law. The incidence makes the model nonlinear. The parameter β, which is often mantained constant or varied seasonally, changes according to the so called *NPIs* (Non-Pharmaceutical Interventions). All policies aiming at restricting contacts, *e.g.* social distancing measures, result in altering its value (β decreases if restrictions become stricter). The parameter γ is dimensionally the inverse of a time $[\frac{1}{T}]$ and represents the removal rate from the infective class by recovery. It can be thought as the inverse of the recovery time.

Manipulating algebraically Equation (2.8), we recover the following relationship between the three compartments:

$$S(t) + I(t) + R(t) = N \Rightarrow R(t) = N - S(t) - I(t). \tag{2.10}$$

Then, the system of equation (2.9) can be reduced to a system of two independent equations,

$$\begin{cases} \dot{S} = -\dfrac{\beta}{N} S\, I, & \forall t \in (0; T], \\ \dot{I} = \dfrac{\beta}{N} S\, I - \gamma I, & \forall t \in (0; T]. \end{cases} \tag{2.11}$$

Equilibrium states for the system are $(\bar{S}, 0)$ for any $\bar{S}$. The Jacobian matrix associated with the system (2.11) is

$$\begin{bmatrix} -\dfrac{\beta}{N} I & -\dfrac{\beta}{N} S \\ \dfrac{\beta}{N} I & \dfrac{\beta}{N} S - \gamma \end{bmatrix}, \tag{2.12}$$

and it is a range-1 matrix when it is computed at the equilibrium point. Hence, Linearization Method for the qualitative assessment of equilibria's stability fails.

The first equation in (2.11) entails the monotone-decreasing character of the susceptibles, since $\dot{S}$ is always negative. The variable $R(t)$ is always increasing, since $\dot{R}(t) = \gamma I \geq 0$. On the other hand, the infectious state can display both growing or decaying behaviour at the initial time, depending on a suitably-defined parameter known in literature as the *basic reproduction number* $\mathcal{R}_0$.
Indeed, if

$$0 > \frac{\beta}{N} S_0 - \alpha > \frac{\beta}{N} S(t) - \alpha, \tag{2.13}$$

the infective curve is decreasing for each time and always positive. If instead

$$\frac{\beta}{N} S_0 - \alpha > 0, \tag{2.14}$$

infected class grows at least in a small right neighbourhood of the initial time, achieving a maximum value. The non-dimensional parameter $\mathcal{R}_0$, accounting for the mean of new infected individuals caused by one infectious, is defined as

$$\mathcal{R}_0 = \frac{\beta S_0}{\gamma N} \tag{2.15}$$

for the `SIR` model. Mathematically, $\mathcal{R}_0$ plays the role of a threshold parameter for the dynamics of the ODE-system and it is related to equilibria's stability. In particular,

- $\mathcal{R}_0 < 1$ means strictly decaying behaviour: the epidemic does not outbreak;
- $\mathcal{R}_0 > 1$ implies infected start to grow: the infectious peak is reached, thus the epidemic outbreaks.

We can prove that SIR model always entails a complete eradication of the disease (*i.e.* $I(t)$ vanishes in the limit as $t \to \infty$) starting from whatever initial conditions and with fixed model parameters. Indeed, Equations (2.11) yield:

$$\frac{d}{dt}(S+I)(t) = -\gamma I \;\Rightarrow\; S(t)+I(t) = S_0 + I_0 - \gamma \int_0^T I(\tau)d\tau. \tag{2.16}$$

Defining $y(t) = \int_0^t I(\tau)d\tau$, $\dot{I}(t) = y'(t)$, $y(0) = 0$, (2.16) becomes

$$\begin{gathered} y'(t) + \gamma\, y(t) = S_0 + I_0 - S(t) \le S_0 + I_0 \Rightarrow \\ \Rightarrow e^{-\gamma t}\frac{d}{dt}(y(t)e^{\gamma t}) \le S_0 + I_0 \Rightarrow \\ \Rightarrow \frac{d}{dt}(y(t)e^{\gamma t}) \le (S_0 + I_0)e^{\gamma t} \Rightarrow \\ \Rightarrow y(t)\, e^{\gamma t} - y(0) \le (S_0 + I_0)\frac{e^{\gamma t}-1}{\gamma} \Rightarrow \\ \Rightarrow \int_0^\infty I(\tau)d\tau \le \int_0^t I(\tau)d\tau = y(t) \le \frac{S_0+I_0}{\gamma} - \frac{S_0+I_0}{\gamma}e^{-\gamma t} \le \frac{S_0+I_0}{\gamma}. \end{gathered} \tag{2.17}$$

Since $y(t)$ is bounded independently on t and

$$\dot{I}(t) = -\gamma I(t) + \frac{\beta}{N}S(t)I(t) \le \frac{\beta}{N}S(t)I(t) \le \beta N, \tag{2.18}$$

we apply Theorem in Appendix B, and deduce that

$$\lim_{t\to\infty} I(t) = 0. \tag{2.19}$$

All previous computations are valid if it is possible to verify that $I(t)$ is defined $\forall t \in [0;T]$. For this purpose, we define the following change of variable:

$$\tau = \int_0^t I(s)ds. \tag{2.20}$$

Since $I(t)$ is always positive, $\tau'(t) = I(t) \ge 0$ and the change of variable is invertible. Hence,

$$d\tau = I(t)dt \;\Rightarrow\; \frac{d}{d\tau} = \frac{1}{I}\frac{d}{dt}, \tag{2.21}$$

and the ODEs system (2.11) can be rewritten as follows

$$\begin{cases} \dfrac{dS}{d\tau} = -\dfrac{\beta}{N}S, \\ \dfrac{dI}{d\tau} = \dfrac{\beta}{N}S - \gamma, \end{cases} \tag{2.22}$$

which is a linear system and its solution is always defined.

Finally, we can integrate the system in order to get an explicit solution for the `SIR` case. Indeed,

$$\dot{S} = -\frac{\beta}{N} S\, I \Rightarrow \log\left(\frac{S}{S_0}\right) = -\frac{\beta}{N}\int_0^t I(s)ds \underset{(2.16)}{=} \frac{\beta}{N\,\gamma}(N - S - I). \tag{2.23}$$

The relationship among infectives and susceptibles is

$$I = N - S - N\frac{\gamma}{\beta}\log\left(\frac{S}{S_0}\right). \tag{2.24}$$

Substituting (2.24) in the first equation of (2.11) and integrating the resulting equation we recover an explicit solution for $S(t), I(t)$ and $R(t)$.

2.2.2 `SIS` model

`SIR` model is suitable for diseases that confer immunity after infection has happened. This is not always the case: for instance cold, influenza and even SARS-CoV-2 do not grant immunity after recovering from the infection. The possibility of some diseases to confer immunity or not is strictly related to the antibodies (proteins that our body produces automatically to respond to the infection). Antibodies coat invading cells and, in the best case, prevent invaders from hijacking cells and replicating. Level of antibodies rapidly decreases when the infection is passed. Reinfections occur if the disease has developed resistance and has genetically changed (this is the case of the outcome of variants) or antibodies level is too low to fight against a new contact with pathogenon agents.

The `SIS` model tries to catch this peculiarity by deleting the removed class and allowing susceptible individuals to return to S class after the infection. In Figure 2.2.b the schematic flowchart for the `SIS` model is showed. The system of equations governing `SIS` is the following:

$$\begin{cases} \dot{S} = -\dfrac{\beta}{N} S\, I + \gamma I, & \forall t \in (0; T], \\ \dot{I} = \dfrac{\beta}{N} S\, I - \gamma I, & \forall t \in (0; T]. \end{cases} \tag{2.25}$$

This model can be reduced to a one-equation dynamical system. Indeed,

$$\dot{S} + \dot{I} = 0 \;\Rightarrow S + I = S_0 + I_0 = N \Rightarrow \tag{2.26}$$

$$\dot{I} = \frac{\beta}{N}(N - I)\, I - \gamma I = \left(\beta - \frac{\beta I}{N} - \gamma\right) I. \tag{2.27}$$

Particularly, the continuous dynamical system associated with (2.27) is a nonlinear dynamical system, also known as the *logistic* system where the evolutive dynamics is a parabolic function.

Two distinct equilibria can be found by solving $\dot{I} = 0$. Indeed,

$$\left(\beta - \frac{\beta I}{N} - \gamma\right) I = 0 \Rightarrow \bar{I}_1 = 0 \vee \bar{I}_2 = \frac{(\beta - \gamma)N}{\beta}. \tag{2.28}$$

The nontrivial equilibrium $\bar{I}_2$ can be rewritten as

$$\bar{I}_2 = \left(1 - \frac{1}{\mathcal{R}_0}\right) N, \tag{2.29}$$

after defining the basic reproduction number for the `SIS` model as

$$\mathcal{R}_0 = \frac{\beta}{\gamma}. \tag{2.30}$$

Depending on the threshold value $\mathcal{R}_0 = 1$, equilibria have different stability properties:

- $\mathcal{R}_0 > 1$ means $\bar{I}_2 > 0$ is asymptotically stable, $\bar{I}_1 = 0$ is unstable;
- $\mathcal{R}_0 < 1$ means $\bar{I}_2 < 0$ is unstable, $\bar{I}_1 = 0$ is asymptotically stable.

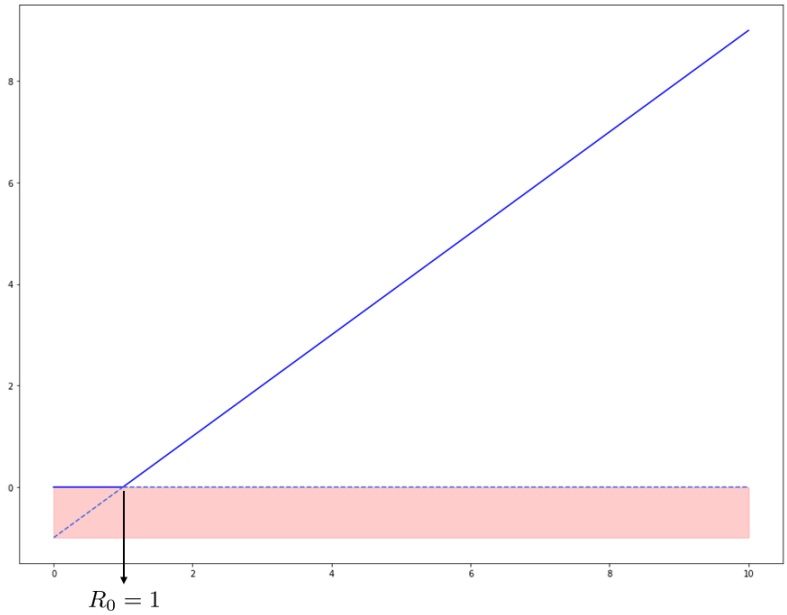


Figure 2.3: Bifurcation diagram for the `SIS` model.

For this system, the bifurcation diagram based on stability of the equilibria is drawn in Figure 2.3. We vary $\mathcal{R}_0$ on the x-axis and plot qualitatively the equilibrium points with blue curves. In the pink region ($I(t) < 0$) equilibria are not admissible. We notice the onset of a transcritical bifurcation point whether $\mathcal{R}_0 = 1$. Actually, if $\mathcal{R}_0 < 1$ the system admits one equilibrium ($\bar{I}_2$ is negative, hence not admissible) which is asymptotically stable ($\bar{I}_1$). When $\mathcal{R}_0 > 1$, $\bar{I}_2$ is asymptotically stable and therefore infectious do not vanish but every solution collapses to a stable point with non-zero infected individuals. In this case the disease will not be extincted.

2.2.3 Vital dynamics

Models introduced in the previous sections cannot describe situations extended over long periods of time. According to [55] a disease is called *endemic* if it is persistent for more than almost 10 years. When endemic diseases are concerned, births and deaths not caused by the infections have to be included in the model (the so-called *vital dynamics*).

One possible linear correction to model (2.9) is the following [81]:

$$\begin{cases} \dot{S} = -\dfrac{\beta}{N} S\, I + \Lambda - \mu S, & \forall t \in (0;T], \\ \dot{I} = \dfrac{\beta}{N} S\, I - \gamma I - \mu I, & \forall t \in (0;T], \\ \dot{R} = \gamma I - \mu R, & \forall t \in (0;T], \end{cases} \tag{2.31}$$

where Λ represents the amount of daily births and μ the average death rate. We assume that Λ is constant in time, even if this is not completely realistic. Indeed, births strictly depend on different factors (*e.g.* availability of resources, age of the individuals, sex-class ripartitions) that often vary in time. The parameter μ can be interpreted as the inverse of the medium living time of the species involved. Increasing the death rate μ can be seen as a reduction on the average life expectancy. Notice that deaths ruled by μ are completely uncorrelated to the disease development: we are under the hypothesis that the disease is not letal.

The two equilibria corresponding to system (2.31) are $(\frac{\Lambda}{\mu}, 0, 0)$, called disease-free equilibrium, and $(\frac{\gamma+\mu}{\beta}N, \frac{\Lambda-\mu\bar{S}}{\gamma+\mu}, \frac{\gamma}{\mu}\bar{I})$, endemic equilibrium.
If $N = 1$ (*i.e.* all state variables are percentages on the total population) and $\Lambda = \mu$ the following theorem can be proven [55]:

Theorem 5. *Define* $\mathcal{R}_0 = \frac{\beta}{\gamma+\mu}$ *and let* $\mathcal{R}_0 \leq 1$. *Then the triangle*

$$T = \{(S, I) | S \geq 0, I \geq 0, S + I \leq 1\}$$

is an asymptotic stability region for the disease-free equilibrium. On the other hand, if $\mathcal{R}_0 > 1$ *then*

$$T - \{(S, 0) | 0 \leq S \leq 1\}$$

is an asymptotically stable region for the endemic equilibrium.

Even in this case the basic reproduction number acts like a bifurcation parameter for the stability properties of the two equilibria, as claimed by the previous theorem. Indeed, if the reproduction number is less than one than the disease dies out since one infective replaces himself with less than one new infective. Instead, if $\mathcal{R}_0$ is bigger than one it is possible to have an epidemic outbreak with solutions tending to the endemic equilibrium.

2.2.4 `SEIR` and `SEIRD` models

In this section the last paradigmatic epidemic models are shortly presented, namely `SEIR` [76] and `SEIRD` [81] models. `SEIR` is constituted by four averaged classes where $\boldsymbol{S}$ represents the class of susceptibles, $\boldsymbol{R}$ the removed one, $\boldsymbol{E}$ the class of *infected* but *not infectious* individuals and $\boldsymbol{I}$ the class of individuals contemporarily *infectious* and *infected*. The resulting system of ODEs is:

$$\begin{cases} \dot{S} = -\dfrac{\beta}{N} S\, I + \Lambda - \mu S, & \forall t \in (0;T], \\ \dot{E} = \dfrac{\beta}{N} S\, I - (\alpha+\mu)E, & \forall t \in (0;T], \\ \dot{I} = \alpha E - (\gamma+\mu)I, & \forall t \in (0;T], \\ \dot{R} = \gamma I - \mu R, & \forall t \in (0;T], \end{cases} \tag{2.32}$$

where γ (which is dimensionally $[\frac{1}{T}]$) represents the recovery rate, whereas α (which is also dimensionally the inverse of a time) can be physically interpreted as the inverse of the average incubation time that strictly depends on the nature of the disease.
Imposing vanishing derivative in (2.32), we recover the equilibria of the associated continuous dynamical system:

$$\begin{cases} 0 = -\dfrac{\beta}{N} S\, I + \Lambda - \mu S, \\ 0 = \dfrac{\beta}{N} S\, I - (\alpha+\mu)E, \\ 0 = \alpha E - (\gamma+\mu)I, \\ 0 = \gamma I - \mu R. \end{cases} \tag{2.33}$$

The equilibrium with infectious extinction is $(\frac{\Lambda}{\mu}, 0, 0, 0)$. The endemic equilibrium (*i.e.* the one with non-vanishing infectious) can be deduced solving the third equation in E,

$$E = \frac{\gamma+\mu}{\alpha} I, \tag{2.34}$$

and, susbstituting it in the second equation of (2.33) yields

$$S = \frac{(\mu+\gamma)(\mu+\alpha)N}{\beta\alpha}. \tag{2.35}$$

From the first equation of (2.33), we get

$$I = \frac{\Lambda N}{\beta S} - \frac{\mu N}{\beta} = \frac{\Lambda\alpha}{(\mu+\gamma)(\mu+\alpha)} - \frac{\mu N}{\beta} = \frac{\mu N}{\beta}(\mathcal{R}_0 - 1), \tag{2.36}$$

where $\mathcal{R}_0$ is defined as follows

$$\mathcal{R}_0 = \frac{\Lambda}{N}\frac{\alpha\beta}{\mu(\mu+\gamma)(\mu+\alpha)}. \tag{2.37}$$

The reproduction number is overall positive, and it is zero when the trasmission rate β is zero. The factor $\frac{\Lambda\beta}{\mu(\mu+\gamma)}$ can be seen as the number of secondary cases produced by one infectious individual during its lifespan as infectious, whilst $\frac{\alpha}{\alpha+\mu}$ is the fraction of exposed individuals that encompasses the exposed period and becomes actually infectious. Applying Linearization Method to the Jacobian computed at the endemic equilibrium, and via the *Hurwitz* criterium for assessing signs of eigenvalues [81], the following proposition can be proven [55]:

Proposition 1. *Assume $\mathcal{R}_0 > 1$. Then the endemic equilibrium previously defined is locally asymptotically stable.*

The `SEIRD` model is a modification of `SEIR` model and it is employed in those cases where the virus can lead to death. It is worth noticing that introducing the compartment of deaths is particularly helpful during the calibration stage for parameters involved in epidemic models. Indeed, the number of deaths is typically less affected by uncertainty than the other compartments, since recognizing deaths caused by the disease is easier than detecting all people with ongoing disease. The model (excluding vital dynamics) reads as follows:

$$\begin{cases} \dot{S} = -\dfrac{\beta}{N} S\, I, & \forall t \in (0;T], \\ \dot{E} = \dfrac{\beta}{N} S\, I - \alpha E, & \forall t \in (0;T], \\ \dot{I} = \alpha E - \gamma I, & \forall t \in (0;T], \\ \dot{R} = (1-f)\gamma I, & \forall t \in (0;T], \\ \dot{D} = f\gamma I, & \forall t \in (0;T]. \end{cases} \tag{2.38}$$

Class $\boldsymbol{D}$ counts the amount of deads due to the disease. The parameter f is instead the fatality rate, and depends on the specific disease.

2.2.5 Compartmental models including medical and social interventions

Complex epidemic situations are commonly faced through the implementation of political and social interventions or through medical treatments as well. In endemic situations it is more probable that time will be sufficient to develop *ad hoc* effective medical interventions, such as vaccines. Tipically, social and medical interventions come at different prices, *i.e.* they have social or economic costs that policy makers or sanitarian competencies have to take into account.

Medical and social interventions involve three main aspects:

1. Quarantine or lockdowns at different levels (often referred to as *NPIs*, Non-Pharmaceutical Interventions);

2. Vaccinations;

3. Treatments.

Quarantine. When diseases spread, a possible containment intervention is to impose prior restrictions on circulation, limiting contacts among people. Etimologically, the term *Quarantine* derives from seventeenth-century Venetian *quarantena*, standing for forty days of strict isolation imposed during the outbreak of Venice-plague in 1630. Isolation is commonly applied to those individuals who have had contacts with infectives and so who could foster the spread of the disease.
Define the class of quarantined/isolated individuals as Q and the Active population $A(t) = N - Q(t) = S(t) + I(t) + R(t)$ as in [39]. A possible compartmental model including quarantine is the following:

$$\begin{cases} \dot{S} = -\dfrac{\beta}{A} S\, I, & \forall t \in (0;T], \\ \dot{I} = \dfrac{\beta}{A} S\, I - (\gamma + \eta) I, & \forall t \in (0;T], \\ \dot{Q} = \eta I - \kappa Q, & \forall t \in (0;T], \\ \dot{R} = \gamma I + \kappa Q, & \forall t \in (0;T], \end{cases} \tag{2.39}$$

where η is called isolation rate ($[\frac{1}{T}]$) and it is strictly linked to monitorning policies that are implemented (*i.e.* to the level of testing). The parameter κ instead ($[\frac{1}{T}]$) is the inverse of the average time that people spend in the quarantined state. Incrementing *NPIs* level, for example introducing local or global lockdowns, can also be modelled through an adequate decreasing of the contact rate β.

Vaccination. Vaccination is the administration of weakened or dead antigenic material in order to develop automatic immunitary response in individuals' body. In this way it is possible to achieve partial or complete immunity without recurring to infections. Generally, there are two different ways for modelling vaccinations.

One possible choice is to define a proportion p of initial susceptibles that are directly moved into the recovered class, whereas the remaining $(1 - p)$ fraction flows in S. The fraction p is the percentage of vaccination administered. This first method is implemented when all vaccinations happen at the beginning of the simulation and it prescribes information about necessary levels of vaccinations for eradicating the disease.

The other way, which we are more accustomed to (see, *e.g.*, [46], [106], [100], [78]) and which corresponds to daily vaccine administrations, is to subtract continuously a

part of vaccinated people from the S class. Each subtracted fraction of susceptible individuals populate a new class (commonly called V), where immunized individuals end up:

$$\begin{cases} \dot{S} = -\dfrac{\beta}{N} S\, I - \mu_S S, & \forall t \in (0;T], \\ \dot{I} = \dfrac{\beta}{N} S\, I - \gamma I, & \forall t \in (0;T], \\ \dot{R} = \gamma I, & \forall t \in (0;T], \\ \dot{Q} = \mu_S S; & \forall t \in (0;T]. \end{cases} \tag{2.40}$$

This is the strategy we adopted in the `SEIHRDVW` model described in Chapter 4 for formulating Optimal Control Problems.

Medical treatments. An infectious disease's treatment refers to the care given to decrease morbidity and death. Medications that alleviate symptoms and aid the immune system in fighting the disease are common examples. Treatments can be directly englobed in the `SEIR` model (2.32) through the introduction of a specific new stage, which is commonly identified by letter T. Define p as the probability of successful treatment, and q as the probability of failure, such as $p + q = 1$. The following `SEIT` model has been introduced for modelling tubercolosis in [38]:

$$\begin{cases} \dot{S} = -\dfrac{\beta_1}{N} S\, I + \Lambda - \mu S, & \forall t \in (0;T], \\ \dot{E} = \dfrac{\beta_1}{N} S\, I + \dfrac{\beta_2}{N} T\, I - (\alpha + r_1 + \mu)E + p r_2 I, & \forall t \in (0;T], \\ \dot{I} = \alpha E - (r_2 + \mu)I, & \forall t \in (0;T], \\ \dot{T} = r_1 E + q r_2 I - \dfrac{\beta_2}{N} T\, I - \mu T, & \forall t \in (0;T], \end{cases} \tag{2.41}$$

where r_1 is the treatment rate of exposed individuals, r_2 the same rate for infectious individuals. They are differentiated since care treatments are often proportional to the progress of the disease.

2.2.6 Techniques for computing the basic reproduction number

Previous models represent baselines for more complex flowcharts, with more compartments and transitions. As far as model's dimension grows it is important to develop adequate tools in order to compute some dimensionless parameters as the basic reproduction number $\mathcal{R}_0$ and its time-dependant homologous $\mathcal{R}_t$ to support the analysis of the results. Two methods for computing $\mathcal{R}_0$ are dealt in this section following [81].

The reproduction number must respect the following properties:

- be nonnegative for nonnegative parameters;
- be zero when trasmission spread is null;
- be interpretable as the number of secondary infections.

Jacobian approach Mathematically, the parameter $\mathcal{R}_0$ gives a threshold condition for assessing the outbreak of the pandemic and the stability of the disease-free equilibrium. Imposing conditions for stability of the non-endemic equilibrium leads to a possible definition of the reproduction number.

As we have seen in Section 2.1, Linearization Method provides a sufficient stability condition through the analysis of the eigenvalues. In particular the Jacobian of the system can be computed at the disease-free equilibrium, and the eigenvalues' real part has to be strictly negative whenever equilibrium is stable. Imposing the aforementioned conditions in order to recover inequalities on the parameters leads to the definition of the basic reproduction number: this method is known in literature as the *Jacobian approach*. In the two-dimensional case, this requirement follows from the conditions $\mathrm{Tr}J < 0$ and $\mathrm{Det}J > 0$, which represent respectively the sum and the product of the two eigenvalues of a 2×2 matrix. In higher dimensional cases, if it is possible to recover a 2×2-form of the Jacobian, the same inequalities can be applied. If this is not possible, the characteristic polynomial associated to the Jacobian has degree higher than 3 and the previous inequalities do not hold anymore. Necessary and sufficient conditions regarding signs of the eigenvalues are given by the **Routh-Hurwitz criterion**, stated in the following theorem [33]:

Theorem 6. *Consider the nth-degree polynomial with real constant coefficients*

$$P(\lambda) = \lambda^n + a_1\lambda^{n-1} + ... + a_{n-1}\lambda + a_n.$$

Define n Hurwitz matrices in the following way:

$$H_1 = \begin{bmatrix} a_1 \end{bmatrix} \quad H_2 = \begin{bmatrix} a_1 & 1 \\ a_3 & a_2 \end{bmatrix} \quad H_3 = \begin{bmatrix} a_1 & 1 & 0 \\ a_3 & a_2 & a_1 \\ a_5 & a_4 & a_3 \end{bmatrix}$$

and

$$H_n = \begin{bmatrix} a_1 & 1 & 0 & 0 & \dots & 0 \\ a_3 & a_2 & a_1 & 1 & \dots & 0 \\ a_5 & a_4 & a_3 & a_2 & \dots & 0 \\ a_7 & a_6 & a_5 & a_4 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & 0 & \dots & a_n \end{bmatrix}$$

where $a_j = 0$ if $j > n$. All roots of the polynomial $P(\lambda)$ are negative or have negative real part if and only if the determinants of all Hurwitz matrices are positive:

$$\mathrm{Det}H_j > 0, \quad j = \dots, n.$$

The Jacobian approach provides practical implications if all conditions coming out from Hurwitz criterium can be reduced to one single inequality. This is not always the case, especially in models combining host heterogeneities, *e.g.* when susceptibles can be differentiated on the base of different demographic characteristic or on the base of different inclinations with respect to the disease as in multi-age models.

Next Generation Matrix approach The basic reproduction number $\mathcal{R}_0$ is referred to the possibility of giving birth to new infected individuals. In this sense the onset of an epidemic outbreak can be qualified as a demographic process with consecutive generations of infected. If the sizes of subsequent generations grow, the offspring of an epidemic occur. Hence, $\mathcal{R}_0$ is then the characterizing factor per generation that corresponds to a potential infected growth. For demographic processes that are categorized in discrete compartments, a common approach is to define a matrix for relating infected individuals in temporary consecutive generations. This matrix known as *Next Generation Matrix*, NGM, was firstly introduced by Diekmann and Heersterbeek in 1990 [35] and its spectral radius is the parameter $\mathcal{R}_0$. We consider the *Van de Driessche and Watmough* approach [113] as a possible method to assemble the NGM.

More precisely, we divide compartments in two main groups, *infected* and *noninfected*, depending on whether they are constituted by infected or healthy individuals. Assume that there are $n \in \mathbb{N}$ infected compartments and $m \in \mathbb{N}$ healthy classes so that the entire amounts of compartments is $m+n$. Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ be the vectors of variables of infected and healthy compartments. Then, proceed with the following steps:

1. Arrange the equations so that the first n components correspond to infected compartments. Thus, the system can be rewritten as

$$\begin{aligned} x_i' &= f_i(x,y), \quad i = 1,\dots,n, \\ y_j' &= g_j(x,y), \quad j = 1,\dots,m; \end{aligned} \tag{2.42}$$

2. Split the right-hand side in the infected equations as

$$\begin{aligned} x_i' &= \mathcal{F}_i(x,y) - \mathcal{V}_i(x,y), \quad i = 1,\dots,n \\ y_j' &= g_j(x,y), \quad j = 1,\dots,m, \end{aligned} \tag{2.43}$$

   with $\mathcal{F}_i(x,y)$ representing the rate of appearance of new infections in i and $\mathcal{V}_i(x,y)$ the remaining flows. This decomposition is not unique, and it has to satisfy some properties:

   - $\mathcal{F}_i(0,y) = 0$ and $\mathcal{V}_i(0,y) = 0$ for $y \geq 0, i = 1,\dots,n$, so that all new infections are secondary infections from the infected host and there is no immigration of susceptibles into the disease compartments;
   - $\mathcal{F}_i(x,y) \geq 0$ for all $x, y \geq 0$;

- $\mathcal{V}_i(x,y) \leq 0$ whenever $x_i = 0$ for $i = 1, \ldots, n$;
- $\sum_{i=1}^{n} \mathcal{V}_i(x,y) \geq 0$ for all $x, y \geq 0$, meaning that the total outflow of all infected is positive.

3. Assume the disease free system

$$y' = g(0, y) \tag{2.44}$$

has a unique disease-free equilibrium $(0, y_0)$ and determine it.

4. Define the matrices F and V as

$$\begin{aligned} F_{ij} &= \left[\frac{\partial \mathcal{F}_i}{\partial x_j}(0, y_0)\right], \\ V_{ij} &= \left[\frac{\partial \mathcal{V}_i}{\partial x_j}(0, y_0)\right]. \end{aligned} \tag{2.45}$$

Those matrices arise from the linearization of the system around the equilibrium point.

5. The next-generation matrix is defined as

$$K = FV^{-1},$$

and

$$\mathcal{R}_0 = \rho(FV^{-1}).$$

Moreover, it can be shown that if $\mathcal{R}_0 < 1$ than the disease-free equilibrium is asymptotically stable.

2.3 Other epidemic models

Epidemic models are not always ruled by ODE-based dynamical systems. For instance, all the previous compartmental models rely on the hypotheses of spatial homogeneity and on neglecting age dependancy. In most cases those two factors come up and influence the dynamics or can be of paramount importance in order to get non-trivial and realistic predictions on the epidemics. In these cases, PDE-based models arise and have to be investigated within a proper mathematical framework. In the next subsections two PDE-based models are synthetically presented.

2.3.1 Spatial models with diffusion

In order to embody the spatial dependancy reaction-diffusion equations, which are second order partial differential equations, are widely adopted in modelling processes. In [22] an example of a diffusion-reaction model for one-dimensional spatial epidemic spread is introduced. Spatial heterogeneity is embodied in ODE-based models by adding diffusion terms to the standard time-dependant formulations. For example, consider an `SI` model with second order nonlinear trasmission, *i.e.*

$$\begin{aligned} S' &= -\frac{\beta}{N} SI, \\ I' &= \frac{\beta}{N} SI. \end{aligned} \tag{2.46}$$

Assume susceptibles and infected are homogeneously spatially distributed, $S = S(x,t)$ and $I = I(x,t)$, with $x \in \mathbb{R}$. By adding diffusion to model (2.46) a parabolic system of PDE arises:

$$\begin{aligned} \frac{\partial S}{\partial t} &= -\frac{\beta}{N} S(x,t) I(x,t) + D\frac{\partial^2 S}{\partial x^2}, \\ \frac{\partial I}{\partial t} &= \frac{\beta}{N} S(x,t) I(x,t) + D\frac{\partial^2 I}{\partial x^2}, \end{aligned} \tag{2.47}$$

where D is the diffusion constant. After defining the total population as $N(x,t) = S(x,t) + I(x,t)$, we verify that $N(x,t)$ satifies a random diffusion equation as follows:

$$\frac{\partial N}{\partial t} = D\frac{\partial^2 N}{\partial x^2}. \tag{2.48}$$

When PDE-based models are introduced two types of conditions have to be prescribed in order to guarantee well-posedness of the problem:

1. **Initial Conditions (IC)**. They give the complete spatial distribution of all variables at the initial time. They also have to be prescribed in ODE-based Cauchy problems.

2. **Boundary conditions (BC)**. They are conditions of different kind which are imposed at the borders of the spatial domain for each time. We can impose different types of BC:

   - **Dirichlet-like** if at the two boundaries time-depending distributions are fixed (if the domain is one-dimensional and finite, otherwise a trace operator on the domain has to be defined);
   - **Neumann-like** if $\frac{\partial V}{\partial x}$ is fixed at the boundaries;

- **Robin-like** when a linear combination of the values and the fluxes of the unknown function is prescribed at the boundaries ($\alpha_1 \frac{\partial V}{\partial x} + \alpha_2 V$).

Neumann and Robin boundary conditions are burdensome to be determined. For instance coming back to epidemic models and considering a monodimensional domain, for both conditions one needs to extimate entering and exiting people, hence a demanding monitoring policy has to be taken into account. Moreover, Robin conditions are even tougher to be dealt since we need to monitor contemporarily individuals belonging to each compartment at the boundary and the fluxes across the borders at each time. Homogeneous Neumann conditions correspond to the case of closed and non-accessible domains. Homogeneous Robin conditions correspond to the case of proportionality between individuals at the boundary and the balance of the fluxes across borders of the domain (the so-called *Hook effect*).

2.3.2 Age-structured model

Age-dependance is a fundamental factor for those models referring to diseases that behave differently with different age-classes, similarly to COVID-19. In this respect, the ISS (Istituto Superiore di Sanità) in its report on 21st April 2021[2] extimates that almost 1.1% of deceased individuals due to SARS-CoV-2 in Italy is under 50 and that the medium age of deceased is about 75 (Figure 2.4).

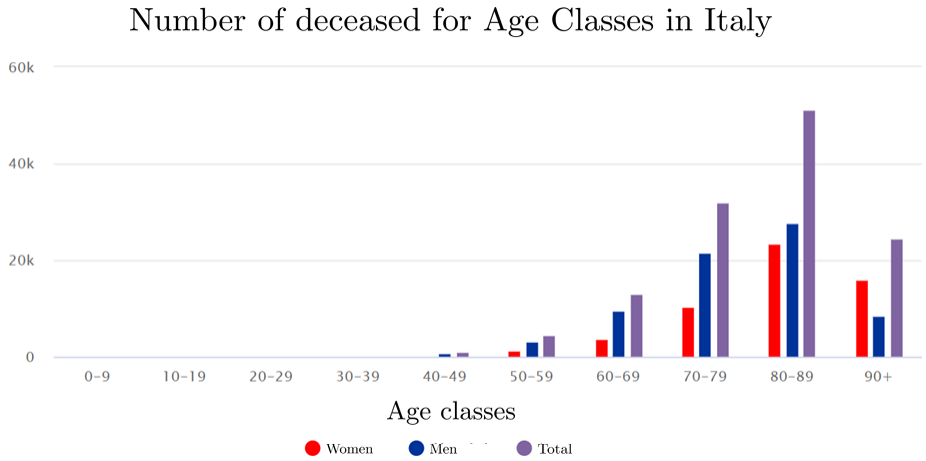


Figure 2.4: Barplot of deceased individuals for SARS-CoV-2 infection in Italy split in age-classes. Source: ISS Report, 21st Aprile 2021.

In the sequel we briefly describe a possible age-structured compartmental model [62]. Define $s(t,a)$, $i(t,a)$ and $r(t,a)$ as the number of susceptibles, infectious and

[2] `https://www.epicentro.iss.it/coronavirus/sars-cov-2-decessi-italia`

removed individuals at time t with age a, respectively. Then, the following holds

$$s(t+\epsilon, a+\epsilon) = s(t,a) - \left[\mu(a) + \int_0^{+\infty} k(t,a,\zeta) i(t,\zeta) d\zeta \right] s(t,a)\epsilon, \tag{2.49}$$

namely, the amount of susceptibles at time $t+\epsilon$ whose age is $a+\epsilon$ are those who were susceptibles at time t of age a, minus the deceased ones during the time-interval of length ϵ (*i.e.* the factor $\mu(a)s(t,a)\epsilon$) and those who have been infected in the interval $(t; t+\epsilon)$. The incidence term $(\int_0^{+\infty} k(t,a,\zeta) i(t,\zeta) d\zeta \; s(t,a)\epsilon)$ takes into account for interactions among $s(t,a)$ with infected of all ages through the continuous function of three variables $k(t,a,\zeta)$, to be properly prescribed.

Considering (2.49), dividing both members by ϵ and taking the limit as $\epsilon \to 0$, we obtain

$$\frac{\partial s}{\partial a}(t,a) + \frac{\partial s}{\partial t}(t,a) = -\mu(a)s(t,a) - s(t,a) \int_0^{+\infty} k(t,a,\zeta) i(t,\zeta) d\zeta. \tag{2.50}$$

With the same proceedings, we obtain the PDEs governing the evolution of $i(t,a)$ and $r(t,a)$:

$$\begin{aligned} &\frac{\partial i}{\partial a}(t,a) + \frac{\partial i}{\partial t}(t,a) = -\mu(a) i(t,a) + s(t,a) \int_0^{+\infty} k(t,a,\zeta) i(t,\zeta) d\zeta - \alpha\, i(t,a), \\ &\frac{\partial r}{\partial a}(t,a) + \frac{\partial i}{\partial t}(t,a) = -\mu(a) r(t,a) + \alpha\, i(t,a), \end{aligned} \tag{2.51}$$

where α is the so-called recovery parameter.

Define $n(t,a) = s(t,a) + i(t,a) + r(t,a)$ as the number of individuals of age a at time t. Summing Equations (2.50) and (2.51), we can verify that

$$\frac{\partial n}{\partial t}(t,a) + \frac{\partial n}{\partial a}(t,a) = -\mu(a) n(t,a), \tag{2.52}$$

which is known in literature as the *Von Foerster's equation*, a classical equation for age-structured models in population dynamics [52].

Eventually, PDE-based models need initial conditions (*e.g.* $s(0,a)$) and birth distribution for each time (*e.g.* $s(t,0)$) which is not often known since it is strictly dependant on the populosity of the classes themselves. To overcome this issue, it is possible to define $s(t,0) = \int_0^{+\infty} \beta_s(\zeta) s(t,\zeta) d\zeta$ where $\beta_s(a)$ represents the birth rate distribution for individuals of age a, which can be deduced avaraging available historical data. Notice that in the definition of births evolution the unknown itself is present.

2.4 Epidemic models for SARS-CoV-2 pandemic

Many epidemic models dealing with SARS-CoV-2 pandemic have been developed during the last two years. Some of the recent studies differ in findings and results mainly because they strictly depend on the assumptions made and on the quality of data upon which models are calibrated. In the context of SARS-CoV-2 pandemic, the calibration process is exacerbated by the lack of detection of infectious individuals, who often are asymptomatical infectious. Testing reporting delays and poor detection affect validity of the output of mathematical models.

A key-issue we need to take into account in the context of COVID-19 is the choice of the epidemic model. Indeed, simple models often provide less valid forecasts since they do not capture human mixting patterns and other characteristics of infectious diffusion. On the other hand, complex models may create the illusion of realism even when they miss key-aspects of the biology of the phenomenon, making it harder to spot crucial omissions. For instance, they are more sensitive to changes in parametric assumptions than simpler models.

Predictive models for COVID-19 are not reliable in large areas, because they aggregate and average heterogenous subepidemics in local areas. Moreover, averaged-classes compartmental models do not take into account for individual factors, such as comorbidities or age, which influence the risk of severe disease from COVID-19.

Referring to the actual state of the art, some models have tried to keep it tight on standard compartmental models (see, *e.g.*, [24], [99], [49], [54], [20], [26], [30], [2] and [109]). Some others have tried to be fully consistent with available data in different countries (see, *e.g.*, [96], [47], [120], [40], [111] and [71]). Other models have incorporated the geospatial dependancy and some others the randomness on parameters that always depend on testing policies, economic investments and other factors, formulating new peculiar PDE-based models (see, *e.g.*, [115], [79] and [11], which distinguishes among scenarios in which diffusion is prominent or not, leading to new hyperbolic and parabolic models).

Some works have focused on the possibility of building new descriptors along with $\mathcal{R}_0$ parameter which can be more robust in the prevision of epidemic peaks (see, *e.g.*, [80] and [12]). Finally, some studies have successfully used Artificial Intelligence, specifically Machine Learning Algorithms such as Neural Networks, in order to forecast COVID-19 diffusion and to develop completely data-driven approaches (see, *e.g.*, [60], [93], [70], [85], [3] and [87]). In particular, in [93] an Artificial Neural Network has been implemented through the use of Johns Hopkins dashboard data for predicting the future reachability of COVID-19 across India.

Let us synthetize some peculiarities of paramount importance for catching the complex phenomenon of the spread of the disease, especially in Italy:

- monitoring and testing policies are often unable to prevent the existence of undetected asymptomatic individuals which are the main responsible for the epidemic spread;

- the spread of the disease is persistent in time. Other epidemics, *e.g.* the plague of Eyam in 1665-1666 [118], were confined in time and space and so the hypotheses for applying standard models (such as the `SIR`) are consistent. This is not our case;
- SARS-CoV-2 can cause severe effects which eventually lead to death, and a considerable amount of infected individuals needs to be hospitalized, sometimes recurring to ICUs (Intensive Care Units). ICUs have been over strain during previous waves characterizing the pandemic, and so it is important to contain the spread in order to minimize their application;
- infection response differs heterogeneously depending on the infected age and comorbidities. Multi-age models are able to catch this difference among classes even though their formulation is computationally demanding (see, *e.g.*, [67] and [121]);
- if one considers global scenarios at national levels, one neglects subepidemics which actually rule the phenomenon. On the other hand, if confined areas are considered, entering and outing fluxes have to be properly modeled.

2.4.1 `SUIHTER` model

In this section it is briefly described the `SUIHTER` model [89] proposed in January 2021 by the Laboratory of Mathematical Modelling and Scientific Computing (MOX) of Politecnico di Milano which tries to overcome some of the already mentioned limitations through a model completely conceived with Italian data available from the Italian department of Protezione Civile [91]. As it has just been said, one of the major limits of COVID-19 models is the difficulty in taking into account for undetected individuals, which are virtually impossible to catch. This model provides specific classes in order to extimate the amount of undetected individuals circulating in the area of interest.

The `SUITHER` formulation reads as follows:

$$\begin{cases} \dot{S}(t) = -S(t)\dfrac{\beta_U U(t) + \beta_I I(t) + \beta_H H(t)}{N} \\ \dot{U}(t) = S(t)\dfrac{\beta_U U(t) + \beta_I I(t) + \beta_H H(t)}{N} - (\delta + \rho_U)U(t) \\ \dot{I}(t) = \delta U(t) - (\rho_I + \omega_I + \gamma_I)I(t) + \theta_H H(t) \\ \dot{H}(t) = \omega_I I(t) - (\rho_H + \omega_H + \theta_H + \gamma_H)H(t) + \theta_T T(t) \\ \dot{T}(t) = \omega_H H(t) - (\theta_T + \gamma_T)T(t) \\ \dot{E}(t) = \gamma_I I(t) + \gamma_H H(t) + \gamma_T T(t) \\ \dot{R}(t) = \rho_U U(t) + \rho_I I(t) + \rho_H H(t) \end{cases} \tag{2.53}$$

where the compartmental letters stand for

- S: susceptible individuals;
- U: undetected (symptomatic or not) individuals;
- I: infected individuals isolated at home;
- H: hospitalized individuals;
- T: threatened individuals hosted in ICUs;
- E: extinct/deceased individuals;
- R: recovered individuals.

The total population amounts at $N = S + U + I + H + T + E + R$. This model is described by 14 parameters which can be interpreted as follows:

- $\beta_U, \beta_I, \beta_H$ are the transmission rates due to contacts among the referring class and susceptible individuals;
- ω_I denotes the rate at which individuals develop clinically relevant symptomps;
- ω_H the rate at which hospitalized individuals develop life-threatening symptomps;
- θ_T, θ_H are the improval rates of conditions from T and H class respectively;
- δ is the probability rate of detection;
- ρ_U, ρ_I and ρ_H are the recovery rates for each class;
- γ_I, γ_H and γ_T are the mortality rates.

For this model, $\mathcal{R}_0$ can be computed through the method of the Next Generation Matrix, yielding

$$\mathcal{R}_0 = \frac{\beta_U}{r_1} + \frac{\delta}{r_1}\left(\frac{\beta_I(r_3 r_4 - \theta_T \omega_H) + \beta_H \omega_I r_4}{r_2 r_3 r_4 - r_4 \theta_H \omega_I - r_2 \theta_T \omega_H}\right), \tag{2.54}$$

where $r_1 = \delta + \rho_U$, $r_2 = \rho_I + \omega_I + \gamma_I$, $r_3 = \rho_H + \omega_H + \theta_H + \gamma_H$, and $r_4 = \theta_T + \gamma_T$.

The European COVID-19 Forecast Hub[3] collects previsions made by mathematical models for COVID-19 implemented worldwide and provides rankings about their accuracy in predicting the evolution of the pandemic compared to data. In the week of 3rd to 9th May, `SUIHTER` was the best at predicting daily deaths and awarded second at predicting new infections in Italy. In the week of 10th to 16th May, `SUIHTER` ranked first in predicting the evolution of both compartments (daily deaths and new positives).

[3] `https://covid19forecasthub.eu/`

Chapter 3

Optimal Control for ODE systems

This chapter is devoted to a short description of basic elements of Optimal Control theory for ODE-based systems of equations. Designing control logic that commands dynamical systems for obtaining desired outputs is a common feature in different fields, such as decision making in economic or social processes, or trajectories' controls for mechanical systems. One typical example of optimal control problem is the *Brachistocrone problem*, which aims at determining the fastest curve connecting two distinct points in the two dimensional plane (Figure 3.1).

Optimal control theory is the mathematical theory displaying how to act on some forcing variables in order to maximize or minimize some quantitative measures of performances. Main references for this chapter are represented by [107], [18], [68] and [98]. In order to formulate a feasible control problem, two issues have to be investigated:

- *Existence and Uniqueness of the solution.* Mathematical problems dealing with dynamical systems are defined *Well-posed problems* according to Hadamard if they admit one and only one solution, with continuous dependance of the solution from data. Therefore, we have to assure that the optimal control problem is well-posed, as well as each ODE system that will consequently arise in the formulation. In the case of systems of Ordinary Differential Equations we refer to Lyapunov theory for existence and uniqueness of local and global solutions in [16]. We refer to [112] for a theoretical dissertation about existence and uniqueness of the optimal control solution;

- *Necessary and Sufficient conditions for Optimality.* Once the solution has been recovered, we need some criteria for assessing optimality of the solution.

In Section 3.1 we define the mathematical formulation of an optimal control problem governed by ODEs. Section 3.2 provides a short introduction to Calculus of Variations applied to control theory. Section 3.3 deals with the *Pontryagin Maximum principle*, which provides a practical mathematical tool to solve optimal control problems. Finally, in Section 3.4 we apply Pontryagin's theory to the calibration problem of a `SEIR` model.

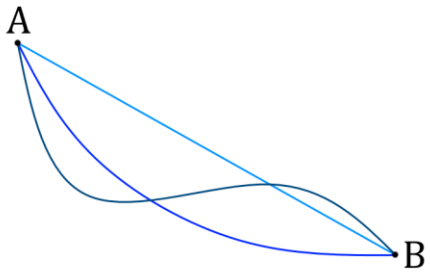


Figure 3.1: Three possible curves of the Brachistocrone problem. Among all curves in $\mathbb{R}^2$ starting from A and ending at B, the solution is the curve which is crossed in the smallest timeframe.

3.1 Problem formulation and definitions

Define $n \in \mathbb{N}$ the number of states and $x_1(t), \dots, x_n(t) \in \mathbb{R}$ the set of state variables of the process, *i.e* the set of all variables which evolve during the timeframe of interest $[0; T_f]$. Define $\boldsymbol{x}(t) = [x_1(t), x_2(t), \dots, x_n(t)]^T \in \mathbb{R}^n$ the vector collecting all states at time t. Define $m \in \mathbb{N}$ the number of process controls and $u_1(t), \dots, u_m(t) \in \mathbb{R}$ the set of controls, *i.e* forcing variables that can be varied in order to get different solutions of the state problem. Define $\boldsymbol{u}(t) = [u_1(t), u_2(t), \dots, u_m(t)]^T \in \mathbb{R}^m$ the vector of controls at time t.

The *state problem* is defined as the dynamical system associated with the following system of Ordinary Differential Equations,

$$\dot{\boldsymbol{x}}(t) = F(\boldsymbol{x}, \boldsymbol{u}, t) \quad \forall t \in (0; T_f], \tag{3.1}$$

where $F(\boldsymbol{x}, \boldsymbol{u}, t)$ represents the evolution law of the system. In order to consider a well-posed differential problem initial conditions have to be provided, *i.e*

$$\boldsymbol{x}(0) = \boldsymbol{x_0} \in \mathbb{R}^n. \tag{3.2}$$

Define X_{ad} as the space of admissible state variables, and U_{ad} as the space of admissible controls. Such spaces are determined in order to fulfill constraints of different nature. For instance, one thinks of box-constrained variables, *i.e*

$$0 \leq \boldsymbol{x}(t) \leq M, \forall t \in (0; T_f], \tag{3.3}$$

or

$$0 \leq \boldsymbol{u}(t) \leq C, \forall t \in (0; T_f]. \tag{3.4}$$

When costraints have to be considered, the optimal control problem is said to be a *Constrained Optimal Control Problem*, while it is an *Unconstrained Optimal Control Problem* if controls and states can be freely chosen in the $\mathbb{R}^d$ respective dimensional space. The optimal control problem can be formulated as follows:

Optimal Control Problem. *Let $\boldsymbol{u} \in U_{ad}$, $\boldsymbol{x} \in X_{ad}$ and $F(\boldsymbol{x}, \boldsymbol{u}, t)$ the dynamical evolution of the state problem. Find $\boldsymbol{u}^* \in U_{ad}$, optimal control, which causes*

$$\dot{\boldsymbol{x}}(t) = F(\boldsymbol{x}, \boldsymbol{u}, t) \quad \forall t \in (0; T_f]$$

to follow a trajectory $\boldsymbol{x}^* \in X_{ad}$ that minimizes the following ***performance measure*** *(or* ***cost functional****)*

$$J = h(\boldsymbol{x}(T_f), T_f) + \int_0^{T_f} g(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, dt. \tag{3.5}$$

In the general framework the cost functional is constituted by final-time controls, ruled by function $h : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, and an integral component, which takes into account for measures during the whole interval of concern, ruled by function $g : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, also known as *Lagrangian Function*.

Designing a proper performance measure is not at all an easy subject to deal with. Indeed, the designer has in mind behaviours which have to be satisfied when optimal performances are reached and that have to be translated into mathematical expressions depending on state variables and controls. Moreover, its expression must be regular enough in order to guarantee conditions for existence and uniqueness of the solution. Once a suitable performance measure has been selected, the next task is to determine a way for minimizing this criterion. In [107] two methods are investigated in order to achieve the minimization goal: the **Dynamic Programming procedure** that leads to a functional equation that is amenable to solution through the use of digital calculators; a variational approach based on **Pontryagin principle**. In this thesis, we have extensively used the second approach that leads to derive theoretical conditions for assessing optimality of the solution.

3.2 Basic elements of Calculus of Variations

In order to state the *Pontryagin maximum principle*, we recall some basic concepts from Calculus of Variations.

Let $J : \Omega \to \mathbb{R}$ a functional. We need to introduce some quantities that are necessary to identify extremal points of J.

Definition 9. *Let f and $f + \delta f$ functions in Ω. Then, the* ***increment*** *ΔJ of J is*

$$\Delta J = J(f + \delta f) - J(f), \tag{3.6}$$

depending on both f and δf. Instead, δf is called ***variation*** *of f.*

The variation of a functional plays the same role in determining extrema of functionals as the differential does to find maxima and minima points for functions. Indeed, the increment of a function $f : \mathbb{R}^n \to \mathbb{R}$ can be decomposed in the following way,

$$\Delta f(\boldsymbol{x}, \Delta\boldsymbol{x}) = df(\boldsymbol{x}, \Delta\boldsymbol{x}) + g(\boldsymbol{x}, \Delta\boldsymbol{x}) \cdot \|\Delta\boldsymbol{x}\|, \tag{3.7}$$

where df is a linear function of the increment $\Delta\boldsymbol{x}$. If

$$\lim_{\|\Delta\boldsymbol{x}\|\to 0} g(\boldsymbol{x}, \Delta\boldsymbol{x}) = 0,$$

f is said to be differentiable at $\boldsymbol{x}$, and df is the differential of f in $\boldsymbol{x}$. From the definition of df it is possible to develop a rule for the differential of a differentiable function of n variables, *i.e*

$$df = \frac{\partial f}{\partial q_1}\Delta q_1 + \frac{\partial f}{\partial q_2}\Delta q_2 + \ldots + \frac{\partial f}{\partial q_n}\Delta q_n. \tag{3.8}$$

In the same way calculus defines differentials for functions, we define the *variation* of a functional J in the following way:

Definition 10. *The increment ΔJ of a functional J can be rewritten as*

$$\Delta J(f, \delta f) = \delta J(f, \delta f) + g(f, \delta f) \cdot \|\delta f\|, \tag{3.9}$$

where δJ is linear in δf. If

$$\lim_{\|\delta f\|\to 0} g(f, \delta f) = 0,$$

*the functional J is said to be **differentiable** on f and δJ is the **variation** of J computed at the function f.*

Hence, δJ represents the linear approximation of the difference in the functional caused by two sufficiently close functions, namely if the two curves (or functions) are close enough the variation is a good approximation of the increment.

We introduce the definition of extrema for cost functionals and the *Fundamental Theorem of the Calculus of Variations*, which gives a necessary condition for functions in order to be extremal points (analogous to *Fermat Theorem* for functions).

Definition 11. *A functional $J : \Omega \to \mathbb{R}$ has f^* as relative minimum if*

$$\exists \epsilon > 0 \,|\, \forall\, f \in \Omega,\ \|f - f^*\| < \epsilon \Rightarrow \Delta J = J(f) - J(f^*) \geq 0, \tag{3.10}$$

it has f^* as relative maximum if

$$\exists \epsilon > 0 \,|\, \forall\, f \in \Omega,\ \|f - f^*\| < \epsilon \Rightarrow \Delta J = J(f) - J(f^*) \leq 0. \tag{3.11}$$

If ϵ can be chosen arbitrarily large, the extremum is said to be global or absolute.

Theorem 7. ***(Fundamental Theorem of Calculus of Variations)*** *Assume f is a function in Ω, domain of the functional of interest J. Assume J is a differentiable functional in f and suppose that functions in Ω are not constrained by any boundaries. If f^* is an extremal of J, then $\delta J(f^*, \delta f) = 0$ for all admissible δf.*

For the proof, which can be easily recovered by contradiction, a possible reference is [68].

3.3 Pontryagin maximum principle

In this section we derive optimality conditions for the solution of an optimal control problem.

Consider the state problem (3.1). Assume that h is a differentiable function, and that admissible states and controls are not bounded. Then, integrating by parts,

$$h(\boldsymbol{x}(T_f), T_f) = \int_0^{T_f} \frac{d}{dt}[h(\boldsymbol{x}(t), t)]\, dt + h(\boldsymbol{x_0}, 0), \tag{3.12}$$

and, substituting (3.12) in the performance measure, one obtains

$$J = \int_0^{T_f} \left\{ g(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \frac{d}{dt}[h(\boldsymbol{x}(t), t)] \right\} dt + h(\boldsymbol{x_0}, 0). \tag{3.13}$$

The second addendum of the right-hand side is neglected since it is determined by initial conditions. Using the chain rule, (3.13) becomes

$$J = \int_0^{T_f} \left\{ g(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \left[\frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x}(t), t)\right]^T \dot{\boldsymbol{x}}(t) + \frac{\partial h}{\partial t}(\boldsymbol{x}(t), t) \right\} dt. \tag{3.14}$$

To include the state problem in the cost functional, we define the augmented cost functional introducing a vector function $\boldsymbol{p} : \mathbb{R} \to \mathbb{R}^n \in P_{ad}$, known as *vector of Lagrange multipliers*. Its value at the solution represents the rate of change in the maximal value of the objective function when the constraint is relaxed.

Hence,

$$\begin{aligned} J_a = \int_0^{T_f} \Bigg\{ & g(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \left[\frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x}(t), t)\right]^T \dot{\boldsymbol{x}}(t) + \frac{\partial h}{\partial t}(\boldsymbol{x}(t), t) + \\ & + \boldsymbol{p}(t)^T [F(\boldsymbol{x}, \boldsymbol{u}, t) - \dot{\boldsymbol{x}}(t)] \Bigg\} dt. \end{aligned} \tag{3.15}$$

The integrand of the augmented cost functional can be defined in the following way,

$$\begin{aligned} g_a(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{p}, t) = g(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \left[\frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x}(t), t)\right]^T \dot{\boldsymbol{x}}(t) + \frac{\partial h}{\partial t}(\boldsymbol{x}(t), t) + \\ + \boldsymbol{p}(t)^T [F(\boldsymbol{x}, \boldsymbol{u}, t) - \dot{\boldsymbol{x}}(t)]. \end{aligned} \tag{3.16}$$

Introduce the variations $\delta\boldsymbol{x}, \delta\dot{\boldsymbol{x}}, \delta\boldsymbol{u}, \delta\boldsymbol{p}$ and set to zero the variation of the augmented cost functional δJ_a. Employing to the Fundamental Theorem of Calculus of Variations,

(Theorem 7), yields

$$\begin{aligned}\delta J_a(\boldsymbol{u}^*) = 0 = &\left[\frac{\partial g_a}{\partial \dot{\boldsymbol{x}}}(\boldsymbol{x}^*(T_f), \dot{\boldsymbol{x}}^*(T_f), \boldsymbol{u}^*(T_f), T_f)\right]^T \delta \boldsymbol{x}_f \\ &+ \Bigg[g_a(\boldsymbol{x}^*(T_f), \dot{\boldsymbol{x}}^*(T_f), \boldsymbol{u}^*(T_f), T_f) \\ &- \left[\frac{\partial g_a}{\partial \dot{\boldsymbol{x}}}(\boldsymbol{x}^*(T_f), \dot{\boldsymbol{x}}^*(T_f), \boldsymbol{u}^*(T_f), T_f)\right]^T \dot{\boldsymbol{x}}^*(T_f)\Bigg] \delta T_f \\ &+ \int_0^{T_f} \Bigg\{ \Bigg[\left[\frac{\partial g_a}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t), \dot{\boldsymbol{x}}^*(t), \boldsymbol{u}^*(t), t)\right]^T \\ &- \frac{d}{dt}\left[\frac{\partial g_a}{\partial \dot{\boldsymbol{x}}}(\boldsymbol{x}^*(t), \dot{\boldsymbol{x}}^*(t), \boldsymbol{u}^*(t), t)\right]^T\Bigg] \delta \boldsymbol{x}(t) \\ &+ \left[\frac{\partial g_a}{\partial \boldsymbol{u}}(\boldsymbol{x}^*(t), \dot{\boldsymbol{x}}^*(t), \boldsymbol{u}^*(t), t)\right]^T \delta \boldsymbol{u}(t) \\ &+ \left[\frac{\partial g_a}{\partial \boldsymbol{p}}(\boldsymbol{x}^*(t), \dot{\boldsymbol{x}}^*(t), \boldsymbol{u}^*(t), t)\right]^T \delta \boldsymbol{p}(t)\Bigg\} dt.\end{aligned} \tag{3.17}$$

At this point we proceed isolating all terms involving the function h, and, applying integration by parts and the state problem. Then we reduce the principle to a set of $2n$ first order differential equations. For further details we refer to [107].

We introduce the so-called *Hamiltonian function*, defined as follows

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) = g(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \boldsymbol{p}^T(t)[F(\boldsymbol{x}, \boldsymbol{u}, t)]. \tag{3.18}$$

Therefore, the necessary optimality conditions can be rewritten for $\mathcal{H}$ as

$$\begin{cases} \dot{\boldsymbol{x}}^*(t) = \dfrac{\partial \mathcal{H}}{\partial \boldsymbol{p}}(\dot{\boldsymbol{x}}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t), \\ \dot{\boldsymbol{p}}^*(t) = -\dfrac{\partial \mathcal{H}}{\partial \boldsymbol{x}}(\dot{\boldsymbol{x}}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t), \\ \dfrac{\partial \mathcal{H}}{\partial \boldsymbol{u}}(\dot{\boldsymbol{x}}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t) = 0, \end{cases} \tag{3.19}$$

$\forall t \in (0; T_f]$. Initial Conditions for the state problem are *apriori* prescribed. For the costate problem, or multipliers evolution, the following final time conditions are imposed:

$$\boldsymbol{p}^*(T_f) = \frac{\partial h}{\partial \boldsymbol{x}}(T_f). \tag{3.20}$$

The set of equations (3.19) which corresponds to a set of necessary conditions for optimality is known in literature as *Pontryagin Maximum Principle*. This is the set of equations that is solved for determining states and controls in the implemented algorithm further specified in Chapter 5.

It is possible to formulate sufficient conditions for triplets of states, multipliers and controls for granting optimality of the solution [98]:

Theorem 8. *Let J be concave with respect to $\boldsymbol{x}$, $\boldsymbol{u}$ and let one among the following conditions hold:*

- *Lagrangian g concave in $\boldsymbol{x}$, $\boldsymbol{u}$ and $\boldsymbol{p}^* \geq 0$;*
- *Lagrangian g convex in $\boldsymbol{x}$, $\boldsymbol{u}$ and $\boldsymbol{p}^* \leq 0$;*
- *Lagrangian g linear in $\boldsymbol{x}$;*

Then $\boldsymbol{u}^*$ and $\boldsymbol{x}^*$ are the optimal control and the optimal trajectory respectively.

For a detailed proof see [98].

3.4 An example: calibration of the SEIR model

As in [108], the Classical Inverse Problem (CIP) can be built in order to estimate parameters of ODE systems. The CIP can be formulated as follows:

Optimal Control Problem.

$$\arg\min_{\theta \in \mathbb{P}} \|u(\theta) - u_{obs}\| \tag{3.21}$$

subject to

$$\dot{u} = F(u, \theta) \tag{3.22}$$

where θ is the parameter to calibrate, u_{obs} is the observation of the solution u of the differential problem.

The CIP is an optimal control problem where the cost functional measures adherence of the solution to data in a suitably-defined norm. Let us introduce the CIP for the transmission rate β of the SEIR model. More precisely, consider the cost functional,

$$\begin{aligned} J(\beta) = &\frac{1}{2}\int_0^T \{(S(\beta) - S_{obs})^2 + (E(\beta) - E_{obs})^2 + (I(\beta) - I_{obs})^2 + \beta^2\}\,\mathrm{d}t + \\ &+ (S(T,\beta) - S_{obs})^2 + (E(T,\beta) - E_{obs})^2 + (I(T,\beta) - I_{obs})^2, \end{aligned} \tag{3.23}$$

with the parameter β belonging to the space of parameters $\mathbf{P}$ (assume for instance the space of continuous functions in the interval $[0;T]$). S_{obs}, E_{obs} and I_{obs} represent interpolations of sampled data, for instance polynomial interpolations. The term $\int_0^T \beta^2\,dt$

is a regularization term that is often introduced in optimal control problems, which helps in convergence and stability, mantaining under control the value of the parameter itself. The other quadratic terms measure the distance between the solution variables and the interpolation of data during the whole timeframe and at the final time.

Let us write down the complete formulation of the CIP applied to `SEIR`:

Optimal Control Problem.

$$\arg\min_{\beta \in \mathbf{P}} J(\beta)$$

subject to the state problem

$$\begin{cases} \dot{S} = -\dfrac{\beta}{N} S\, I, & \forall t \in (0;T], \\ \dot{E} = \dfrac{\beta}{N} S\, I - \alpha E, & \forall t \in (0;T], \\ \dot{I} = \alpha E - \gamma I, & \forall t \in (0;T], \\ R = N - I - E - S, & \forall t \in (0;T], \end{cases} \tag{3.24}$$

with initial condition for each state variable (S_0, E_0, I_0, R_0).

Since the recovered variable can be easily deduced algebraically by the other unknowns, from now on it will be neglected. We cope with the derivation of the optimal control conditions in two ways:

1. The Hamiltonian function introduced in the previous section can be defined and we can directly apply Pontryagin Maximum principle. Hence, defining the vector of Lagrange multipliers $\mathbf{p}(t) = \begin{bmatrix} p_1(t) \\ p_2(t) \\ p_3(t) \end{bmatrix}$ and the Hamiltonian function as

$$\begin{aligned} \mathcal{H}(S, E, I, \mathbf{p}, \beta) =& \frac{1}{2}((S - S_{obs})^2 + (E - E_{obs})^2 + (I - I_{obs})^2 + \beta^2) + \\ &+ \mathbf{p}(t)^T \begin{bmatrix} -\frac{\beta}{N} S\, I \\ \frac{\beta}{N} S\, I - \alpha E \\ \alpha E - \gamma I \end{bmatrix} + (S(T, \beta) - S_{obs})^2 + \\ &+ (E(T, \beta) - E_{obs})^2 + (I(T, \beta) - I_{obs})^2, \end{aligned} \tag{3.25}$$

   then solving conditions (3.19), one gets control and state variables. Finally, the functional derivative of the cost functional, also known as *Euler-Lagrange derivative* or *Fréchet derivative*, with respect to β is :

$$\frac{\delta J}{\delta \beta} = \mathcal{H}_{,\beta}|_{S^*, E^*, I^*, p_1^*, p_2^*, p_3^*} = \beta - \frac{p_1^*}{N} S(\beta) I(\beta) + \frac{p_2^*}{N} S(\beta) I(\beta). \tag{3.26}$$

The steepest descending direction along the cost functional is

$$v_{max}(t) = -\left(-\frac{p_1}{N}S(\beta^*)I(\beta^*) + \frac{p_2}{N}S(\beta^*)I(\beta^*) + \beta^*\right), \tag{3.27}$$

opposite to the gradient. This is the direction to compute at each step of descending algorithms in order to minimize the cost functional.

2. Another possible way is the variational procedure suggested in [98].

We detail the second strategy in the following computations.

Define the vector of Lagrange multipliers $\mathbf{p}(t) = \begin{bmatrix} p_1(t) \\ p_2(t) \\ p_3(t) \end{bmatrix}$ and rewrite the cost functional including the state costraints in the following way:

$$\begin{aligned} J(\beta) = &\frac{1}{2}\int_0^T \{(S(\beta) - S_{obs})^2 + (E(\beta) - E_{obs})^2 + (I(\beta) - I_{obs})^2 + \beta^2 + \\ &+ \mathbf{p}(t)^T \begin{bmatrix} -\frac{\beta}{N}S\,I - \dot{S} \\ \frac{\beta}{N}S\,I - \alpha E - \dot{E} \\ \alpha E - \gamma I - \dot{I} \end{bmatrix} \} \,\mathrm{d}t + \\ &+ (S(T,\beta) - S_{obs})^2 + (E(T,\beta) - E_{obs})^2 + (I(T,\beta) - I_{obs})^2. \end{aligned} \tag{3.28}$$

Integrating by parts yields

$$\int_0^T p_1(t)\dot{S}(t,\beta)\,\mathrm{d}t = p_1(T)S(T,\beta) - p_1(0)S_0 - \int_0^T \dot{p_1}(t)S(t)\,\mathrm{d}t,$$

$$\int_0^T p_2(t)\dot{E}(t,\beta)\,\mathrm{d}t = p_2(T)E(T,\beta) - p_2(0)E_0 - \int_0^T \dot{p_2}(t)E(t)\,\mathrm{d}t,$$

$$\int_0^T p_3(t)\dot{I}(t,\beta)\,\mathrm{d}t = p_3(T)I(T,\beta) - p_3(0)I_0 - \int_0^T \dot{p_3}(t)I(t)\,\mathrm{d}t.$$

Consider now $\beta^*(t)$ as the unknown optimal value of the optimal control problem, and define $\beta(t) = \beta^*(t) + hv(t)$ another parameter belonging to $\mathbf{P}$ obtained increasing β^* of step h, small enough for $\beta(t)$ to remain in $\mathbf{P}$, along the fixed direction $v(t)$.

Introduce the function $\Phi(h)$ which evaluates the cost functional at distance h from the unknown optimum β^* along $v(t)$, *i.e.*

$$\begin{aligned}\Phi(h) = \int_0^T \Big\{ & \frac{1}{2}(S(\beta^*+hv) - S_{obs})^2 + \frac{1}{2}(E(\beta^*+hv) - E_{obs})^2 + \frac{1}{2}(I(\beta*+hv) - I_{obs})^2 + \\ & + \frac{1}{2}(\beta^*+hv)^2 + \dot{p_1} S(\beta^*+hv) - p_1 \Big(\frac{\beta^*}{N} S(\beta^*+hv) I(\beta^*+hv) + \\ & + \frac{hv}{N} S(\beta^*+hv) I(\beta^*+hv) \Big) + \dot{p_2} E(\beta^*+hv) + \\ & - p_2 \Big(-\frac{\beta^*}{N} S(\beta^*+hv) I(\beta^*+hv) - \frac{hv}{N} S(\beta^*+hv) I(\beta^*+hv) + \\ & + \alpha E(\beta^*+hv) \Big) + \dot{p_3} I(\beta^*+hv) - p_3(\gamma I(\beta^*+hv) - \alpha E(\beta^*+hv)) \Big\} \mathrm{d}t + \\ & + (S(T,\beta) - S_{obs})^2 - p_1(T) S(T, \beta^*+hv) + (E(T,\beta) - E_{obs})^2 + \\ & - p_2(T) E(T, \beta^*+hv) + (I(T,\beta) - I_{obs})^2 - p_3(T) I(T, \beta^*+hv) + \\ & + p_1(0) S_0 + p_2(0) E_0 + p_3(0) I_0 \;=\; J(\beta^*+hv). \end{aligned} \tag{3.29}$$

Consider its derivative with respect to h,

$$\begin{aligned}\Phi'(h) = \int_0^T \Big\{ & (S(\beta^*+hv) - S_{obs}) \frac{\partial S}{\partial h} + (E(\beta^*+hv) - E_{obs}) \frac{\partial E}{\partial h} + (I(\beta^*+hv) - I_{obs}) \frac{\partial I}{\partial h} + \\ & + (\beta^*+hv) v + \dot{p_1} \frac{\partial S}{\partial h} - p_1 \frac{\beta^*}{N} \frac{\partial S}{\partial h} I(\beta^*+hv) - p_1 \frac{\beta^*}{N} \frac{\partial I}{\partial h} S(\beta^*+hv) + \\ & - p_1 \frac{v}{N} S(\beta) I(\beta) - p_1 \frac{hv}{N} \Big(\frac{\partial S}{\partial h} I(\beta^*+hv) + \frac{\partial I}{\partial h} S(\beta^*+hv) \Big) + \\ & + \dot{p_2} \frac{\partial E}{\partial h} - p_2 \alpha \frac{\partial E}{\partial h} + p_2 \frac{\beta^*}{N} \frac{\partial S}{\partial h} I(\beta^*+hv) + p_2 \frac{\beta^*}{N} \frac{\partial I}{\partial h} S(\beta^*+hv) + \\ & + p_2 \frac{v}{N} S(\beta^*) I(\beta^*) + p_2 \frac{hv}{N} \Big(\frac{\partial S}{\partial h} I(\beta^*+hv) + \frac{\partial I}{\partial h} S(\beta^*+hv) \Big) + \\ & + \dot{p_3} \frac{\partial I}{\partial h} - \gamma p_3 \frac{\partial I}{\partial h} + \alpha p_3 \frac{\partial E}{\partial h} \Big\} \mathrm{d}t + (2(S(T,\beta^*+hv) - S_{obs}) - p_1(T)) \frac{\partial S}{\partial h} + \\ & + (2(E(T,\beta^*+hv) - E_{obs}) - p_2(T)) \frac{\partial E}{\partial h} + (2(I(T,\beta^*+hv) - I_{obs}) + \\ & - p_3(T)) \frac{\partial I}{\partial h}, \end{aligned} \tag{3.30}$$

and compute it at $h = 0$ ($\beta(t) = \beta^*$).

In order to have a stationary point we impose $\Phi'(0) = 0$:

$$\begin{aligned}\Phi'(0) = \int_0^T \Big\{ &(S(\beta^*) - S_{obs})\frac{\partial S}{\partial h} + (E(\beta^*) - E_{obs})\frac{\partial E}{\partial h} + (I(\beta^*) - I_{obs})\frac{\partial I}{\partial h} + \beta^* v + \\ &+ \dot{p_1}\frac{\partial S}{\partial h} - p_1\frac{\beta^*}{N}\frac{\partial S}{\partial h}I(\beta^*) - p_1\frac{\beta^*}{N}\frac{\partial I}{\partial h}S(\beta^*) - p_1\frac{v}{N}S(\beta^*)I(\beta^*) + \dot{p_2}\frac{\partial E}{\partial h} + \\ &- p_2\alpha\frac{\partial E}{\partial h} + p_2\frac{\beta^*}{N}\frac{\partial S}{\partial h}I(\beta^*) + p_2\frac{\beta^*}{N}\frac{\partial I}{\partial h}S(\beta^*) + p_2\frac{v}{N}S(\beta^*)I(\beta^*) + \dot{p_3}\frac{\partial I}{\partial h} + \\ &- \gamma p_3\frac{\partial I}{\partial h} + \alpha p_3\frac{\partial E}{\partial h}\Big\}\mathrm{d}t + (2(S(T,\beta^*) - S_{obs}) - p_1(T))\frac{\partial S}{\partial h} + \\ &+ (2(E(T,\beta^*) - E_{obs}) - p_2(T))\frac{\partial E}{\partial h} + \\ &+ (2(I(T,\beta^*) - I_{obs}) - p_3(T))\frac{\partial I}{\partial h}.\end{aligned} \tag{3.31}$$

Collecting all terms multiplied by $\frac{\partial S}{\partial h}$, $\frac{\partial E}{\partial h}$ and $\frac{\partial I}{\partial h}$ respectively, we choose the multipliers in order to neglect those terms. Therefore, we obtain the following ODE systems of backward problems:

$$\begin{cases} \dot{p_1^*} = p_1^*\frac{\beta^*}{N}I - p_2^*\frac{\beta^*}{N}I - (S - S_{obs}) \\ p_1^*(T) = 2(S(T,\beta^*) - S_{obs}) \end{cases}$$

$$\begin{cases} \dot{p_2^*} = \alpha p_2^* - \alpha p_3^* - (E - E_{obs}) \\ p_2^*(T) = 2(E(T,\beta^*) - E_{obs}) \end{cases} \tag{3.32}$$

$$\begin{cases} \dot{p_3^*} = \gamma p_3^* + p_1^*\frac{\beta^*}{N}S - p_2^*\frac{\beta^*}{N}S - (I - I_{obs}) \\ p_3^*(T) = 2(I(T,\beta^*) - I_{obs}). \end{cases}$$

In this way,

$$\Phi'(0) = \int_0^T \left[v\left(-\frac{p_1^*}{N}S(\beta^*)I(\beta^*) + \frac{p_2^*}{N}S(\beta^*)I(\beta^*) + \beta^* \right) \right] \mathrm{d}t. \tag{3.33}$$

From the abitrariness of the descending direction $v(t)$, we obtain the last necessary condition:

$$\beta^* = \frac{p_1^*}{N}S(\beta^*)I(\beta^*) - \frac{p_2^*}{N}S(\beta^*)I(\beta^*), \tag{3.34}$$

implying that

$$\Phi'(0) = 0. \tag{3.35}$$

Chapter 4

Vaccination and Optimal Control

Examining the state of the art, many models integrate vaccination through the introduction of compartments which take into account for vaccinated individuals (see, *e.g.*, [77], [56], [75], [58], [103]). Based on their knowledge, we aim at formulating an epidemic model for the epidemic context of SARS-CoV-2 in Italy. This is a challenging setting to work on, since we have to deal with many vaccines, most of them requiring two administrations with different efficacy properties, which have to fulfill constraints regarding elapsing time between administrations and which have to take into account for personal clinical history. Policy makers have to arrange the vaccination campaign in order to take into account for fragilities, comorbidities and different ages of the individuals and trying to relieve social and economical pressure imposed by lockdown restrictions. Some works have focused on developing sophisticated models for coping with the aforementioned issues, for instance [83], [29], [82], [61], [57], [74], [102].

This chapter is devoted to the introduction of two new *vaccination-oriented* epidemic models and to the corresponding formulation of Optimal Control problems. The two models try to incorporate the features of the vaccination campaign started in December 2020/January 2021 against COVID-19 pandemic in most countries around the world. They have been built in order to capture key-characteristics of the disease itself (*e.g.* the re-infection possibility, different stages of the infective path, a delayed fatality function which takes into account for the average days from the infection to death), as well as key-features typical of the vaccines adopted against SARS-CoV-2 (*e.g.* two-shots vaccine cycle, efficacy against transmissibility and efficacy against severe disease).

Section 4.1 introduces the aforementioned mathematical compartmental models, underlining consistency of these models with the Italian pandemic scenario. In Section 4.2 we formulate two optimal control problems we will solve numerically in the next chapters.

4.1 SEIHRDVW models

For the purpose of this thesis, we are interested in describing the dynamics of SARS-CoV-2 infection, as well as studying possible vaccination policies and their contribution to eradicate the pandemic. To this end, we consider the SEIHRDW model in two variants. This novel model was developed starting from the SEIHRD model formulated by EpiMox research group at MOX (Luca Dede' private communication) and integrating two specific compartments accounting for vaccinated individuals with one (V) or two doses (W). Thus, we obtain a new epidemic model which is completely conceived with Italian vaccination program as well as the progressive evolution of COVID-19 infection represented by classes E, I and H. We present two different versions of the same model.

4.1.1 SEIHRDVW model: version 1

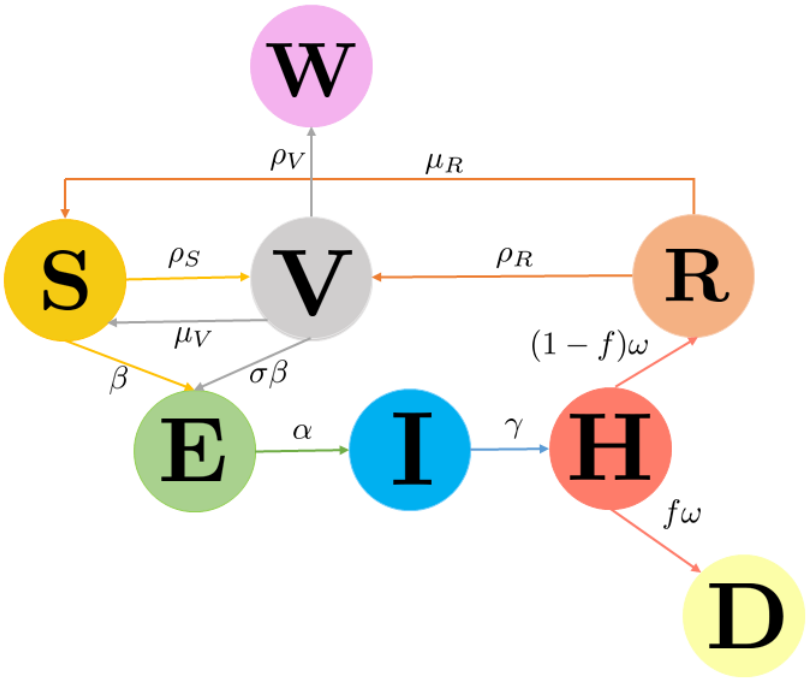


Figure 4.1: Flowchart of the SEIHRDVW model, version 1.

The first version of the `SEIHRDVW` reads as follows:

$$\begin{cases} \dot{S} = -\beta \dfrac{SI}{N} - \rho_S S + \mu_R R + \mu_V V \\ \dot{E} = \beta \dfrac{(S + \sigma V) I}{N} - \alpha E \\ \dot{I} = \alpha E - \gamma I \\ \dot{H} = \gamma I - \omega H \\ \dot{R} = (1 - f(S,V))\, \omega H - \mu_R R - \rho_R R \\ \dot{D} = f(S,V)\, \omega H \\ \dot{V} = -\beta \dfrac{\sigma V I}{N} + \rho_S S + \rho_R R - \mu_V V - \rho_V V \\ \dot{W} = \rho_V V \end{cases} \tag{4.1}$$

with

$$f(S,V) = \bar{f} \frac{S(t-15) + \theta \sigma V(t-15)}{S(t-15) + \sigma V(t-15)}. \tag{4.2}$$

Each compartment stands for one average population class involved in the infection process:

- ***S***: the susceptibles class is constituted by individuals which have no protection against the virus. By a medical point of view, no antibodies have been developed by people in this class;
- ***E***: when contacts with infectious individuals cause an infection, people enter in the exposed class, where after an incubation time is passed individuals are infectious too;
- ***I***: both detected and undetected infectious individuals belong to this class. People in this class could be both symptomatic and asymptomatic;
- ***H***: after a transient period in E and I classes, infectious individuals can be still infected, but they are no more infectious. In this case they belong to the H class, where H stands for healing. We assume that no deaths neither recoverings happen without passing in the healing class;
- ***R***: people healed from the disease belong to the recovered class. We assume that people in this class have developed at least a threshold level of antibodies against the infection. From this class, after a certain amount of time immunisation could run out (still object of study) and therefore individuals can return to the S class;

- $\boldsymbol{D}$: it stands for deceased individuals. This class counts deceased individuals due to the infection;

- $\boldsymbol{V}$: we assume to consider a two-shots vaccine with certain effectiveness properties after the first administration, that can help in preventing severe effects leading to death and/or in the transmission spread. Indeed, the majority of authorized vaccines in Italy (except for the *Jannsen* vaccine) are based on two administrations in order to complete the vaccination cycle. This is the case of the mRNA-based vaccines Moderna and BioNTech/Pfizer, but even for the adenovirus-based vaccine developed by AstraZeneca. Two shots are necessary for these vaccines since they work by exposing the body to different parts of the virus, hence more doses imply better possibilities for the immune system to counter-attack a future infection. Vaccines create memory cells (*T-cells*) that are prepared to respond immediately to future infections.

  When vaccine doses are administered to recovered or susceptibles individuals they are moved to the V class. If the vaccine does not guarantee perfect immunisation, people in V can be infected through adequate contacts with infectious individuals;

- $\boldsymbol{W}$: this class comprises all individuals who completed the vaccination cycle. We assume that perfect immunisation is granted after the administration of the second dose, which is often called *booster shot* since it responds to an already estabilished exposure. This class is a kind of *black hole absorbing class* similarly to the deceased class, meaning that once individuals are in W there is no possibility of contracting the infection.

We clarify the physical and biological meaning of the parameters involved in (4.1). Notice that, for extimating an accurate Confidence Interval for each parameter, a calibration problem similar to the one introduced in Section 3.4 is required for each parameter. We do not deal with this issue but we approximately infer the values of the parameter from studies and articles which have been already approved (see, *e.g.*, [94], [114], [44], [6]).

- $\boldsymbol{\beta}$: transmission rate (refer to Section 2.2.1), depending on NPIs and transmissibility of the disease itself;

- $\boldsymbol{\alpha}$: it is dimensionally the inverse of a time and corresponds to the inverse of the average incubation time of the disease (medium time spent by individuals in this class);

- $\boldsymbol{\gamma}$: healing rate, namely rate at which people become healing from infectious class (inverse of the medium time spent in class I);

- $\boldsymbol{\omega}$: healed rate, inverse of the medium time spent in class H;

- $\bar{\boldsymbol{f}}$: fatality constant, which is the rate of fatality extimated without vaccines support. It can be extimated using statistics providing medium deaths over total infectious individuals;
- $\boldsymbol{\sigma}$: vaccine effectiveness on transmissibility, which reduces the transmission rate among contacts between I and V individuals, ranging from 0 to 1;
- $\boldsymbol{\theta}$: vaccine effectiveness on mortality, reduces mortality of vaccinated individuals, ranging from 0 to 1;
- $\boldsymbol{\rho_R}$: vaccination rate of recovered individuals, imposed by vaccination policies;
- $\boldsymbol{\rho_S}$: vaccination rate of susceptibles individuals, imposed by vaccination policies;
- $\boldsymbol{\rho_V}$: vaccination rate of administrations of second doses, imposed by vaccination policies;
- $\boldsymbol{\mu_R}$: inverse of the time lapse when recovered coverage is effective;
- $\boldsymbol{\mu_V}$: inverse of the lapse when vaccine's coverage is effective.

Equation (4.2) represents the *fatility function* that has been considered in our models. It is proportional to the fatality rate through a ratio depending on the populosity of compartments that could become infected, *i.e.* S and V. Vaccine effectiveness θ acts reducing the numerator of the ratio, hence the fatality function. This function has shown more adherence to realistic situations if state variables are evaluated at $(t-15)$, where 15 days is the medium time to flow from susceptible's class to deceased one. For this model, we can derive the reproduction number $\mathcal{R}_t$ applying the Method of the Next Generation Matrix, previously introduced. We obtain:

$$\mathcal{R}_t = \frac{\beta}{\gamma}\frac{(S(t) + \sigma V(t))}{N}. \tag{4.3}$$

4.1.2 `SEIHRDVW` model: version 2

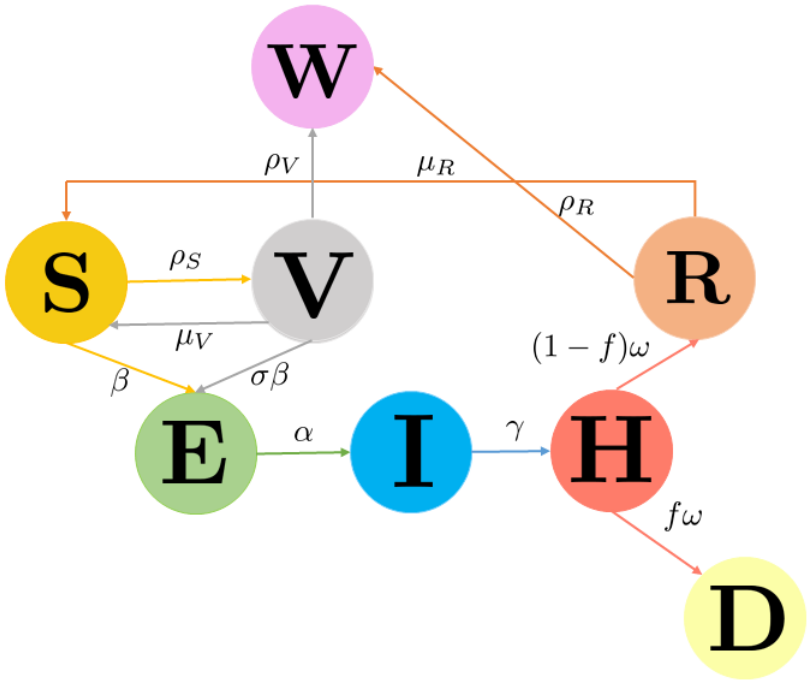


Figure 4.2: Flowchart of the `SEIHRDVW` model, version 2.

We introduce a modified version of the `SEIHRDVW` model. The fundamental difference with respect to the previously-presented model consists in the vaccination policy for recovered individuals. Recent studies (*e.g.* [48]) assess that recovered individuals which have previously contracted the virus have more chance to develop very high antibodies' level in order to fight against possible infections. This is not surprising: indeed, even the first dose of a two-dose vaccination cycle has the goal of creating a first contact with virus cells, whilst the second administration is responsible for the actual immunisation. Hence, the first effect is skipped in individuals which have just encountered the disease. The mild change in the model reflects in the number of administrations necessary to recovered individuals to be completely immunized: after the first administration, recovered individuals gain direct immunity, without passing through the V class.

The SEIHRDVW2 model reads as follows:

$$\begin{cases} \dot{S} = -\beta \dfrac{SI}{N} - \rho_S S + \mu_R R + \mu_V V \\ \dot{E} = \beta \dfrac{(S + \sigma V) I}{N} - \alpha E \\ \dot{I} = \alpha E - \gamma I \\ \dot{H} = \gamma I - \omega H \\ \dot{R} = (1 - f(S,V))\, \omega H - \mu_R R - \rho_R R \\ \dot{D} = f(S,V)\, \omega H \\ \dot{V} = -\beta \dfrac{\sigma V I}{N} + \rho_S S - \mu_V V - \rho_V V \\ \dot{W} = \rho_V V + \rho_R R \end{cases} \tag{4.4}$$

with fatality function defined as

$$f(S,V) = \bar{f} \frac{S(t-15) + \theta \sigma V(t-15)}{S(t-15) + \sigma V(t-15)}. \tag{4.5}$$

Similarly to SEIHRDVW1 model, we compute the reproduction number through the Next Generation Matrix approach. We obtain:

$$R_t = \frac{\beta}{\gamma} \frac{(S(t) + \sigma V(t))}{N}. \tag{4.6}$$

4.2 Optimal Control Problems (OCP)

With the aim of finding optimal vaccination policies, we are going to define optimal control problems for both models SEIHRDVW1 and SEIHRDVW2 as well as the different cost functionals that will be used in next chapters. The control variables are the amount of first and second doses adminstered to susceptibles and recovered individuals. Indeed, vaccination acts in the ODEs in three terms which are subtracted in the evolution of S, R and V variables, *i.e.*

1. $\rho_S S$, takes into account for daily first doses administered to susceptibles;
2. $\rho_R R$, takes into account for daily first doses adminstered to recovered;
3. $\rho_V V$, is the number of daily second doses administrations.

Some works dealing with optimal vaccination policies for epidemic models use as control variables the vaccination rates ρ_* (see, *e.g.*, [119], [59], [117]). We use a different approach by defining the control variables in the following way:

1. $U_1(t) = \rho_S S(t)$, with $U_1(t) \geq 0$;
2. $U_2(t) = \rho_R R(t)$, with $U_2(t) \geq 0$;
3. $U_3(t) = \rho_V V(t)$, with $U_3(t) \geq 0$.

In this way each control variable represents the daily amount of doses administered. We deal with two different constrained formulation, one assuming that the total amount of doses per day is fixed, the other assuming to have a fixed weekly amount of doses and a maximum adminstration capacity per day. To summarize, we would like to answer to the following questions:

Is it better to have more individuals with a partial coverage given by one single administration or to take as much people as possible to complete immunity in a specified timeframe? Does the optimal vaccination policy change when different performance measures are applied?

4.2.1 Cost functionals

We overview three different cost functionals that have been employed in Chapters 6 and 7. These three cost functionals have been chosen since they represent different aspects that stakeholders such as policy makers would want to control during pandemic events.

- $\int_0^{T_f} \boldsymbol{I(t)^2}\ \mathbf{d}\boldsymbol{t}$: taking into account for this cost functional means to monitor over the whole simulation timeframe the level of infected. The minimum achievable is $I(t) = 0$ over the whole time, even though there is strict dependance on initial conditions. If $\mathcal{R}_0 \geq 1$ infected will always have an increasing behaviour at the beginning, reaching a positive peak of infectious.
- $\int_0^{T_f} \boldsymbol{I(t)^2 + \dot{E}(t)^2}\ \mathbf{d}\boldsymbol{t}$: this cost functional takes into account for variations of slopes in the exposed curve together with populosity of the infected class. Actually, the $\dot{E}$-component has no explicit interpretation on its own, since the minimum solution achievable is a constant solution for the exposed variable which could also assest to an high positive value. On the other hand this component coupled with the above cost functional lead us to monitor contemporarily infected values and variations of the exposed curve, obtaining a more stable and robust solution.
- $\boldsymbol{D(T_f)^2}$: minimizing deceased individuals at the end of the simulation time is the third considered cost functional. Since deceased curve is always increasing, one should obtain similar optimal policy with respect to the case of minimization of $J = \int_0^{T_f} D(t)^2\,\mathrm{d}t$.

4.2.2 Control Formulations with SEIHRDVW models as State Problems

In this section we are going to formulate optimal control problems. The first tries to answer to the following question:

Assume to have a fixed daily amount of doses to administer. What is the optimal ripartition of doses among first doses for susceptibles and recovered and second doses?

The corresponding optimal control problem is formulated as follows:

Optimal Control Problem 1.

$$\arg\min_{U_1, U_2, U_3} J(S,\, E,\, I,\, H,\, R,\, D,\, V,\, W) \tag{4.7}$$

*under the state evolution (**SEIHRDVW1** model)*

$$\begin{cases} \dot{S} = -\beta\dfrac{SI}{N} - U_1 + \mu_R R + \mu_V V \\ \dot{E} = \beta\dfrac{(S+\sigma V)I}{N} - \alpha E \\ \dot{I} = \alpha E - \gamma I \\ \dot{H} = \gamma I - \omega H \\ \dot{R} = (1 - f(S,V))\,\omega H - \mu_R R - U_2 \\ \dot{D} = f(S,V)\,\omega H \\ \dot{V} = -\beta\dfrac{\sigma V I}{N} + U_1 + U_2 - U_3 - \mu_V V \\ \dot{W} = U_3, \end{cases} \tag{4.8}$$

*or (**SEIHRDVW2** model)*

$$\begin{cases} \dot{S} = -\beta\dfrac{SI}{N} - U_1 + \mu_R R + \mu_V V \\ \dot{E} = \beta\dfrac{(S+\sigma V)I}{N} - \alpha E \\ \dot{I} = \alpha E - \gamma I \\ \dot{H} = \gamma I - \omega H \\ \dot{R} = (1 - f(S,V))\,\omega H - \mu_R R - U_2 \\ \dot{D} = f(S,V)\,\omega H \\ \dot{V} = -\beta\dfrac{\sigma V I}{N} + U_1 - U_3 - \mu_V V \\ \dot{W} = U_2 + U_3. \end{cases} \tag{4.9}$$

Control variables have to fulfill the following external constraints:

- U_1, U_2, $U_3 \in [0; U_b]$, *where* U_b *is the upper bound of administrations per day that can also vary in time;*
- $U_1(t)+U_2(t)+U_3(t) \leq U_b \, \forall \, t \in (0; T]$, *meaning that the sum of all administrations cannot overcome a maximum daily value;*
- $U_1(t) \leq S(t), \forall \, t \in (0; T]$, *namely we cannot administer more first doses to susceptibles than the number of susceptibles itself;*
- $U_2(t) \leq R(t), \forall \, t \in (0; T]$, *namely we cannot administer more first doses to recovered than the number of recovered itself;*
- $U_3(t) \leq V(t), \forall \, t \in (0; T]$, *i.e we cannot administer second doses to individuals who have not received the first administration;*
- $\int_0^t U_3(\tau) \, \mathrm{d}\tau \geq \int_0^{t-t_{max}} U_1(\tau) + U_2(\tau) \, \mathrm{d}\tau \;\; \forall t \geq t_{max}$ *(respectively* $\int_0^t U_3(\tau) \, \mathrm{d}\tau \geq \int_0^{t-t_{max}} U_1(\tau) \, \mathrm{d}\tau \;\; \forall t \geq t_{max}$ *for the* `SEIHRDVW2` *model), meaning that second doses must be at least greater than the first doses administered* t_{max} *days before (*t_{max} *is indeed the maximum elapsing time between first and second doses administrations);*
- $\int_0^t U_3(\tau) \, \mathrm{d}\tau \leq \int_0^{t-t_{min}} U_1(\tau) + U_2(\tau) \, \mathrm{d}\tau \;\; \forall t \geq t_{min}$ *(respectively* $\int_0^t U_3(\tau) \, \mathrm{d}\tau \leq \int_0^{t-t_{min}} U_1(\tau) \, \mathrm{d}\tau \;\; \forall t \geq t_{min}$ *for the* `SEIHRDVW2` *model), meaning that second doses must not be greater than the first doses administered* t_{min} *days before (*t_{min} *is the minimum elapsing time between first and second doses administrations).*

The second optimization problem to formulate is the mathematical translation of the following question:

Assume to have a fixed weekly amount of doses to administer and a maximum administration capacity per day. What is the optimal ripartition of doses among first doses for susceptibles and recovered and second doses?

The optimization problem reads as follows:

Optimal Control Problem 2.

$$\arg\min_{U_1, U_2, U_3} J(S,\, E,\, I,\, H,\, R,\, D,\, V,\, W) \tag{4.10}$$

under the state evolution (`SEIHRDVW1` model)

$$\begin{cases} \dot{S} = -\beta \dfrac{SI}{N} - U_1 + \mu_R R + \mu_V V \\ \dot{E} = \beta \dfrac{(S + \sigma V) I}{N} - \alpha E \\ \dot{I} = \alpha E - \gamma I \\ \dot{H} = \gamma I - \omega H \\ \dot{R} = (1 - f(S,V))\, \omega H - \mu_R R - U_2 \\ \dot{D} = f(S,V)\, \omega H \\ \dot{V} = -\beta \dfrac{\sigma V I}{N} + U_1 + U_2 - U_3 - \mu_V V \\ \dot{W} = U_3, \end{cases} \tag{4.11}$$

or (`SEIHRDVW2` model)

$$\begin{cases} \dot{S} = -\beta \dfrac{SI}{N} - U_1 + \mu_R R + \mu_V V \\ \dot{E} = \beta \dfrac{(S + \sigma V) I}{N} - \alpha E \\ \dot{I} = \alpha E - \gamma I \\ \dot{H} = \gamma I - \omega H \\ \dot{R} = (1 - f(S,V))\, \omega H - \mu_R R - U_2 \\ \dot{D} = f(S,V)\, \omega H \\ \dot{V} = -\beta \dfrac{\sigma V I}{N} + U_1 - U_3 - \mu_V V \\ \dot{W} = U_2 + U_3. \end{cases} \tag{4.12}$$

Control variables have to fulfill the following external constraints:

- $\int_{T_j}^{T_{j+1}} U_1 + U_2 + U_3 \, dt \leq N_{DosesDeliveredPerWeek}$, *where* $T_j = 7j$ *where* $j = 0, \ldots, M_{weeks}$ *number of weeks;*
- $U_1(t) + U_2(t) + U_3(t) \leq C$, *where* C *stands for the maximum administration capacity per day, which can depend on medical staff availability, administration centers and other capabilities constraints;*

- $U_1(t) \leq S(t), \forall\, t \in (0;T]$, *namely we cannot administer more first doses to susceptibles than the number of susceptibles itself;*
- $U_2(t) \leq R(t), \forall\, t \in (0;T]$, *namely we cannot administer more first doses to recovered than the number of recovered itself;*
- $U_3(t) \leq V(t), \forall\, t \in (0;T]$, *i.e we cannot administer second doses to individuals who have not received the first administration;*
- $\int_0^t U_3(\tau)\,\mathrm{d}\tau \geq \int_0^{t-t_{max}} U_1(\tau) + U_2(\tau)\,\mathrm{d}\tau \;\; \forall t \geq t_{max}$ *(respectively* $\int_0^t U_3(\tau)\,\mathrm{d}\tau \geq \int_0^{t-t_{max}} U_1(\tau)\,\mathrm{d}\tau \;\; \forall t \geq t_{max}$ *for the* `SEIHRDVW2` *model), meaning that second doses must be at least greater than the first doses administered* t_{max} *days before (*t_{max} *is indeed the maximum elapsing time between first and second doses administrations);*
- $\int_0^t U_3(\tau)\,\mathrm{d}\tau \leq \int_0^{t-t_{min}} U_1(\tau) + U_2(\tau)\,\mathrm{d}\tau \;\; \forall t \geq t_{min}$ *(respectively* $\int_0^t U_3(\tau)\,\mathrm{d}\tau \leq \int_0^{t-t_{min}} U_1(\tau)\,\mathrm{d}\tau \;\; \forall t \geq t_{min}$ *for the* `SEIHRDVW2` *model), meaning that second doses must not be greater than the first doses administered* t_{min} *days before (*t_{min} *is the minimum elapsing time between first and second doses administrations).*

Solutions of both Problems 1 and 2 are the vaccination policies minimizing the performance measure J.

Chapter 5

Numerical methods

Introduce the following abstract optimization problem:

$$\arg\min_{x} f(x) \text{ subject to } \dot{x} = A(x) \text{ and } x \in C, \tag{5.1}$$

where f is a convex and smooth cost functional, $A(x)$ is the differential system ruling evolution of the vectorial variable x and C is a convex set subset of $\mathbb{R}^n$. In order to face numerically Problem (5.1), we state a Meta-Algorithm which addresses the main issues to cope with during the optimization process:

Algorithm 1 Optimization Meta-Algorithm

Input: Initial guess for control variables.
1. Solve ODE systems associated with state problem and costate problem (multipliers).
2. Apply iteratively an optimization method for dealing with constrained optimization problems. This stage ends with proper stopping criteria.
Output: State variables and control variables at the numerical optimum.

This chapter details the Meta-Algorithm through the description of numerical methods employed at each step.

Section 5.1 shortly reviews Runge-Kutta Methods we use for solving ODE systems of equations. Section 5.2 introduces the Projected Gradient Descent method, the Multi-Projection Algorithm and projection operators we need for assembling the optimisation method. In Section 5.3 we define the complete numerical formulation of optimization problems defined in Chapter 4 and we present the adopted Multi-Projected Gradient Descent optimization algorithm.

5.1 Runge-Kutta 4 method for ODE systems

The ODE-system solver that we used in this work implements the *Runge-Kutta* method of order 4 that we shortly review. Refer to [95] for more details.

Define the following ODEs problem,

$$\dot{\boldsymbol{u}}(t) = \boldsymbol{F}(t, \boldsymbol{u}),\ \forall t \in (0; T]. \tag{5.2}$$

Define a time discretization $\{t_n\}_{n \leq N \in \mathbb{N}}$ and $\boldsymbol{u}_n \approx \boldsymbol{u}(t_n)$, $n \leq N \in \mathbb{N}$, approximations of the unknown function at grid points. These values are the unknowns of Runge-Kutta methods (RK), which mantain the structure of one-step methods increasing accuracy at the price of getting more functional evaluations at each timestep, and losing linearity properties.

Assuming the time sequence is equally spaced and fixing $h = t_{n+1} - t_n$, the more general form of a RK method is the following,

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h\, \boldsymbol{G}(t_n, \boldsymbol{u}_n, h; \boldsymbol{F}),\ n \geq 0, \tag{5.3}$$

where $\boldsymbol{G}$ is the *increment function* defined as

$$\begin{aligned} \boldsymbol{G}(t_n, \boldsymbol{u}_n, h; \boldsymbol{F}) &= \sum_{i=1}^{s} b_i \boldsymbol{K_i}, \\ \boldsymbol{K_i} &= \boldsymbol{F}(t_n + c_i h, \boldsymbol{u}_n + h \sum_{j=1}^{s} a_{ij} \boldsymbol{K_j}),\ i = 1, \ldots, s. \end{aligned} \tag{5.4}$$

Values of the coefficients $\{a_{ij}\}$, $\{b_i\}$ and $\{c_i\}$ completely define the RK method and are collected in the so-called *Butcher array* [95]. The number of stages of the Runge-Kutta methods is s. If a_{ij} are null when $j \geq i$, with $i \leq s$, then $\boldsymbol{K}_i$ can be explicitly computed in terms of $\boldsymbol{K}_j$ with $j < i$, and therefore the method is explicit.

Define the local truncation error $\tau_{n+1}(h)$ at node t_{n+1} as

$$h\tau_{n+1}(h) = \boldsymbol{y}_{n+1} - \boldsymbol{y}_n - h\boldsymbol{G}(t_n, \boldsymbol{y}_n, h; \boldsymbol{F}), \tag{5.5}$$

with $\boldsymbol{y}(t)$ exact solution of (5.2). RK method is said to be **consistent** if

$$\tau(h) = \max_n |\tau_{n+1}(h)| \to 0 \,\text{if}\, h \to 0. \tag{5.6}$$

In [73] it is proved that this happens if and only if

$$\sum_{i=1}^{s} b_i = 1. \tag{5.7}$$

The method is said to be **convergent** of order p if $\tau(h) = \mathcal{O}(h^p)$ if $h \to 0$. As for convergence, since RK are one-step methods, consistency implies stability, and, in turn, convergence. It is possible to prove that if the local truncation error $\tau_n(h) = \mathcal{O}(h^p)$ for any n, then also convergence order is p [95]. The following theorem can be deduced:

Theorem 9. *The order of an s-stage explicit RK method cannot be greater than s. Also, RK methods with order s do not exist if* $s \geq 5$.

Runge-Kutta method of order 4 which will be further adopted reads as follows:

Algorithm 2 Runge-Kutta 4

Input: Initial condition $\boldsymbol{u}_0 = \boldsymbol{u}(0)$.
For $n = 0, \dots, N-1$, compute $\boldsymbol{K}_1$, $\boldsymbol{K}_2$, $\boldsymbol{K}_3$ and $\boldsymbol{K}_4$:

$$\begin{aligned}
\boldsymbol{K}_1 &= \boldsymbol{F}_n, \\
\boldsymbol{K}_2 &= \boldsymbol{F}\left(t_n + \frac{h}{2}, \boldsymbol{u}_n + \frac{h}{2}\boldsymbol{K}_1\right) \\
\boldsymbol{K}_3 &= \boldsymbol{F}\left(t_n + \frac{h}{2}, \boldsymbol{u}_n + \frac{h}{2}\boldsymbol{K}_2\right) \\
\boldsymbol{K}_4 &= \boldsymbol{F}(t_{n+1}, \boldsymbol{u}_n + h\boldsymbol{K}_3)
\end{aligned}$$

The approximated solution at each point is computed as follows:

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \frac{h}{6}(\boldsymbol{K}_1 + 2\boldsymbol{K}_2 + 2\boldsymbol{K}_3 + \boldsymbol{K}_4)$$

5.2 Optimization procedure: Projected Gradient Descent (PGD) and Multi-Projection Algorithm

We define a descending algorithm (Projected Gradient Descent Algorithm) and projection operators for dealing with constraints introduced in OC Problems 1 and 2. Moreover, we need a Multi-Projection Algorithm to assure that multiple constraints are fulfilled contemporarily.

5.2.1 Projected Gradient Descent Algorithm

We introduce the Projected Gradient Descent Algorithm in the simpler context of the minimization of function $f : \mathbb{R}^n \to \mathbb{R}$ on a constrained set C, namely neglecting state evolution ruled by ODEs.

PGD starts from a given initial guess $x^{(0)}$, and updates for $k = 1, \dots, N_{maxit}$ by first performing typical gradient descent step, and then projecting back the solution to the constraint set C. The algorithm is formalized as follows:

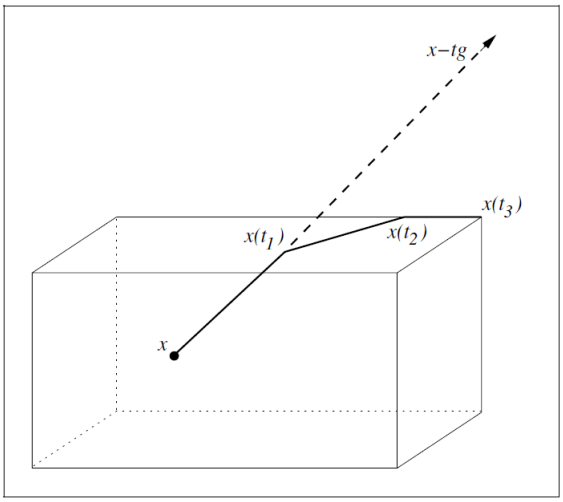


Figure 5.1: Projected Gradient Descent step in $\mathbb{R}^3$. The constrained set C is represented by the rectangular parallelepiped. Source: [86].

Algorithm 3 Projected Gradient Descent

Input: Initial guess $x^{(0)}$.
For $k = 1, \dots, N_{maxit}$ and while $err > tol$, compute

$$x^{(k)} = P_C(x^{(k-1)} - t_k \nabla f(x^{(k-1)})), \tag{5.8}$$

with P_C projecting operator on C, and t_k adaptable step.

Output: $x^{(K)}$, coordinate which minimizes the cost functional satisfying the constraint ($K = N_{maxit}$ if the stopping criterium fails at each iteration).

The algorithm stops whether the maximum number of iterations is achieved or a stopping criterium is satisfied, *i.e.* we compute a suitably-defined error at each iteration and we compare its value with a specified tolerance. Most of the times, a plausible measure of the error is the magnitude of the gradient of the function itself.

A special case of proximal gradient, motivated by local quadratic approximation of convex function f is

$$x^{(k)} = P_C\left(\left\{\arg\min_y \nabla f(x^{(k-1)})^T(y - x^{(k-1)}) + \frac{1}{2t}\|y - x^{(k-1)}\|_2^2\right\}\right). \tag{5.9}$$

Note that Algorithm 3 depends strictly on the choice and performances of the projection operator that is implemented. The *Frank-Wolfe method* [14] is an alternative way to deal with this kind of problems overcoming the definition of the projection operator. PGD algorithm is a local optimization method, as well as the Gradient descent method, hence we achieve local, rarely global, optima. The following Theorem derived in [63] infers on convergence properties of this method:

Theorem 10. *If f is convex, PGD algorithm with constant stepsize α satisfies the following extimate*

$$f\left(\frac{1}{K+1}\sum_{k=0}^{K} x^{(k)}\right) - f^* \leq \frac{\|x^{(0)} - x^*\|_2^2}{2\alpha(K+1)} + \frac{\alpha}{2(K+1)}\sum_{k=0}^{K}\|\nabla f(x^{(k)})\|_2^2, \tag{5.10}$$

where f^* is the cost functional evaluation at local optimum, x^* is the minimizer, α is the constant stepsize, K the total number of performed iterations.

The term $\frac{1}{K+1}\sum_{k=0}^{K} x^{(k)}$ is the average of a sequence of iterations after K iterations. Hence, defining it as $\bar{x}$, equation (5.10) can be seen as

$$\bar{f} - f^* \leq \frac{\|x^{(0)} - x^*\|_2^2}{2\alpha(K+1)} + H, \tag{5.11}$$

where H is a positive constant. Convergence rate is like $\mathcal{O}(\frac{1}{K})$, as long as $\sum_{k=0}^{K}\|\nabla f(x^{(k)})\|_2^2$ does not grow or increases slower than K.

If the cost functional is a Lipschitz function, convergence is of the order $\mathcal{O}(\frac{1}{\sqrt{K}})$ as stated in the following theorem:

Theorem 11. *Let f be a Lipschitz function. For the point $\bar{x}_K = \frac{1}{K+1}\sum_{k=0}^{K} x^{(k)}$, and constant stepsize $\alpha = \frac{\|x^{(0)} - x^*\|}{L\sqrt{K+1}}$, we have*

$$f(\bar{x}_K) - f^* \leq \frac{L\|x^{(0)} - x^*\|}{\sqrt{K+1}} \tag{5.12}$$

Proof. Put $\bar{x}_K$ and α into Theorem 10 directly and note that $\|\nabla f\| \leq L$. □

This is obviously a theoretical and not practical theorem, since the stepsize required implies knowing directly the minimum of the control problem.

5.2.2 Projection Operators

Consider Optimal Control Problems 1 and 2 for the two already introduced `SEIHRDVW` versions. As we have previously noted, U_{ad}, space of admissible constraints, is not unbounded; therefore, PGD scheme in (5.8) need suitable projection operators, on which performances will depend strictly. Two different kinds of constraints arise during the discretization process:

1. Box-like constraints: $U_* \leq U_b$ and $U_* \geq L_b$;
2. Linear constraints on the simplex: $\sum_{j=0}^{k} U_j \leq U_b$ or $\sum_{j=0}^{k} U_j \geq L_b$.

The second kind of constraint arises from the discretization in time of integral constraints. We detail the projection operators for both cases:

1. Box-like projectors can be directly implemented by their definition. Indeed, assume to have a constraint as follows,

$$L_b \leq U_* \leq U_b, \tag{5.13}$$

a possible nonlinear projector is defined as U_b if $U_* \geq U_b$, L_b if $U_* \leq L_b$, U_* otherwise. In a compact way,

$$P_{bl}(U_*) = \max(\min(U_*, U_b), L_b). \tag{5.14}$$

2. The problem of computing linear projections can be seen as a Euclidean projection problem onto the positive simplex $\sum_{j=0}^{k} U_j \leq U_b,\ U_j \geq 0,\ j = 1, \ldots, m$.

Define β a vector in $\mathbb{R}^k$ that we aim at projecting in $\beta_i \geq 0,\ \forall i \leq k$ and $\beta^T \mathbf{1} \leq z$, with $z > 0$. We propose a method due to *Shalev-Shwartz* and *Singer* [101] based on sorting each of the β_i parameters to recover the projection on the simplex. The method takes $\mathcal{O}(k \log(k))$ iterations due to sorting procedure..

The algorithm is schematized below:

Algorithm 4 Euclidean projection Algorithm

Input: $\beta \in \mathbb{R}^k$, a scalar upperbound $z > 0$
Sort β into μ: $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_k$
Find $\rho = \max\{m \in [k] : \mu_m - \pi(m) > 0\}$
Output: β where $\beta_m^* = [\beta_m - \pi(\rho)]_+\ \forall m \in \mathbb{N},\ m \leq k$

The complexity can be reduced to expected $\mathcal{O}(k)$ time while maintaining exactness of the solution by using a randomized pivot algorithm. We refer to [13] for more details about the implementation of linear constraints.

5.2.3 Multi-Projection Algorithm

Suppose we have two convex sets U_{ad1} and U_{ad2} which are constituted by controls satisfying different constraints. Assume x is the feasible control which we want to project on $C \bigcap D$, namely we come up with a solution which satisfies contemporarily the two distinct constraints.

Suppose C and D are closed convex sets in $\mathbb{R}^n$, and let P_C and P_D projection operators on the two sets respectively. In [15], an alternating projection algorithm is introduced, and its convergence properties are proved.

Algorithm 5 Multi-Projection Algorithm

Input: Initial guess $x_0 \in C$.
Compute alternated projections in the following way:

$$y_k = P_D(x_k), \quad x_{k+1} = P_C(y_k), \quad k = 0, 1, \ldots, N_{maxit} - 1.$$

Output: $x_{N_{maxit}}$, projected solution.

Will the sequence of $\{x_k\}_{k\in\mathbb{N}}$ converge to the projection in the intersection of the two sets?

A general result shown in [27] is that, if $C \bigcap D \neq \emptyset$, the sequence converges to the projection $x^* \in C \bigcap D$ (see Figure 5.2). This is not assured in a finite number of step, therefore it has to be checked manually. When the two sets do not intersect it can be proven that

$$x_k \to x^*, \;\; y_k \to y^*, \;\; \|x^* - y^*\| = \mathbf{dist}(C, D). \tag{5.15}$$

In other words the algorithm produces a pair of points that have minimum distance among the two sets (see Figure 5.3).

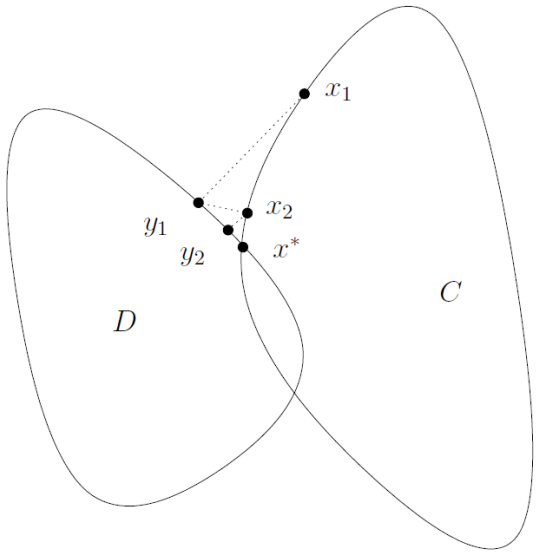


Figure 5.2: Subsequent steps of the Multi-Projection Algorithm when $C \bigcap D \neq \emptyset$. Source: [15].

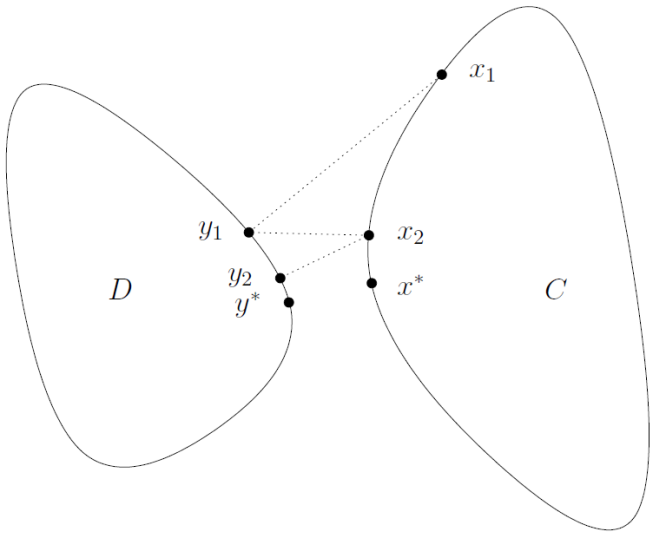


Figure 5.3: Subsequent steps of the Multi-Projection Algorithm when C and D are disjoint sets. Source: [15].

5.3 Numerical formulation and `MultiPGD`

Let us revise the Meta-Algorithm 1 introduced at the beginning of this chapter, detailing the numerical implementation of each step through the numerical methods presented in Sections 5.1 and 5.2:

1. Propose an initial guess of controls at each timestep

$$\boldsymbol{U_1}^{(0)} = \boldsymbol{U_1^0},\ \boldsymbol{U_2}^{(0)} = \boldsymbol{U_2^0},\ \boldsymbol{U_3}^{(0)} = \boldsymbol{U_3^0}. \tag{5.16}$$

   Define Initial Conditions for the state problem and the cost functional J to minimize. Set the tolerance (tol), step-length (λ) and the maximum number of iterations (N) for the PGD Algorithm, and the maximum number of iterations for the Multi-Projection Algorithm (M).

2. Assembling Projected Gradient Descent Method with the Multi-Projection Algorithm, we introduce and apply the following Multi-Projected Gradient Descent (`MultiPGD`).
   For $k = 1, \dots, N$:

   2a. Solve the state problem given $\boldsymbol{U_1}^{(k-1)}$, $\boldsymbol{U_2}^{(k-1)}$, $\boldsymbol{U_3}^{(k-1)}$ through the Runge-Kutta 4 method, namely

$$\boldsymbol{x}^{(k)} = \texttt{SEIHRDVWmodel}(\boldsymbol{U_1}^{(k-1)}, \boldsymbol{U_2}^{(k-1)}, \boldsymbol{U_3}^{(k-1)}). \tag{5.17}$$

   2b. Solve the costate (or multipliers problem) given $\boldsymbol{x}^{(k)}$, $\boldsymbol{U_1}^{(k-1)}$, $\boldsymbol{U_2}^{(k-1)}$, $\boldsymbol{U_3}^{(k-1)}$ through RK4, namely

$$\boldsymbol{p}^{(k)} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}(\boldsymbol{x}^{(k)}, \boldsymbol{U_1}^{(k-1)}, \boldsymbol{U_2}^{(k-1)}, \boldsymbol{U_3}^{(k-1)}). \tag{5.18}$$

   2c. Define the three operators P_1, P_2, P_3 projecting control variables on the constraints imposed by maximum daily administrations, maximum elapsing time among doses and minimum elapsing time among doses respectively (OC Problem 1 case). The same has to be defined when dealing with OC Problem 2 substituting the constraint on maximum daily administrations with the one related to weekly delivered doses and the one imposing maximum daily capacity. We introduce

$$\boldsymbol{U}^{(\cdot)} = [\boldsymbol{U_1}^{(\cdot)}, \boldsymbol{U_2}^{(\cdot)}, \boldsymbol{U_3}^{(\cdot)}]^T.$$

   2c.1. Define

$$\boldsymbol{r} = \boldsymbol{U}^{(k-1)} - \lambda \frac{\partial \mathcal{H}}{\partial \boldsymbol{U}}(\boldsymbol{x}^{(k)}, \boldsymbol{p}^{(k)}, \boldsymbol{U_1}^{(k-1)}, \boldsymbol{U_2}^{(k-1)}, \boldsymbol{U_3}^{(k-1)});$$

2c.2. Apply the Multi-Projection Algorithm.
For $s = 0, \ldots, M-1$ and fixing $\boldsymbol{d}_0 = \boldsymbol{r}$, compute the following quantities:

$$\boldsymbol{f}_s = P_1(\boldsymbol{d}_s), \quad \boldsymbol{g}_s = P_2(\boldsymbol{f}_s), \quad \boldsymbol{d}_{s+1} = P_3(\boldsymbol{g}_s).$$

2c.3

$$\boldsymbol{U}^{(k)} = \boldsymbol{d}_M,$$

and assign each component of $\boldsymbol{U}^{(k)}$ to the respective components of $\boldsymbol{U_1}^{(k)}$, $\boldsymbol{U_2}^{(k)}$, $\boldsymbol{U_3}^{(k)}$.

2d. Stopping criterium: compute the following incremental quotient,

$$\frac{|J(\boldsymbol{x}^{(k)}) - J(\boldsymbol{x}^{(k-1)})|}{|J(\boldsymbol{x}^{(k-1)})|}. \tag{5.19}$$

If this quantity almost vanishes, Pontryagin principle assures that local minimum is achieved.

Part II
Numerical Results

The second part is devoted to presenting and discussing numerical results obtained solving Optimal Control problems defined in Chapter 4. Our goal is to underline key-features of optimal vaccination policies in both artificial and in more realistic scenarios. The results are organized in two chapters:

- **Chapter 6** sets the simulations in artificial scenarios. From this set of simulations we aim at extracting qualitative guidelines to implement optimal vaccination campaigns;
- **Chapter 7** collects results framed in more realistic scenarios, *i.e.* extracting values for vaccine effectivenesses or the transmission rate function from actual available data referred to Lombardy in 2021.

We solve optimization problems choosing the `SEIHRDVW2` model as state problem, in accordance to the real vaccination policy adopted by Italian Regional Governments. Indeed, from recent scientific studies, recovered individuals have shown higher antibody levels after one single administration than completed vaccinated susceptibles. Therefore, in the Italian 67th Rapporto COVID [34], the Health Ministry suggested to administer one dose to individuals who had contracted the virus during the precedent 3-6 months, no administration whether the infection has happened in less than three months. Moreover, an Italian study carried out by Infectious Diseases and Microbiology Department of IRCCS *Sacro Cuore Don Calabria* [48] underlined that the later first dose is administered to recovered individuals the higher is their antibodies response. Notice that each result section is independent from the others.

Details about implementation

The code implementing the complete Meta-Algorithm (Algorithm 1) is written in `python3`. This programming language is particularly suited for optimization problems, since it is actually optimized for Machine Learning algorithms which require *ad hoc* optimization methods (for example SDG, GD or Momentum variant).

In particular, we used `jax` library which implements automatic differentiation, necessary for computing derivatives of the Hamiltonian function. Furthermore, the provided just-in-time compilation of functions (`jax.jit`) has led to cut computational times off. We exploited `pandas` library in order to extract averaged quantities from data available by the Italian department of Protezione Civile ([91] and [92]).

In Appendix C one can find two codes we implemented for both formulations of the optimization problems (OC Problem 1 and 2).

Chapter 6

Artificial scenarios

From this set of simulations we aim at extracting qualitative guidelines to implement optimal vaccination campaigns in different situations (choosing the `SEIHRDVW2` model as state problem).

Having in mind the goals of Chapter 7, in this Chapter we set parameters and initial conditions ideally assuming to be framed in Lombardy on the 1st January 2021, considered as the initial simulation day. More precisely, in Table 6.1 we find the initial conditions that have been set for the solution of the state problem. They correspond to feasible initial conditions extracted by data available from Italian department of Protezione Civile [91] with uncertainty mainly due to monitoring policy. In Table 6.2 we find values of model parameters that have been fixed for the simulations. These parameters come from biological studies of COVID-19 disease. Lastly, we set the step of the Runge-Kutta 4 method at $dt = 1$ day and the maximum number of iterations for the Multi-Projection Algorithm at $k = 30$. Other numerical parameters adopted for specific simulations are specified in the following sections.

Notice that simulations have been run setting the timeframe heterogeneously, although always a multiple of seven to allow the interpretation in terms of weeks.

S(0)	**E(0)**	**I(0)**	**H(0)**	**R(0)**	**D(0)**	**V(0)**	**W(0)**
9.62e6	2.5e4	5.5e4	5.05e4	4.01e5	2.52e4	1.78e3	0

Table 6.1: Initial Conditions (IC) for `SEIHRDVW` model. They have been deduced setting the problem on the 1st January 2021 in Lombardy.

The rest of the Chapter is structured as follows: in Section 6.1 we compare two optimal vaccination problems obtained minimizing infectious during the whole timeframe and the number of deceased at the final time, respectively; Section 6.2 collects results about optimization of parametrized UK and US-like vaccination policies; Section 6.3 aims at underlining differences in optimal vaccination policies when different social restrictions are introduced; finally, in Section 6.4 we compare optimal vaccination strategies imagining to deal with virus variants.

Parameter	Physical meaning	Value
$\boldsymbol{\alpha}$	Inverse of the incubation time	0.182
$\boldsymbol{\gamma}$	Infectious rate	0.211
$\boldsymbol{f}$	Fatality rate	2.794e-3
$\boldsymbol{\mu_R}$	Susceptible – Recovered passing rate	4.76e-3
$\boldsymbol{\mu_V}$	Susceptible – Vaccinated passing rate	4.76e-3
$\boldsymbol{\omega}$	Healing rate	0.0690
$\boldsymbol{\sigma}$	Fraction of vaccine-effectiveness on infection trasmission	0.25
$\boldsymbol{\theta}$	Fraction of vaccine-effectiveness on mortality	0.15
$\boldsymbol{t_{min}}$	Minimum elapsing time among consecutive doses	21
$\boldsymbol{t_{max}}$	Maximum elapsing time among consecutive doses	42

Table 6.2: Table of parameters for model `SEIHRDVW` for the SARS-CoV-2 case in Lombardy (extimated through available data recovered in Italy during the second pandemic wave in early October 2020).

6.1 Comparison between optimal strategies for minimizing deaths and infectious individuals

We compare optimal vaccination policies considering two different cost functionals defined as follows:

- **Case 1**: $J_1 = \int_0^{T_f} I(\tau)^2 \, \mathrm{d}\tau$;
- **Case 2**: $J_2 = D(T_f)^2$.

6.1.1 Methods and parameters

The results that are illustrated in the following subsection are obtained through the solution of Optimal Control Problem 1 formulated in Chapter 4. The control variables are the daily amount of first doses to susceptibles and recovered, and the amount of second doses. We fix the tolerance of the PGD algorithm at $1e-6$ and the maximum number of achievable iterations at 100. The step of PGD is fixed at $1e-1$. The transmissibility rate β is mantained fixed at 0.26712[1]. Maximum vaccination supply during the period of simulation is 10000 administrations per day. We simulate a timeframe of 280 days (namely 40 weeks). The approximation of neglecting inner and outer fluxes of people coming from different regions is implied.

[1]This is a plausible value for Lombardy in the period 01-02-2021/01-03-2021 (corresponding to the *yellow* level of social restrictions) for the considered model. We derived this value inverting $\mathcal{R}_t$ index definition for the `SEIHRDVW`, which was fit by data available from the Italian Department of Protezione Civile [91]. We recovered $\mathcal{R}_t$ values on the EpiMox dashboard (`https://www.epimox.polimi.it/`), which collects regional and national data and makes epidemic prevision of the italian scenario through the `SUIHTER` model.

6.1.2 Results

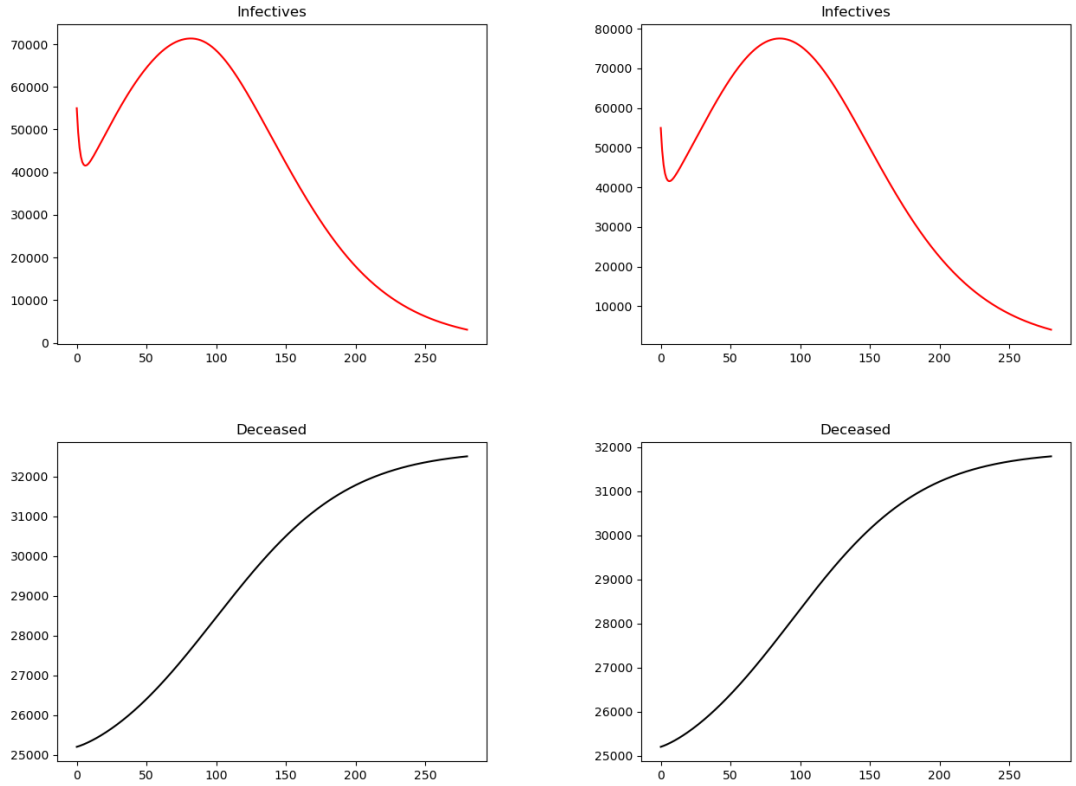


Figure 6.1: Evolution of infectious individuals and deceased obtained minimizing J_1 (infected individuals during the total timeframe, left figures) or minimizing J_2 (deaths at the final time, right figures).

Figure 6.1 refers to the evolution of infectious individuals and deceased cases for the two cost functionals (respectively, left figures refer to minimization of infected cases, right figures refer to the minimization of deceased). Both cases reach 100 iterations without achieving the desired tolerance.

As one expects, the curve of infectious in Case 1 reaches lower values with respect to Case 2, whereas the converse is true for the curve representing deceased. For the Case 1, we reach the infectious peak amounting at almost 70000 individuals on the 86th day of simulation. In Case 2, the infectious peak is delayed to the 90th day and amounts at almost 79000 infectious. After the infectious peak, both curves are monotonically decreasing. We note that, administering 10000 daily doses, the epidemic is not completely eradicated after 280 days from the beginning of the vaccination campaign. Deceased assest at 32500 in Case 1, 32000 in Case 2.

In Figure 6.2 we focus on the ripartition of doses for the three control variables, namely U_1 first doses administered to susceptibles, U_2 doses administered to recovered,

U_3 daily administered second doses. Since the maximum value of administrations per day is mantained constant, these pictures are interpretable as daily percentage-ripartitions among the three different controls.

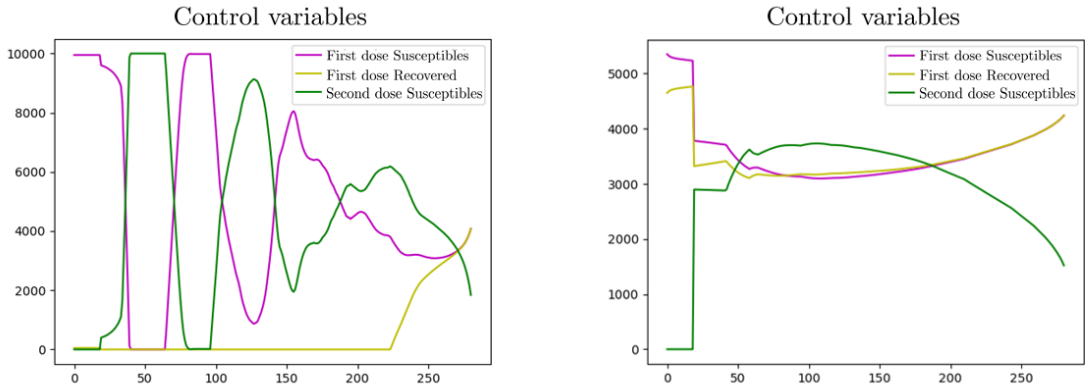


Figure 6.2: Output controls obtained minimizing infected individuals during the total timeframe (left figures) or minimizing deaths at the final time (right figures).

For what concerns Case 1, political authorities in charge of managing the vaccination campaign can interpret results in Figure 6.2 as the suggestion that, in order to minimize the infectious curve in a fixed timeframe, the optimal strategy is to alternate between first and second doses for susceptible individuals over a typical period of about 42 days, and then consistently vaccinate individuals who recovered from the virus.

The vaccination strategy is different when one wants to minimize deceased at the final time. After the first 20 days where second doses administrations are not allowed, 30% of daily total doses is administered as first doses to susceptibles and another 30% to recovered. The remaining 40% is constituted by second doses. The former distribution of doses is valid during the period 20-150 days, following which second doses decrease up to 15% and first doses administered to susceptibles and recovered increase up to 42.5%.

Although with the second vaccination strategy more individuals are infected, less individuals decease due to the infection. We explain this in light of the different vaccination policies for the two different cost functionals. Indeed, the amount of people who completed the vaccination cycle is $1.75e6$ with the second vaccination strategy, almost $1.4e6$ with the first (see Figure 6.3). We conclude that solutions which aim at vaccinating completely more individuals are more effective for minimizing deceased individuals in the long period, whereas providing more individuals with a partial coverage is a preferred solution when we aim at minimizing infectious individuals (*i.e.* stopping the transmission cycle).

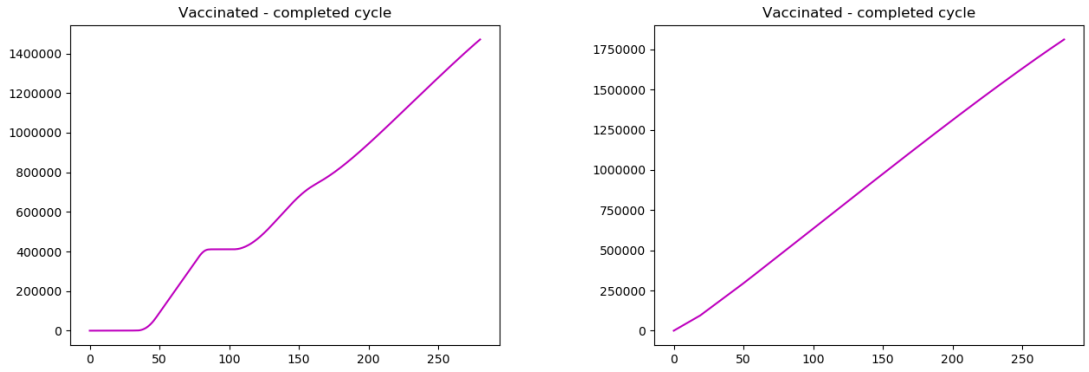


Figure 6.3: Completed vaccinated evolutions minimizing infected individuals during the total timeframe (left figure) or minimizing deaths at the final time (right figure).

6.2 Optimization of paradigmatic vaccination policies

This section deals with the analysis of paradigmatic scenarios deducible from vaccination policies actually implemented in the United Kingdom (UK) and in the United States of America (USA). Our goal is to express approximations of the vaccination policies through parametric functions and then to optimize the solutions of the `SEIHRDVW2` model with imposed vaccinations by varying the parameters themselves.

Among the other countries in the list of those which have administered more doses with respect to the population until June 2021, UK and USA implemented vaccination strategies that can be effortless parametrized. Note that, in this section, no optimal control problem is solved, and so no PGD algorithm is needed. Indeed, we analyze solutions of direct problems with the `SEIHRDVW2`, and we measure performances of each simulation through the evaluation of four different quantities.

6.2.1 Methods and parameters

We solve the direct problem corresponding to Model 4.4 through the Runge-Kutta 4 method. The final time of simulation is fixed at 140 days, *i.e.* 20 weeks. We impose the daily amount of first and second doses following parametric functions which approximate vaccination policies of the two countries. Moreover, first doses are split among recovered and susceptibles following the proportion $\frac{1}{4}$ and $\frac{3}{4}$. Every day 10000 vaccine doses are administered. The transmission rate β is fixed at 0.2612 as in the previous case, making this analysis an artificial non-realistic scenario.

6.2.2 Results

UK-like strategy

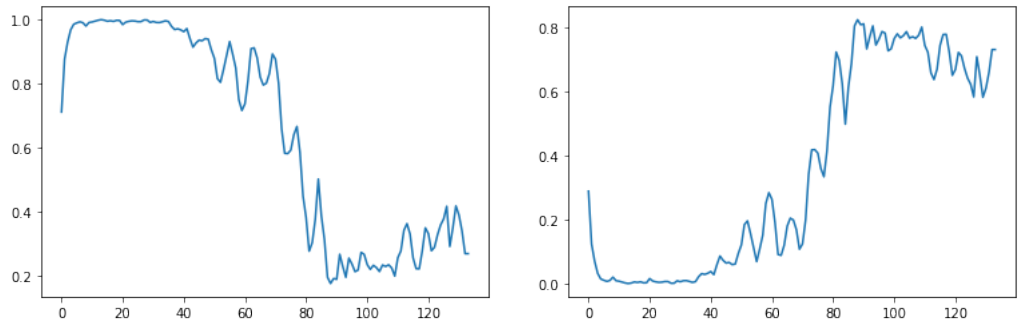


Figure 6.4: Percentage of administrations of first (above figure) and second doses (below figure) in UK from 11th January 2021. Data source: [97]

Vaccination program started in UK on the 8th of December 2020, when an elder british woman became the first person in the world to receive a clinically authorised vaccine for COVID-19. At the time of writing, UK has authorized six different vaccines developed with different approaches: the Oxford (ChAdOx1 nCoV-19) and Janssen (Ad26.COV2.S) vaccines, made from a genetically engineered virus; a vaccine developed by BioNTech/Pfizer (BNT162b1), which uses a novel approach of injecting part of the virus' mRNA; a vaccine created by Valneva (VLA2001), which uses an inactive version of the virus; a vaccine created by Novavax (NVX-CoV2373) and one under development by GlaxoSmithKline/Sanofi Pasteur, both using protein adjuvants to stimulate an immune response.

In Figure 6.4 we represent the percentage of first and second doses administered in UK starting from the 11th January for 133 days, recovered from data available at [97]. We note that, for almost 80 days from the starting date, primarily first doses have been adminstered, delaying consistent second doses administrations.

From the actual percentages of distribution of doses, we extract a parametrized function for the percentage of first doses administrations in a time interval of 140 days:

$$f_{1,T}(t) = \begin{cases} 1, & \text{if } t \in [n\,T; (n+1)\,T) \text{ for } n = 0, 2, 4, \dots \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

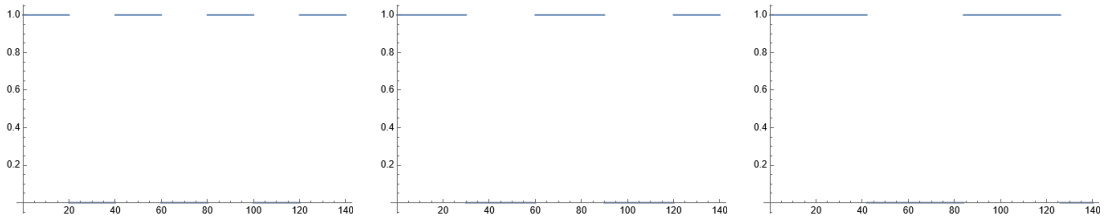
Percentage of adminstered second doses is defined as a piecewise constant function in counter-phase with respect to first doses administrations:

$$f_{2,T}(t) = \begin{cases} 1, & \text{if } t \in [m\,T; (m+1)\,T) \text{ for } m = 1, 3, 5, \dots \\ 0, & \text{otherwise.} \end{cases} \tag{6.2}$$

Figures 6.5 and 6.6 depict three cases of the family of functions representing first and second doses percentages depending on T.

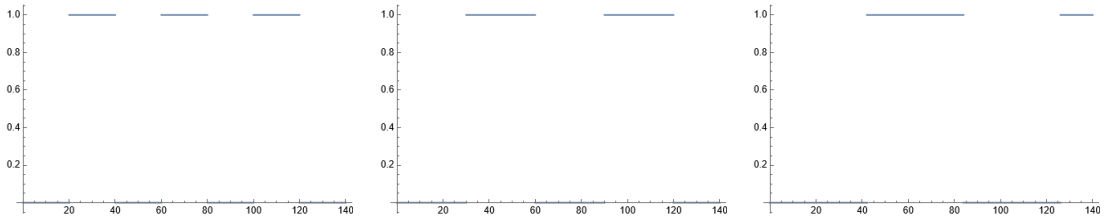
The question we want to face is the following:

What is the optimal frequency (and so the optimal period) for strategies all-in/all-out like this for different cost functionals?



(a) $T = 20$ **(b)** $T = 30$ **(c)** $T = 42$

Figure 6.5: First doses percentage extrapolated from UK policy as a piecewise constant function with alternated values 0 and 1. The period of the three figures is $T = 20$, 30 and 42 days respectively.



(a) $T = 20$ **(b)** $T = 30$ **(c)** $T = 42$

Figure 6.6: Second doses percentage extrapolated from UK policy as a piecewise constant function with alternated values 0 and 1. The period of the three figures is $T = 20$, 30 and 42 days respectively.

We consider the period of the vaccination strategy (T) as the control parameter for an optimization problem with fixed daily administrations according to (6.1) and (6.2):

Optimal Control Problem.

$$\min_{T \in \mathbb{N}} J,$$

with state problem (4.4)*, where* $U_1(t) = U_b\, p\, f_{1,T}(t)$*,* $U_2(t) = U_b\,(1-p)\, f_{1,T}(t)$ *and* $U_3(t) = U_b\, f_{2,T}(t)$*, where* p *is the fractional ripartition of first doses among susceptibles and recovered individuals and* U_b *is the total daily amount of available vaccine jabs.*

J is the performance measure we choose to minimize. Notice that we are working in the scenario with constant daily amount of doses. If we admit that the same quantity varies on time we could obtain different results.

We solve the direct problem corresponding to `SEIHRDVW2` model in a time interval of 140 days, letting T, called *jumping period*, vary in $[20, 42]$. For all $T \in [20, 42]$ and $T \in$

$\mathbb{N}$ the vaccination strategy matches constraint imposed by maximum and minimum elapsing time between doses for vaccines with properties analogous to Pfizer/BioNTech. We evaluate four quantities for each simulation:

- $J = D(T_f)^2$, referred to as *Deaths*;
- $J = \int_0^{T_f} \dot{E}(t)^2 dt$, referred to as *Exposed*;
- $J = \int_0^{T_f} \dot{I}(t)^2 dt$, referred to as *Infected*;
- $J = \int_0^{T_f} \mathcal{R}_t(t)^2 dt$, referred to as *Rt*.

Figure 6.7 depicts their normalized values with respect to the frequency T on the x-axis.

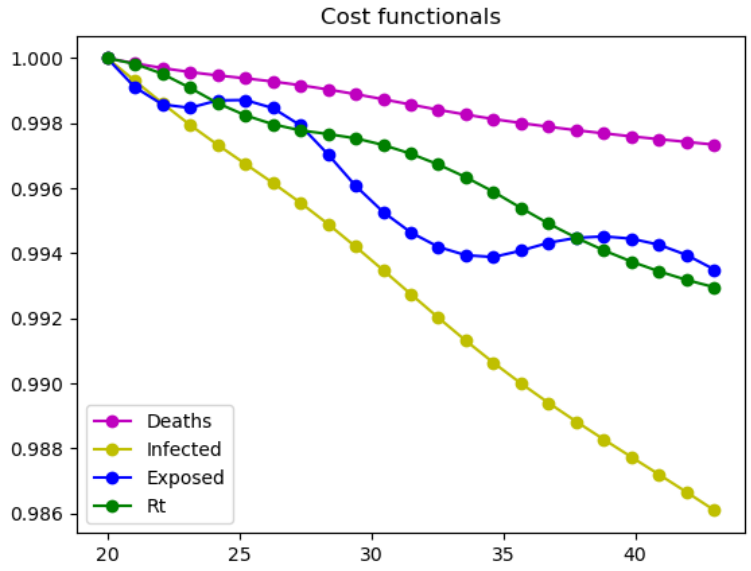


Figure 6.7: Normalized values of each performance measure for the UK-like vaccination strategy. The typical period T is the independent coordinate x.

For quantities named *Deaths, Infected* and *Rt*, increasing typical period of the strategy from 20 to 42 leads to their relative reduction. This fact assures that, among the possible scenarios, and remembering that the maximum elapsing time for doses of AstraZeneca vaccine (which is the most adopted vaccine in the UK) is almost 80 days, the strategy implemented in Great Britain (Figure 6.4) is optimal.

Fluctuations appear evidently in the blue and in the green curves. The former implies that $\mathcal{R}_t$ has local minima at $T \simeq 25, 35$ and local maxima at $T \simeq 30, 40$. Instead, the blue curve, corresponding to the *Exposed* variation, is not a monotone decreasing function. This is not a concerning result since this quantity has no physical interpretation on its own, but it has to be strictly related to the yellow curve which monotonely decreases as T increases.

US-like strategy

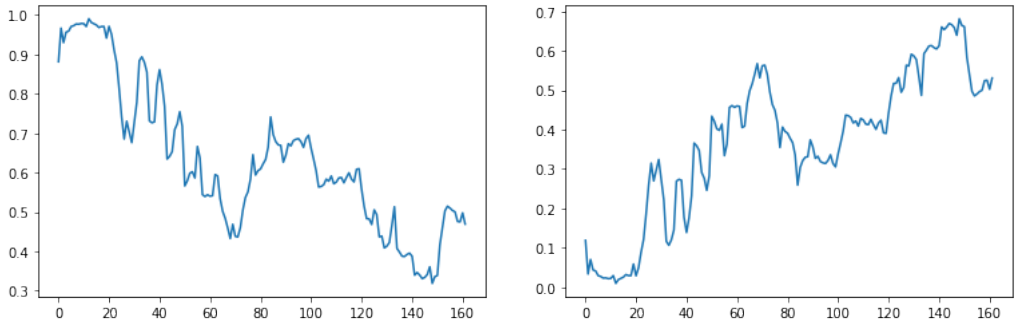


Figure 6.8: Percentage of administrations of first and second doses in USA from 17th December 2020. Data source: [97]

On December 14th, mass vaccination program started in Queens, USA, prior authorization on the 10th of December by the FDA (federal agency Food and Drug Administration) of the Pfizer/BioNTech vaccine (BNT162b1). On the 17th of December also the Moderna vaccine (mRNA-1283) has started being administered. Finally, the one dose vaccine produced by Johnson & Johnson (Ad26.COV2.S) has been authorized, even though its administrations were suspended on April 14th after evidence of correlations with blood clotting episodes. Starting from March 2021 some states of USA have opened to all adults the possibility to book their vaccination (*e.g.* Alaska, Mississippi, Ohio, Connecticut, Arizona, Texas and Georgia), due to the great amount of doses available in the country. Indeed, at the end of June almost 54% of Americans have received at least one vaccine doses, and 46% of them is completely immunized.

In Figure 6.8 it is represented the percentage of first and second doses on the total amount of administered doses per day in the US for 154 days starting from the beginning of the campaign. Notice that, neglecting local minima and fluctuations, after almost 20 days from the beginning of the campaign first doses follow a linear descent behaviour up to 50% of administered doses.

As in the UK case, a question can come to our minds:

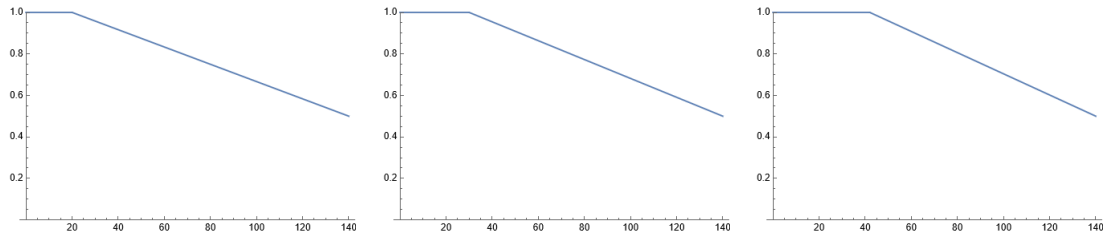
Is it possible to delay the beginning time of consistent administrations of second doses? Would it be a better solution in terms of deaths and infectious spread?

In order to answer these questions, we build the parametric approximations of percentages of first and second doses administrations:

$$f_{1,T} = \begin{cases} 1, & \text{for } t \in [0;T], \\ 1 - 0.5\frac{t-T}{140-T} & \text{for } t \in (T;T_f]; \end{cases} \tag{6.3}$$

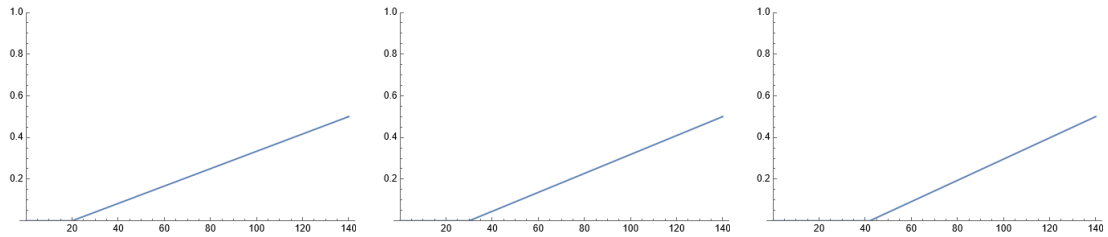
$$f_{2,T} = \begin{cases} 0, & \text{for } t \in [0;T], \\ 0.5\frac{t-T}{140-T} & \text{for } t \in (T;T_f]. \end{cases} \tag{6.4}$$

The parameter on which ripartitions of doses depend is $T \in [20, 42]$ ($T \in \mathbb{N}$), where T indicates the day at which second doses administration starts. In Figures 6.9 and 6.10 we provide three functions belonging to the family of first and second doses administrations with three specific values of the parameter T.



(a) $T = 20$ **(b)** $T = 30$ **(c)** $T = 42$

Figure 6.9: First doses percentage extrapolated from US policy as a constant function at 1 (100%) during the first 20 days that linearly decreases to 0.5 (50%). The descending behaviour starts at $T = 20$, 30 and 42 days from the beginning of the simulation respectively.



(a) $T = 20$ **(b)** $T = 30$ **(c)** $T = 42$

Figure 6.10: Second doses percentage extrapolated from US policy as a constant function at 0 (0%) during the first 20 days that linearly increases to 0.5 (50%). The ascending behaviour starts at $T = 20$, 30 and 42 days from the beginning of the simulation respectively.

The problem we aim to solve is the following optimization problem, where the control variable is the time T at which second doses administrations begin:

Optimal Control Problem.

$$\min_{T \in \mathbb{N}} J,$$

with state problem (4.4)*, where* $U_1(t) = p\, U_b\, f_{1,T}$*,* $U_2(t) = (1-p)\, U_b\, f_{1,T}$ *and* $U_3(t) = U_b\, f_{2,T}$*;* p *is the fractional ripartition of first doses among susceptibles and recovered individuals and* U_b *is the total daily amount of available vaccine jabs.*

The following simulations solve the direct problem associated with vaccinations prescribed by Functions (6.3) and (6.4) for 140 days. As for the UK-like case, the same four quantities have been evaluated for different T. Figure 6.11 represents their normalized values with respect to the frequency T on the x-axis.

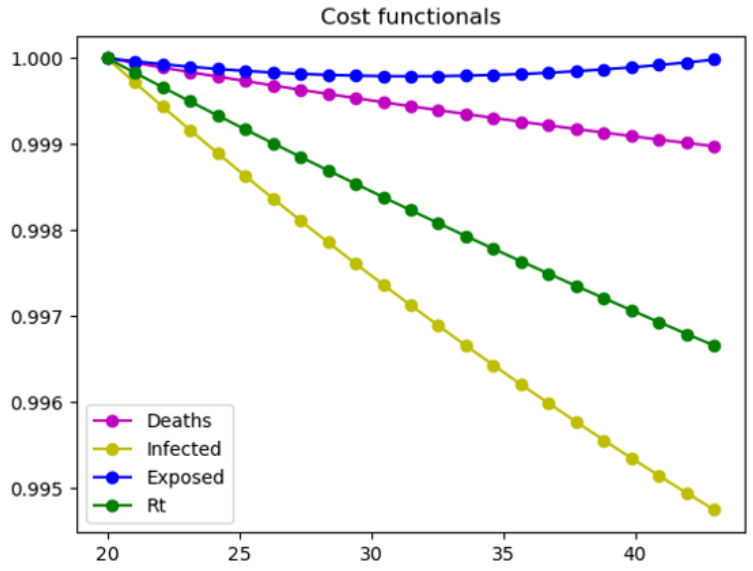


Figure 6.11: Normalized values of each performance measure for the US-like vaccination strategy. The typical period T is the independent coordinate x.

As one deduces by Figure 6.11, in order to minimize deceased, $\mathcal{R}_t$ and the whole infected curve, the optimal strategy is to delay as much as possible the beginning of second dose administrations. On the other hand, evaluation of the exposed cost function shows a parabolic behaviour, with minimum achieved at nearly 31 days. However, reduction in percentage of the four cases is at most of the order of 0.5% for the infectious case ($\simeq 7e4$ infected), and 0.1% in terms of deaths ($\simeq 2000$ deaths with respect to the highest case).

6.2.3 Conclusions

This section aimed at comparing scenarios coming from actual vaccination policies implemented in UK and USA in order to get information about possible improvements in the vaccination campaign.

The optimal strategy, suggested to policy makers who plan to propose UK-like or US-like vaccination campaigns, is to delay the beginning of administrations of second doses as much as possible. In the former scenarios, mantaining an high rate of people who received a partial coverage by the vaccination (belonging to V compartment) performs better in terms of deceased and infectious performance measures.

6.3 Optimal Control strategies varying NPI levels

This section is devoted to the analysis of optimal solutions which are obtained with different transmission rates (β). In particular three cases with different constant β are considered for the minimization of infectious. They have been extracted for `SEIHRDVW2` model by available data for Lombardy Region with three different levels of NPIs (*Non-Pharmaceutical Interventions*) implemented. These different levels correspond to political restrictions on the names of *Red area, Orange area* and *Yellow Area*.

6.3.1 Methods and parameters setting

We present some results inferred by solving OC Problem 1. The PGD method stops when the tolerance $1e-6$ is reached or a maximum number of iterations (100) has been successfully completed. The step of the same method is fixed at $1e-1$. We fix the total amount of daily administrations at 10000. Initial conditions for the state problem are set as in Table 6.1, and other parameters involved in the model are set as in Table 6.2.

We shortly explain the way we recovered plausible values for the transmission rates corresponding to different levels of NPIs. From EpiMox dashboard [88], which is based on data available by the Italian department of Protezione Civile [91], one obtains fitted values for regional $\mathcal{R}_t$ index (actually, we consider the $\mathcal{R}_t^*$ index extimated in the dashboard as explained in [9]). Then, β is computed inverting $\mathcal{R}_t$-definition in the following way:

$$\mathcal{R}_t = \frac{\beta}{\gamma}\frac{(S(t)+\sigma V(t))}{N} \Rightarrow \beta = \frac{\mathcal{R}_t^*\,\gamma\,N}{S(t)+\sigma V(t)}. \tag{6.5}$$

We average the values of $\mathcal{R}_t^*$, $S(t)$ and $V(t)$ assumed from 10 to 15 days after the entrance into force of restrictions indicated by the color, and then apply Equation (6.5).

The three transmission rates used for the different scenarios are (see Figure 6.12):

- $\beta = 0.19872$, *Red Area*;
- $\beta = 0.23225$, *Orange Area*;
- $\beta = 0.26172$, *Yellow Area*.

Simulations have been run for $T_f = 210$ days.

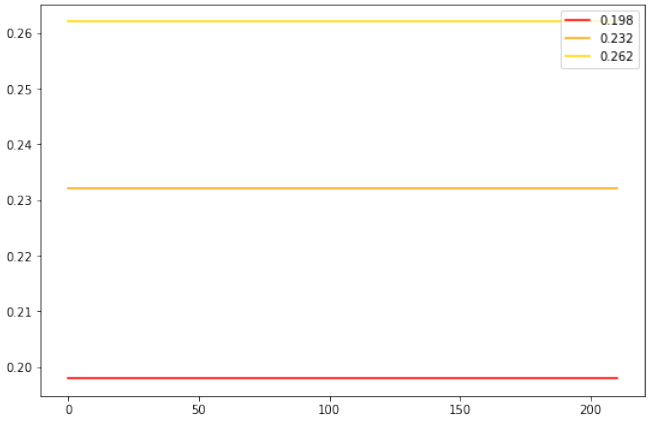


Figure 6.12: Different transmission rates corresponding to *Yellow, Orange* and *Red* regional restrictions.

6.3.2 Results

In this section, we assume the minimization of infectious individuals during the whole time-interval as cost functional (*i.e.* $\int_0^{T_f} I(t)^2 \, \mathrm{d}t$). We focus the attention on the daily distribution of first and second doses (see Figure 6.13) and on the absolute values reached by the cost functional in the three cases (see Table 6.3).

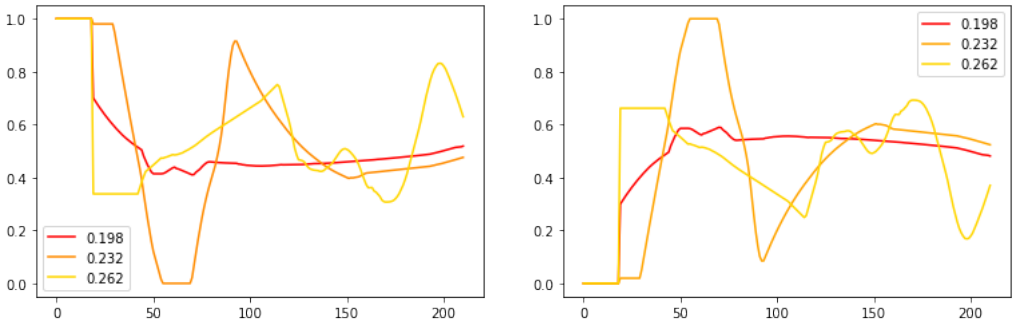


Figure 6.13: First (recovered and susceptibles) and second doses percentages compared at different levels of restrictions.

NPI level	β	Cost functional
1	0.26172	6.60e11
2	0.23225	1.62e11
3	0.19872	5.40e10

Table 6.3: Evaluation of the cost functional $\int_0^{T_f} I(t)^2 \, \mathrm{d}t$ at different levels of restrictions.

Percentage of first doses shows mild periodicity in the yellow case as we have previously observed in Section 6.1.2, that tends to disappear as β decreases. Indeed, as long as β decreases, oscillations decrease and the solutions tend to stabilize. Furthermore, in the long period (days 150-210) both first and second administrations tend to stabilize to 50% of total available doses for the orange and red curves. During the same period we notice the progressive accomodation of the orange curve ($\beta = 0.23225$) on the red curve ($\beta = 0.19872$). Table 6.3 assures the expected reduction in terms of cost functional when the transmission rate decreases.

Comparing these results with Section 6.1.2, we conclude that as β decreases optimal vaccination policy for minimizing infectious tends to take the shape of the solution of the same problem optimizing deceased at final time.

6.4 Impact of virus variants on optimal strategies

When pandemic events occur, if the recession period of the virus is not as short as expected, it is possible that multiple variants, also called *strains*, spread and in some cases become dominant and replace the existing strain. This is the case of the α variant which has replaced the first strain of the SARS-CoV-2 during October 2020, or the case of the δ and κ variants which are more transmissible and fatal than α one. Indeed ISS confirmed that on the 4th September 2021 in Italy the δ-variant accounts for 99.7% of infections. Variants are caused by mutations, *i.e.* changes in the DNA or RNA sequence of the microorganism. During RNA or DNA replication phases, errors can occur, and viruses can change rapidly their genetic sequence in order to have higher probabilities of surviving, developing different antigenic characteristics for evading human defensive response of the immune system [36].

The crucial question is,

How policy makers have to adapt vaccination campaign when variants spread?

In order to explore a possible mathematical answer to this question, we focus our attention on simplified situations dealing with:

1. variants which can be more transmissible (Section 6.4.2);
2. variants which can change vaccine effectiveness on preventing transmissibility (Section 6.4.3);
3. variants which alter vaccine effectiveness on reducing the mortality factor (Section 6.4.3).

6.4.1 Methods and parameters

Each of the following sections collects graphical outputs of solutions obtained via optimization of the OC Problem 2 with `SEIHRDVW2` as state problem. PGD parameters are

set as follows: tolerance $1e-6$, maximum number of iterations 100, step $1e-2$. The number of weekly delivered doses $N_{DosesDeliveredPerWeek}$ is fixed at 385000[2]. Instead, C, the maximum daily administration capacity, is fixed at 140000[3]. Initial conditions are the same proposed in Table 6.1. All parameters, except for σ and θ which are changed in the simulations related to vaccine effectiveness, are the same of Table 6.2. We simulate the optimal control problem for 147 days, *i.e.* 21 weeks.

6.4.2 Impact of transmissibility

We consider two different transmission rates, $\beta_1 = 0.26172$ representing the standard virus transmission rate and $\beta_2 = 0.42587$, obtained setting $\beta_2 = 1.6\beta_1$. In this way, the strain associated with β_2 is almost 60% more transmissible than the base strain β_1[4].

Case 1

We consider the following cost functional to optimize:

$$\int_0^{T_f} I(t)^2 \, dt.$$

Figures 6.14-6.18 represent evolution of all states, with special focus on the infected classes and total deaths for both transmission rates. From a quantitative point of view, in the case with β_2 it is natural to obtain higher numbers of exposed and consequently infected and healing individuals. Almost 40 days after the beginning of the simulation, exposed and infected peaks are reached with the first transmission rate, whereas for the variant case peaks occur on the 50th day of simulation. Maximum values of infected classes achieved in the second case are more than 8 times those caused by the standard strain. With a variant that is 1.6 times more transmissible, maximum number of coexisting infectious is about $4e5$.

[2]385000 is a compatible value with the average number of doses delivered per week in Lombardy up to June 2021.

[3]This value is the maximum number of doses administered in Lombardy during the same period.

[4]We choose 60% according to [21], where it is claimed that δ-variant is 60% more transmissible than the α one.

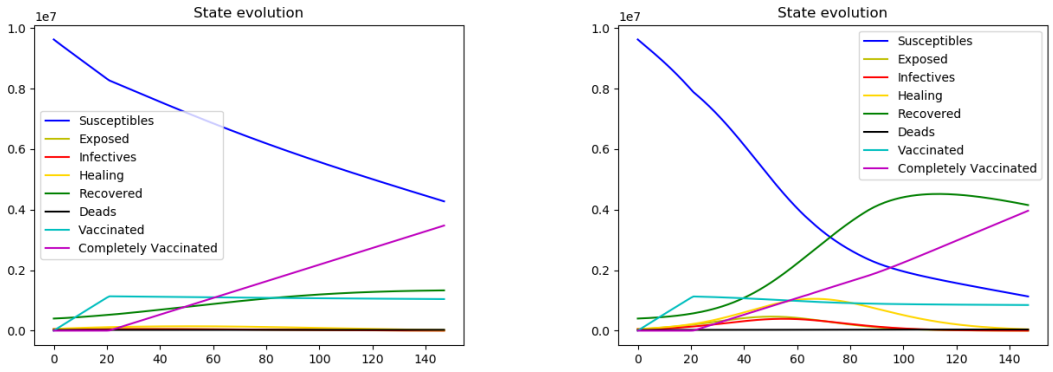


Figure 6.14: States evolution minimizing $J = \int_0^{T_f} I(t)^2\,dt$ for β_1 strain (left figure) and β_2 strain (right figure).

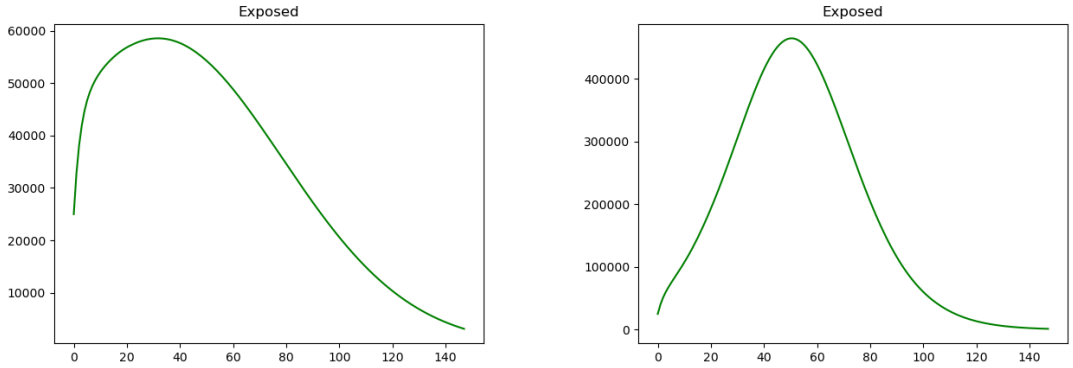


Figure 6.15: Exposed evolution minimizing $J = \int_0^{T_f} I(t)^2\,dt$ for β_1 strain (left figure) and β_2 strain (right figure).

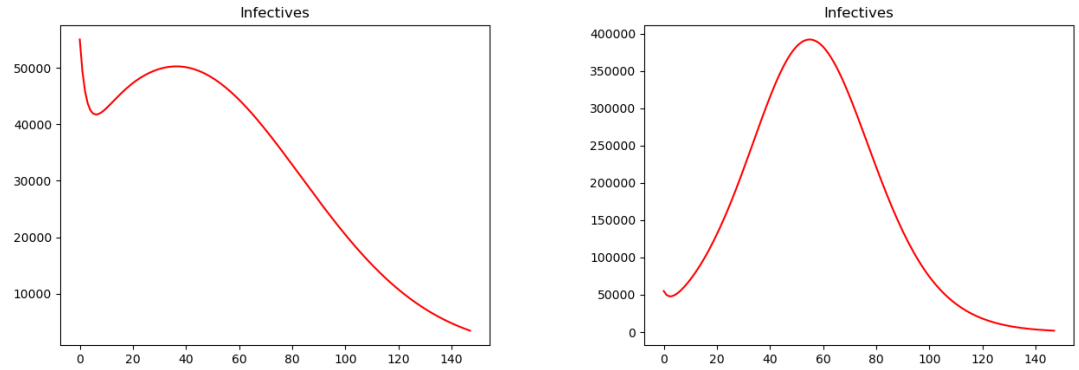


Figure 6.16: Infectious evolution minimizing $J = \int_0^{T_f} I(t)^2\,dt$ for β_1 strain (left figure) and β_2 strain (right figure).

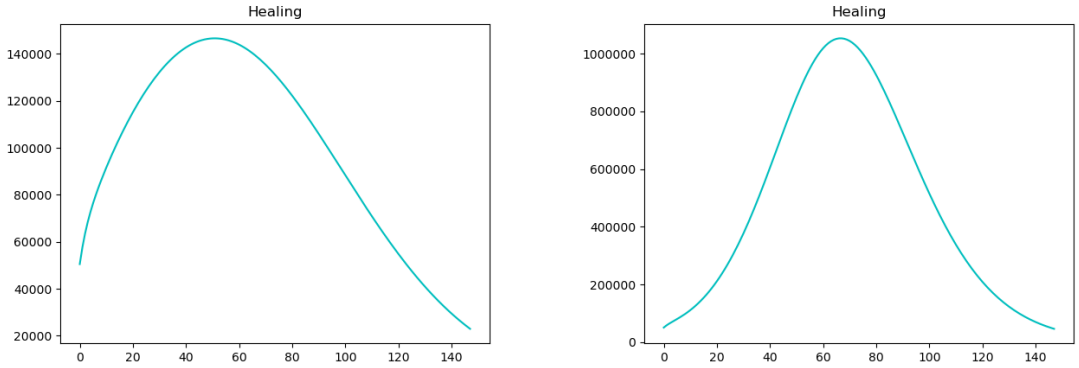


Figure 6.17: Healing evolution minimizing $J = \int_0^{T_f} I(t)^2\,dt$ for β_1 strain (left figure) and β_2 strain (right figure).

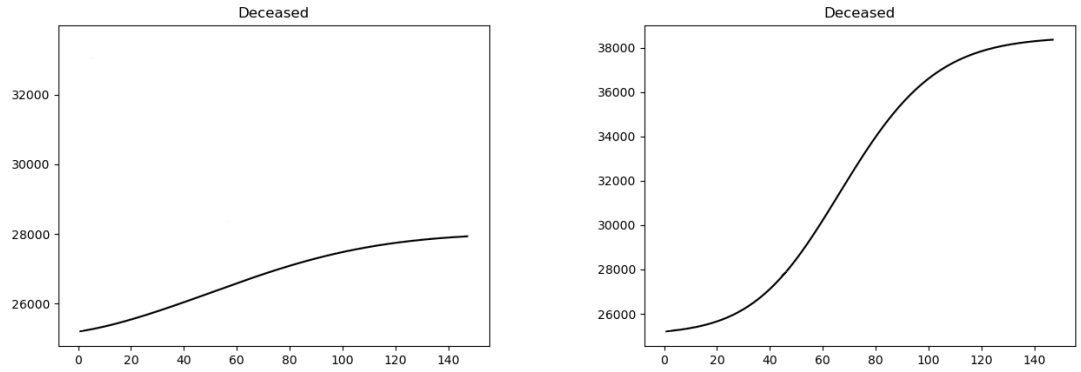


Figure 6.18: Deceased evolution minimizing $J = \int_0^{T_f} I(t)^2\,dt$ for β_1 strain (left figure) and β_2 strain (right figure).

Figures 6.19 and 6.20 represent the two distinct optimal vaccination policies. We first note that the optimal solutions do not include days without administrations, but rather, both solutions tend to stabilize the daily total administrations value at almost 55000, which is the daily average value of 385000 delivered doses per week, without collecting stocks. In the left part of the figures we observe solutions related to β_1, and we conclude that vaccination priority is given to susceptibles. First and second doses to susceptibles are administered equally during the whole timeframe except for the first 20 days of simulation. Recovered individuals are neglected during the timeframe of interest, probably since they are advantaged with respect to complete immunity.

We observe a different optimal vaccination policy with the variant strain in the right side of the same figures. Indeed, nearly after 100 simulation-days the three controls are approximately administered equally, meaning 33% of total administration are made for each control. Beginning administrations to recovered causes susceptibles curve to slow down its course towards zero.

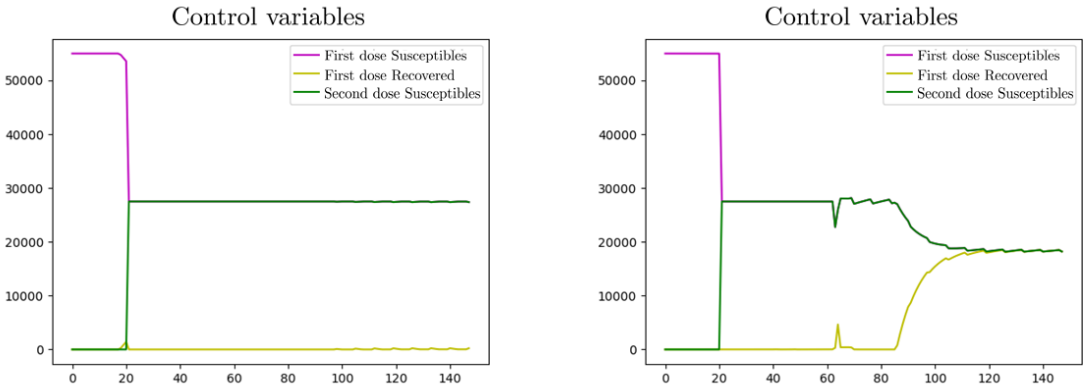


Figure 6.19: Control variables minimizing $J = \int_0^{T_f} I(t)^2\,dt$ for β_1 strain (left figure) and β_2 strain (right figure).

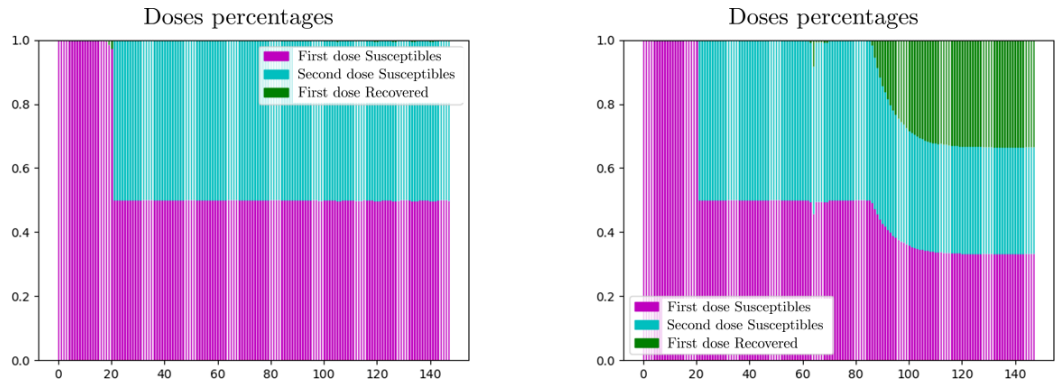


Figure 6.20: Comparison of histograms representing daily percentages of doses minimizing $J = \int_0^{T_f} I(t)^2\, dt$ for β_1 strain (left figure) and β_2 strain (right figure).

Figure 6.21 represents $\mathcal{R}_t$ indexes computed for the two scenarios. As one expects, $\mathcal{R}_t$ index is higher in the right case (≈ 2 at the beginning) with respect to the left, but near T_f it is lower than in the left case (≈ 0.30). This is obvious noting that $\mathcal{R}_t$ is proportional to $V(t)$ and $S(t)$, and that close to T_f this second term - which is predominant - is lower in the second case than in the first since more people have been infected.

We conclude that more transmissible variants impact the vaccination strategy advancing administrations to recovered, in order to create a relevant barrier of immune individuals against the virus.

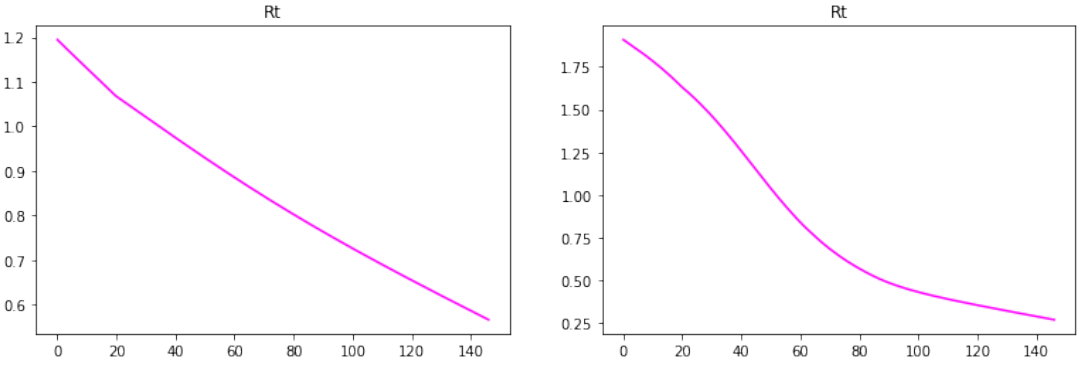


Figure 6.21: $\mathcal{R}_t$ comparison between optimal solutions for both β_1 (left figure) and β_2 (right figure) virus strains. Cost functional: $\int_0^{T_f} I(t)^2\ \mathrm{d}t$.

Case 2

We perform simulations of the optimal control problem with a complete cost functional, *i.e.*

$$\int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \, dt + C_2 D(T_f)^2,$$

with $C_1 = 1e4$ and $C_2 = 1e3$. Figures 6.22-6.26 are the graphical evolutions of all states involved in the problem, with a special zoom on the infectious classes and on deceased. Deceased level seems to be higher for the β_2 case (≈ 40000 deaths) with respect to Case 1. Indeed, convergence minimum is not reached, implying that $D(T_f)$ component is not completely minimized.

At time $T_f = 147$, susceptibles in the β_2 case are lower than in β_1 simulation, even though the final value is higher than the respective value for Case 1 optimization. Another visible difference in the evolution of states is represented by the recovered variable, which vanishes at nearly 90 days from the beginning in the standard strain case, and it is lower than in Case 1 simulation.

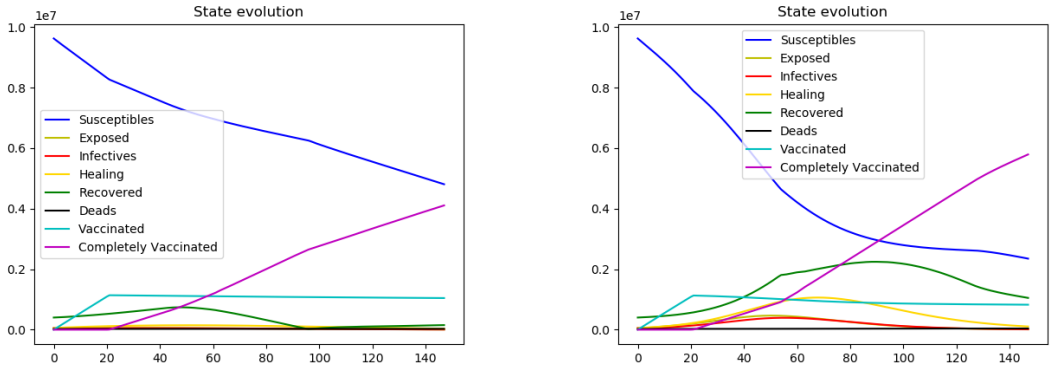


Figure 6.22: States evolution minimizing $J = \int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \, dt + C_2 D(T_f)^2$ for β_1 strain (left figure) and β_2 strain (right figure).

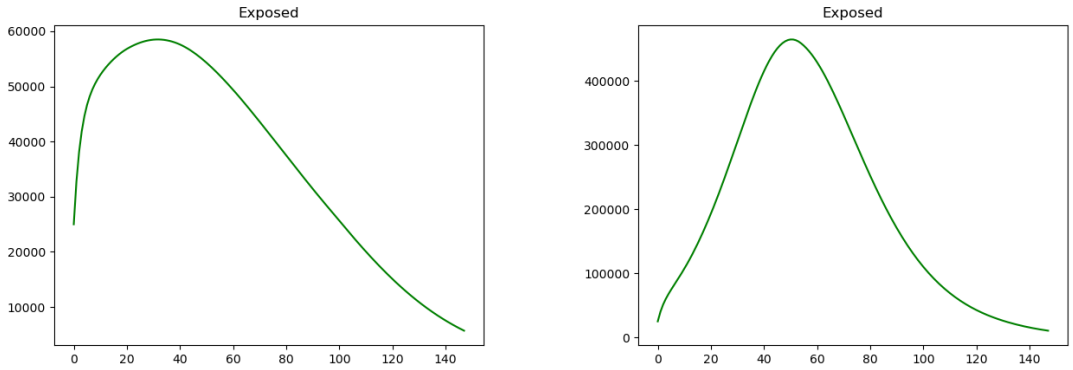


Figure 6.23: Exposed evolution minimizing $J = \int_0^{T_f} I(t)^2 + C_1\dot{E}(t)^2\,dt + C_2 D(T_f)^2$ for β_1 strain (left figure) and β_2 strain (right figure).

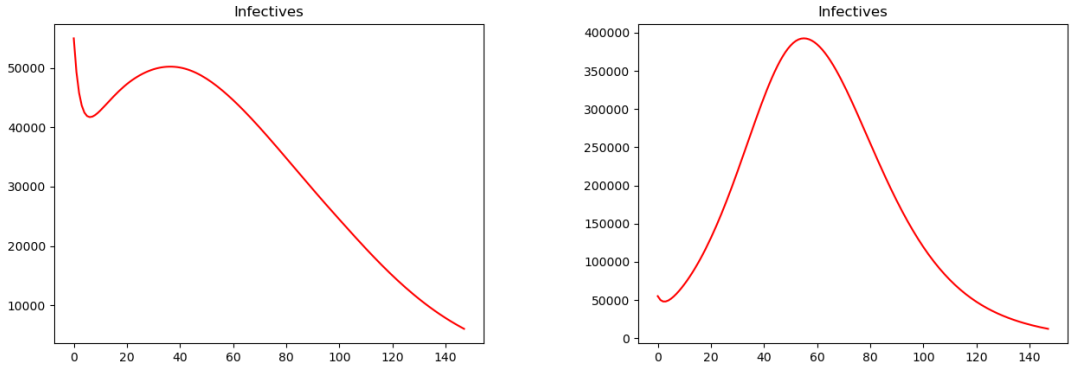


Figure 6.24: Infectious evolution minimizing $J = \int_0^{T_f} I(t)^2 + C_1\dot{E}(t)^2\,dt + C_2 D(T_f)^2$ for β_1 strain (left figure) and β_2 strain (right figure).

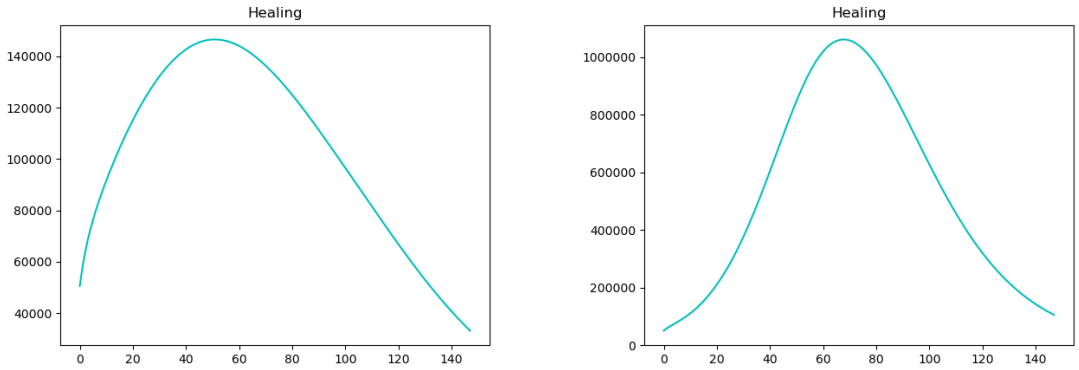


Figure 6.25: Healing evolution minimizing $J = \int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \, dt + C_2 D(T_f)^2$ for β_1 strain (left figure) and β_2 strain (right figure).

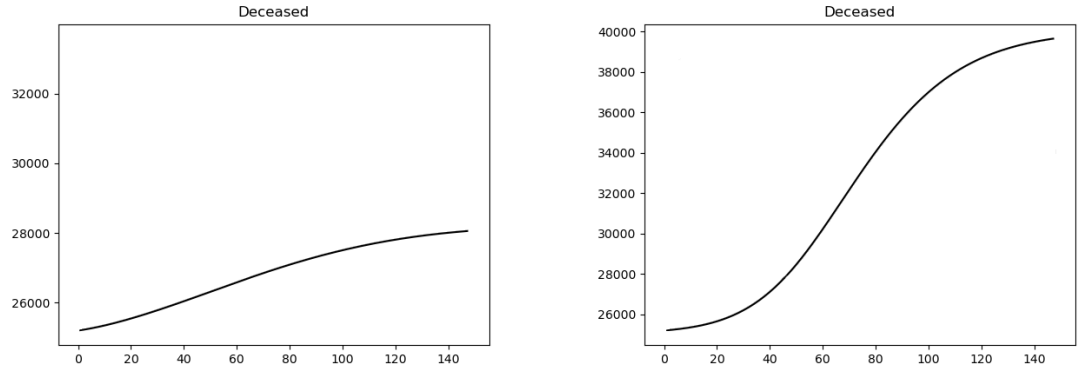


Figure 6.26: Deceased evolution minimizing $J = \int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \, dt + C_2 D(T_f)^2$ for β_1 strain (left figure) and β_2 strain (right figure).

Figures 6.27 and 6.28 refer to vaccination summary and doses administered percentages in this scenario. For the left case we notice that, after 20 days of exclusive vaccinations of susceptibles, and their subsequent immunisation, recovered administrations start taking place until all recovered have been immunised. Contemporarily first and second doses to susceptibles are administered in equal percentages. Administrations per day are quite constant up to $5.5e4$, and there are not days without administrations.

The right case is different from the left one. Indeed, after the first 50 days, when only first and second doses to susceptibles are provided equally, the solution starts vaccinations for the recovered class exclusively until the 125th day when the three controls start to equilibrate to 33% ripartition each. The day when administrations to recovered start coincides with the day when infected peak occurs.

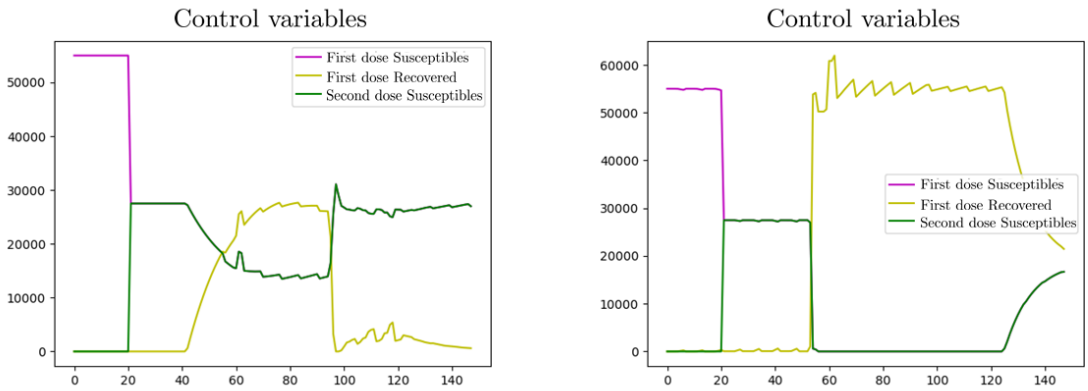


Figure 6.27: Control variables minimizing $J = \int_0^{T_f} I(t)^2 + C_1\dot{E}(t)^2\,dt + C_2 D(T_f)^2$ for β_1 strain (left figure) and β_2 strain (right figure).

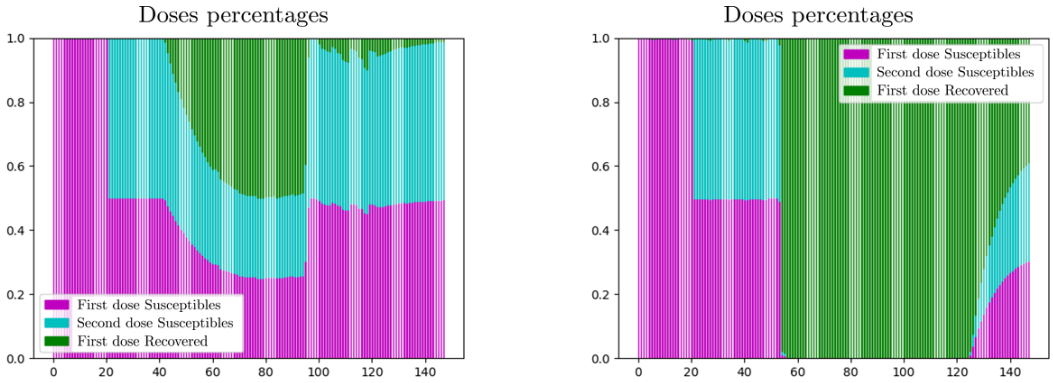


Figure 6.28: Comparison of histograms representing daily percentages of doses minimizing $J = \int_0^{T_f} I(t)^2 + C_1\dot{E}(t)^2\,dt + C_2 D(T_f)^2$ for β_1 strain (left figure) and β_2 strain (right figure).

Figure 6.29 represents the evolution of the reproducing number for each of the two simulations dealt in this case. $\mathcal{R}_t$ tends to stabilize to 0.55 for the right case, while its left counterpart approaches 0.65 approximately.

We conclude that a fundamental role is played again by the amount of administrations to recovered individuals (which is the U_2 control variable). Its value rises in worse scenarios, in order to achieve in advance a sort of herd immunity. Note that this is possible beacause the virus is more transmissible, and therefore more people have been infected and then recovered.

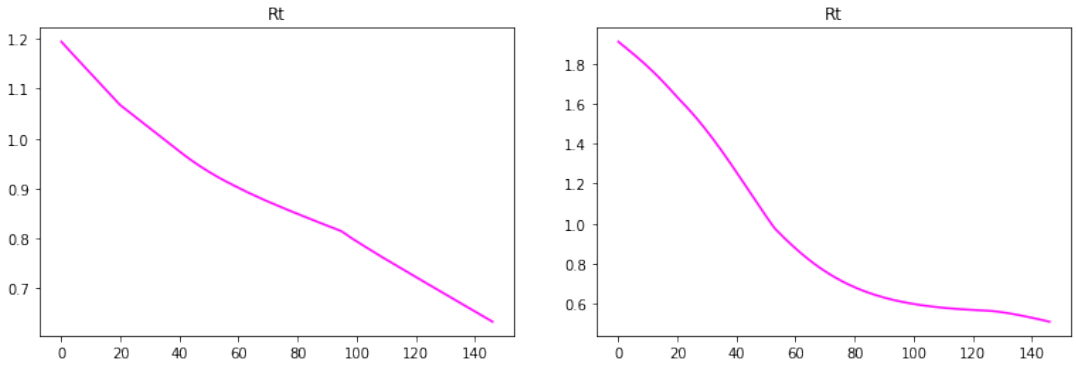


Figure 6.29: $\mathcal{R}_t$ comparison between optimal solutions for both β_1 (left figure) and β_2 (right figure) virus strains. Cost functional: $\int_0^{T_f} I(t)^2 + C_1\dot{E}(t)^2\,dt + C_2 D(T_f)^2$.

6.4.3 Vaccine effectiveness

Vaccines that have been authorized by the EMA (*European Medicine Agency*) and AIFA (*Agenzia Italiana del Farmaco*) have a double impact on the epidemic. Indeed, they can help in reducing transmissibility of the virus and/or they can abate severe disease which often leads to death. In the proposed mathematical model, these two physical circumstances are both englobed through the definition of two parameters, respectively σ and θ (see Models (4.1) and (4.4)).

The question is:

How does the vaccination campaign have to be modified in order to reduce already-defined performance measures when variants acting on vaccine effectiveness occur?

Indeed, we interpret the following results in order to have guidelines for policy makers for planning an optimal vaccination campaign when virus is modified by mutations and can interact with vaccines reducing their effectiveness. Notice that we assume that vaccine effectiveness is constant over time, while it is proved that vaccines do not provide individuals with an istantaneous coverage, and their effectiveness increases when time grows. Moreover, we consider an ideal vaccine which guarantees complete immunity when the cycle is completed[5].

Transmission rate β is fixed during the whole simulation time at 0.26172. We deal with OC Problem 2, with $N_{DosesDeliveredPerWeek} = 385000$ and $C = 140000$, maximum daily administration capability.

[5]This is not actually true for vaccines against SARS-CoV-2, which do not assure immunity after the cycle is completed (*i.e.* $\sigma_W = 0$ and $\theta_W = 0$, considering the model introduced in Section 7.2).

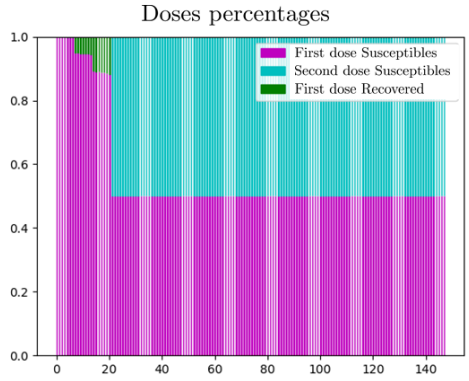
Variants affecting vaccine effectiveness on transmissibility (σ)

We set vaccine effectiveness on mortality (θ) at 0.15. Three different values of σ are considered, $\sigma = 0.40, 0.25, 0.10$, meaning that vaccine effectivenesses in preventing the spread after one single administration are 60%, 75% and 90% respectively.

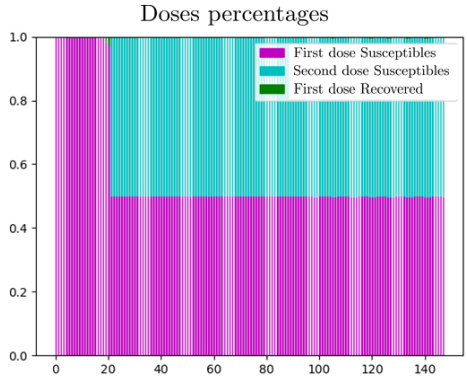
Figure 6.30 represents percentage ripartitions of the three controls optimizing the cost functional

$$\int_0^{T_f} I(t)^2 \, \mathrm{d}t.$$

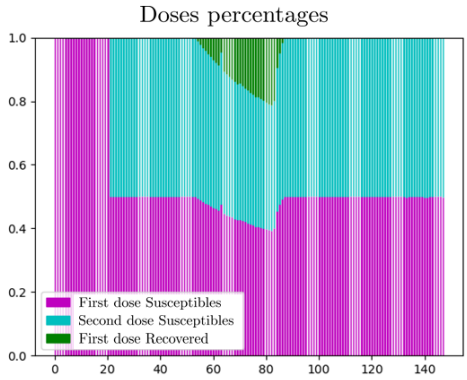
From these results, we note changes on doses administered to recovered individuals.



(a) $\sigma = 0.40$



(b) $\sigma = 0.25$



(c) $\sigma = 0.10$

Figure 6.30: Comparison of histograms representing percentages of administrations to susceptibles and recovered when $\sigma = 0.40, 0.25, 0.10$ respectively.
Cost functional: $\int_0^{T_f} I(t)^2 \, dt$.

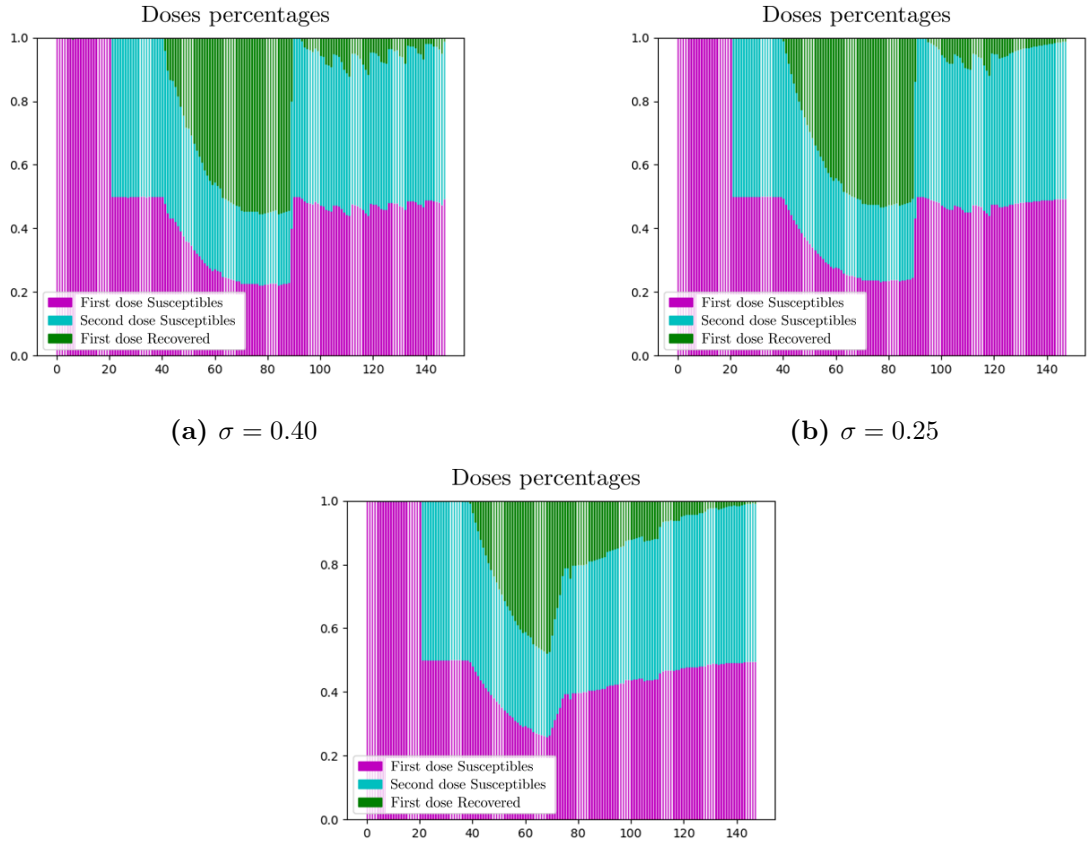
Indeed, during the first 20 days of simulation, administrations to recovered disappear when vaccine effectiveness increases. Instead, in the solution related to 90% of vaccine effectiveness, recovered administrations start 60 days after the beginning. In all three

cases, first and second doses administrations to susceptibles stabilize at 50% each in the long term.

The second cost functional that we consider is the one which takes under control not only the infectious curve but also variations in the exposed curve, *i.e.*

$$\int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \, \mathrm{d}t.$$

Figure 6.31 represents doses ripartitions with the three different levels of vaccine effectiveness.



(a) $\sigma = 0.40$ **(b)** $\sigma = 0.25$

(c) $\sigma = 0.10$

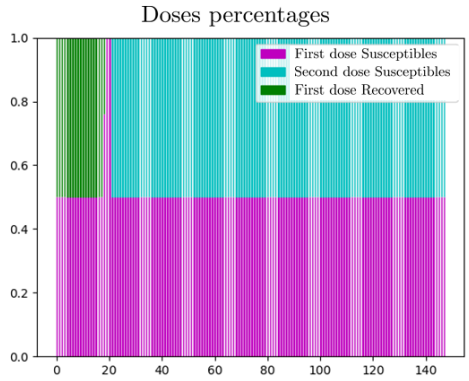
Figure 6.31: Comparison of histograms representing percentages of administrations to susceptibles and recovered when $\sigma = 0.40, 0.25, 0.10$ respectively.
Cost functional: $\int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \, dt$.

In this case, we recognize a trend in the recovered doses administrations when vaccine effectiveness increases and this is merely due to the exposed component. Indeed, the higher vaccine effectiveness the lower is the level of recovered administrations, control

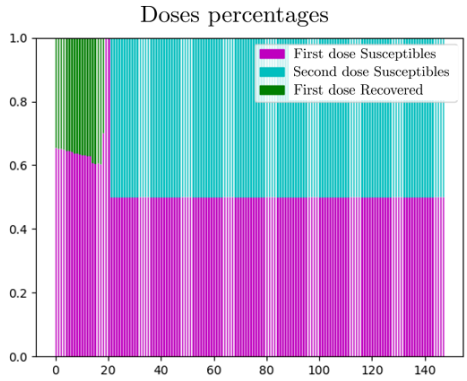
variable U_3, during the timeframe 60-80 days from the beginning of the simulation time. After this transient period all solutions tend to equilibrate to an half-and-half ripartition of administrations of first and second doses to susceptibles.

The last scenario we consider is the one related to optimization of the cost functional taking into account for total deaths during the process, namely

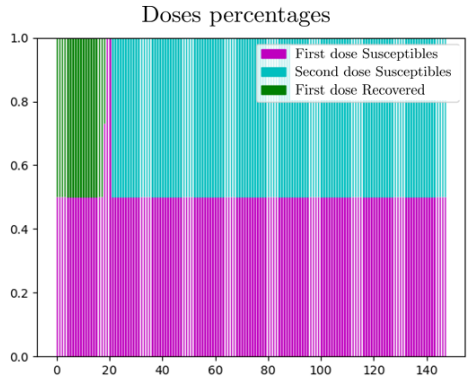
$$D(T_f)^2.$$



(a) $\sigma = 0.40$



(b) $\sigma = 0.25$



(c) $\sigma = 0.10$

Figure 6.32: Comparison of histograms representing percentages of administrations to susceptibles and recovered when $\sigma = 0.40, 0.25, 0.10$ respectively.
Cost functional: $D(T_f)^2$.

From Figure 6.32 it emerges that in the period 20-147 days first and second doses administrations to susceptibles assest to an half-and-half ripartition of daily available doses independently on σ. Even the control variable related to administrations to recovered individuals is almost independent by variations on the parameter σ, and this control only acts during the first 20 days of administration, where no second doses are

allowed to be administered.

Analyzing the three scenarios with different cost functionals we note that the component related to variations of slopes of the exposed curve is the most influenced by variations of vaccine effectiveness in reducing transmissibility. As σ decreases vaccination policy changes increasing administrations to susceptibles and reducing administrations to recovered individuals.

Variants affecting vaccine effectiveness on mortality (θ)

The following results are collected setting all parameters as in Table 6.2 apart from θ. This parameter, which is related to vaccine effectiveness in preventing severe effects which can lead to death, is varied in three different situations, namely $\theta = 0.40, 0.30, 0.15$ meaning that vaccines reduce the possibility of fatal effects of 60%, 70% and 85% respectively.

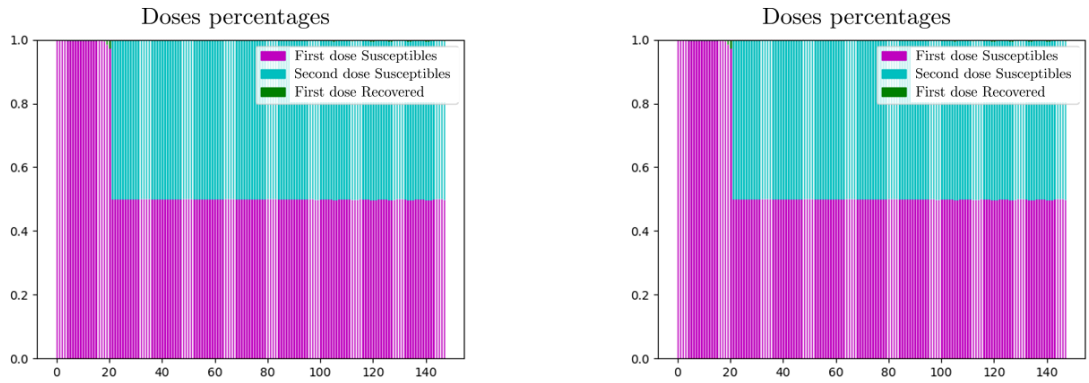
Figure 6.33 represents percentage allocation of control varibales for solutions of the optimal control problem when the cost functional is

$$\int_0^{T_f} I(t)^2 \, \mathrm{d}t.$$

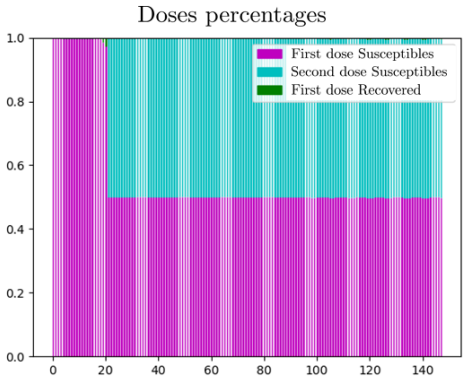
We deduce that in this case the solution is not dependant on variations of the parameter θ. This is not surprising since the parameter θ enters in the model in the fatality function

$$f(S,V) = \bar{f}\frac{S(t-15) + \sigma\theta V(t-15)}{S(t-15) + \sigma V(t-15)},$$

which regulates fluxes from the healing class to deceased and recovered. These classes do not have direct influence on I class. From the recovered class it is possible to be re-infected, even though the μ_R parameter - which represents the possibility that antibodies run out after a certain time from infection - is very small (Table 6.1).



(a) $\theta = 0.40$ **(b)** $\theta = 0.30$



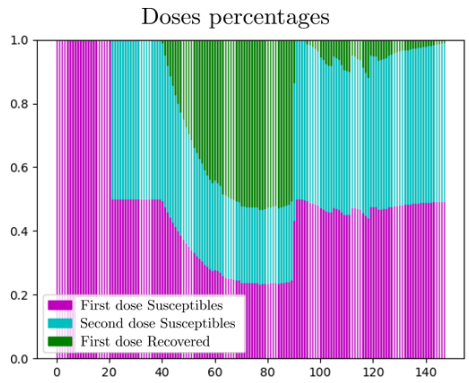
(c) $\sigma = 0.15$

Figure 6.33: Comparison of histograms representing percentages of administrations to susceptibles and recovered when $\theta = 0.40, 0.30, 0.15$ respectively.
Cost functional: $\int_0^{T_f} I(t)^2\, dt$

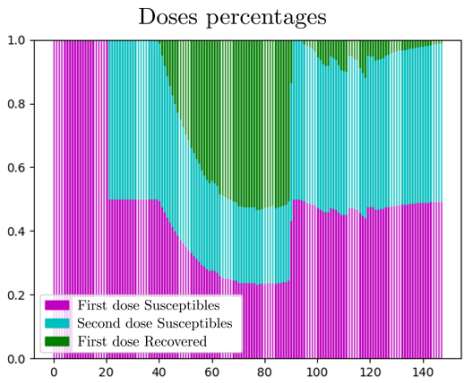
From Figure 6.34 we draw the same conclusions herebefore. The solutions represented in the picture are obtained via optimization of the following cost functional

$$\int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \ \mathrm{d}t,$$

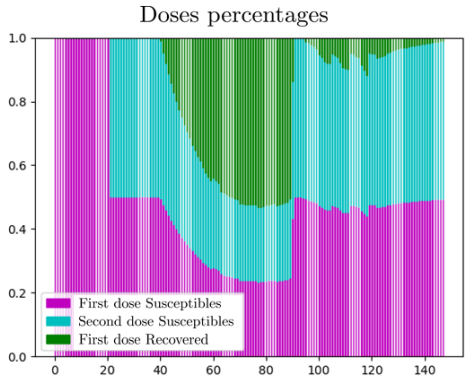
where C_1 is $1e4$. Although θ has been varied as explained before, we do not observe variation in the doses ripartitions among the three cases. Indeed, neither I either E classes are directly influenced by parameter θ itself.



(a) $\theta = 0.40$



(b) $\theta = 0.30$



(c) $\sigma = 0.15$

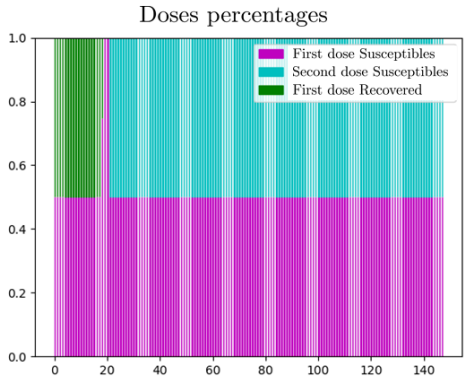
Figure 6.34: Comparison of histograms representing percentages of administrations to susceptibles and recovered when $\theta = 0.40, 0.30, 0.15$ respectively.
Cost functional: $\int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \, dt$.

Finally, we are left to the case where the cost functional to optimize is directly related to total deaths at the final simulation time, namely

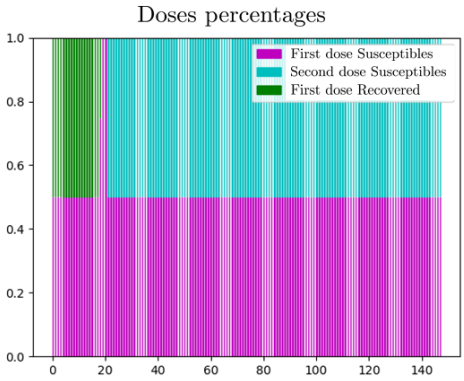
$$D(T_f)^2.$$

For this case, one expects variations in the solutions for the three cases, since, as it was previously noted, θ is directly involved in the fatality function. Actually, the mild change that is observed among the three cases is represented by the variable accounting for administrations to recovered, which is reduced from 50% to 30-35% during the first 20 days of simulation with $\theta = 0.15$.

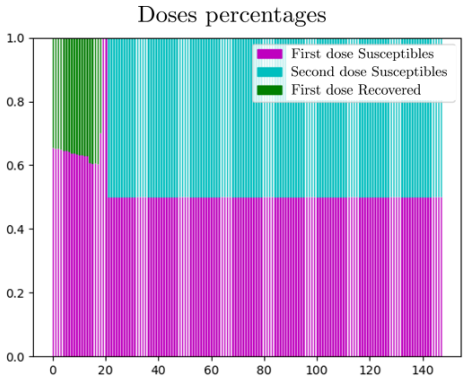
We conclude that the optimal strategy seems to be independent on the value of vaccine effectiveness on mortality for each of the considered cost functions.



(a) $\theta = 0.40$



(b) $\theta = 0.30$



(c) $\sigma = 0.15$

Figure 6.35: Comparison of histograms representing percentages of administrations to susceptibles and recovered when $\theta = 0.40, 0.30, 0.15$ respectively.
Cost functional: $D(T_f)^2$.

Chapter 7

Semi-Realistic scenarios

This chapter includes results coming from quasi-realistic situations, deducing some parameters from the actual scenario of Lombardy in 2021.

The chapter is structured is constituted by two sections: Section 7.1 deals with optimal vaccination policies setting the problem in Lombardy, and using real data for the transmission rate and daily vaccinations. In Section 7.2 we propose a modified version of the `SEIHRDVW2` model that integrates the possibility of incomplete immunity after completing the vaccination cycle. The modified model is employed to compare results obtained assuming BioNTech/Pfizer and AstraZeneca vaccines.

7.1 Optimal vaccination policy with realistic amount of doses and transmission rate

In the previous chapter we mantained at a constant average value the transmission rate (β) as well as the total number of daily administrations (fixed at 10000, which is lower than the actual mean of daily administrations in Lombardy in January-May 2021, amounting at almost 60000). In Figure 7.1 we report the actual values of total administrations per day and the daily values of the transmission rate in a timeframe of length 147 days from 01-01-2021 in Lombardy. We extracted these functions starting from available data from the Italian department of Protezione Civile ([92],[91]). For the transmission rate we applied the same procedure introduced in Section 6.3.

Next results collect and discuss the solutions of different Optimal Control problems with different cost functionals setting total daily administrations and daily transmission rate as in Figure 7.1. In this way we aim at contributing to underline possible improvements in the vaccination campaign implemented in Lombardy. Notice that this comparison is not completely fair since Lombardy's policy makers have designed vaccination campaign basing the vaccination order in the way that fragile categories and age-advanced categories should be immunized with absolute priority. Instead, our model concerns average compartments, and therefore we do not take into account any social status or age ripartition of individuals. However, our purpose is to extract from

these results some instructions about the optimal average ripartition of first and second doses administrations.

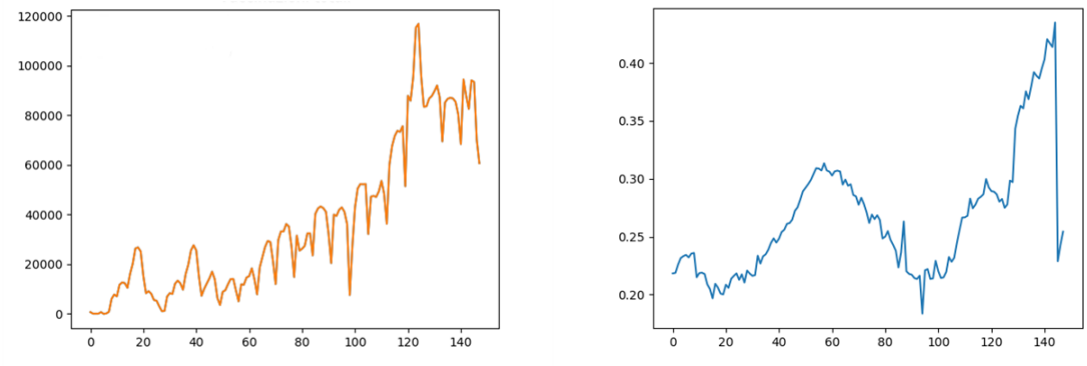


Figure 7.1: Daily vaccine available doses (left figure) and transmission rate (right figure) in Lombardy in the timeframe of 147 days starting on the 1st January 2021.

We highlight that, surely, implementing optimal vaccination campaign leads to remarkable differences in terms of saved deceased with respect to the case without vaccinations, as seen from Figure 7.2.

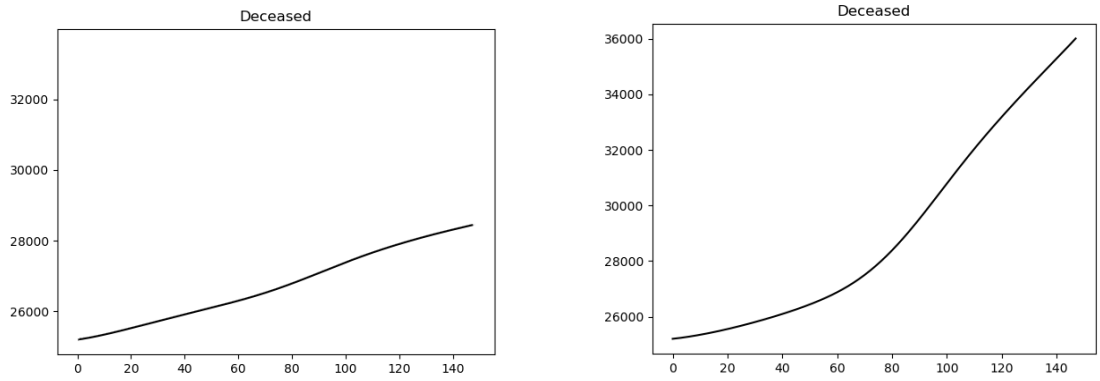


Figure 7.2: Simulated evolution of deceased individuals with optimal vaccination strategy (left) and without employing vaccines (right).

The following results cannot be straightforwardly compared with the real evolution of deceased and infectious for the case of Lombardy in the same period, mainly due to the low parameter accuracy. On the other hand, we obtain solutions where infectives and deceased have comparable orders of magnitude with real data.

7.1.1 Methods and parameters

All parameters associated with `SEIHRDVW2` model have been set following Tables 6.1 and 6.2. Parameters of the PGD method are set as in Chapter 6, *i.e* tolerance at $1e-6$, step at $1e-1$ and maximum iterations at 100. We deal with OC Problem 1 formulation, apart from Case 4 where we compare solutions for both OC Problems 1 and 2. The main differences from previous simulations are represented by the number of administrations per day and the transmission rate, set as in Figure 7.1.

7.1.2 Results

Case 1

We present the solution of the Optimal Control Problem 1 choosing the following cost functional:

$$\int_0^{T_f} I(t)^2 \, \mathrm{d}t.$$

The algorithm stops after the maximum number of iterations is reached and returns $2.46e11$ as the value of the functional. The amount of infected individuals at the end of the optimization process is about 40890.

In Figure 7.3 we observe the evolution of all variables involved, *i.e* S, E, I, H, R, D, V and W. Figure 7.4 shows the percentage ripartitions of first, second doses and doses administered to recovered. A suitable interpolation obtained through the *Least Square* method with polynomials of order 6 is shown in the same figures. Figure 7.5 compares first and second doses actually administered in Lombardy with those obtained with the optimal solution. Figures 7.6 and 7.7 represent the computed $\mathcal{R}_t$ index and deceased evolution for the optimal strategy versus the solution of the direct problem of `SEIHRDVW2` imposing the implemented vaccination campaign in the same period in Lombardy and β as in Figure 7.1. From this point onward, we refer to this last strategy as *Direct Problem in Lombardy*, acronym *DPL*. For the DPL we set the proportion of first doses adminstered to suceptibles and recovered at $\frac{3}{4}$ and $\frac{1}{4}$ respectively.

Case 1: Comments Maximum value of exposed individuals is greater than 60000 and it is reached at nearly 80 days from the beginning of the vaccination campaign. The highest value of infectious that is reached during the 147 days is 55000. Deceased variable is, as expected, a monotone increasing curve, which reaches its maximum of 28000 deaths. The increment of new deaths during the simulation timeframe is 3000. Solving the DPL nearly 30000 deaths are reached in the period of interest. Hence, optimal solution makes a reduction of 7% in terms of deaths with respect to the DPL.

The most relevant difference in the administration of doses with the implemented vaccination campaign is the delay in the start of the administrations of second doses

and this fact is completely in agreement with results coming from Section 6.2. Actually, this period corresponds to the complete immunisation of healthcare workers. Administrations of second doses tends to settle at a constant value of more than 15000 doses per day. We note that as well as β increases (days 20-60), second doses administrations increase. In time intervals where β decreases (days 60-100), the percentage of first doses administrations increases.

For what concerns the reproduction number $\mathcal{R}_t$, we notice a similar trend in terms of growth and decaying phases for both the optimal strategy and the DPL. However, as expected the optimal policy improves the index itself especially in the last days of the simulation, where the index reaches 1.1 peak value for the optimal policy, 1.6 for the solution of the DPL.

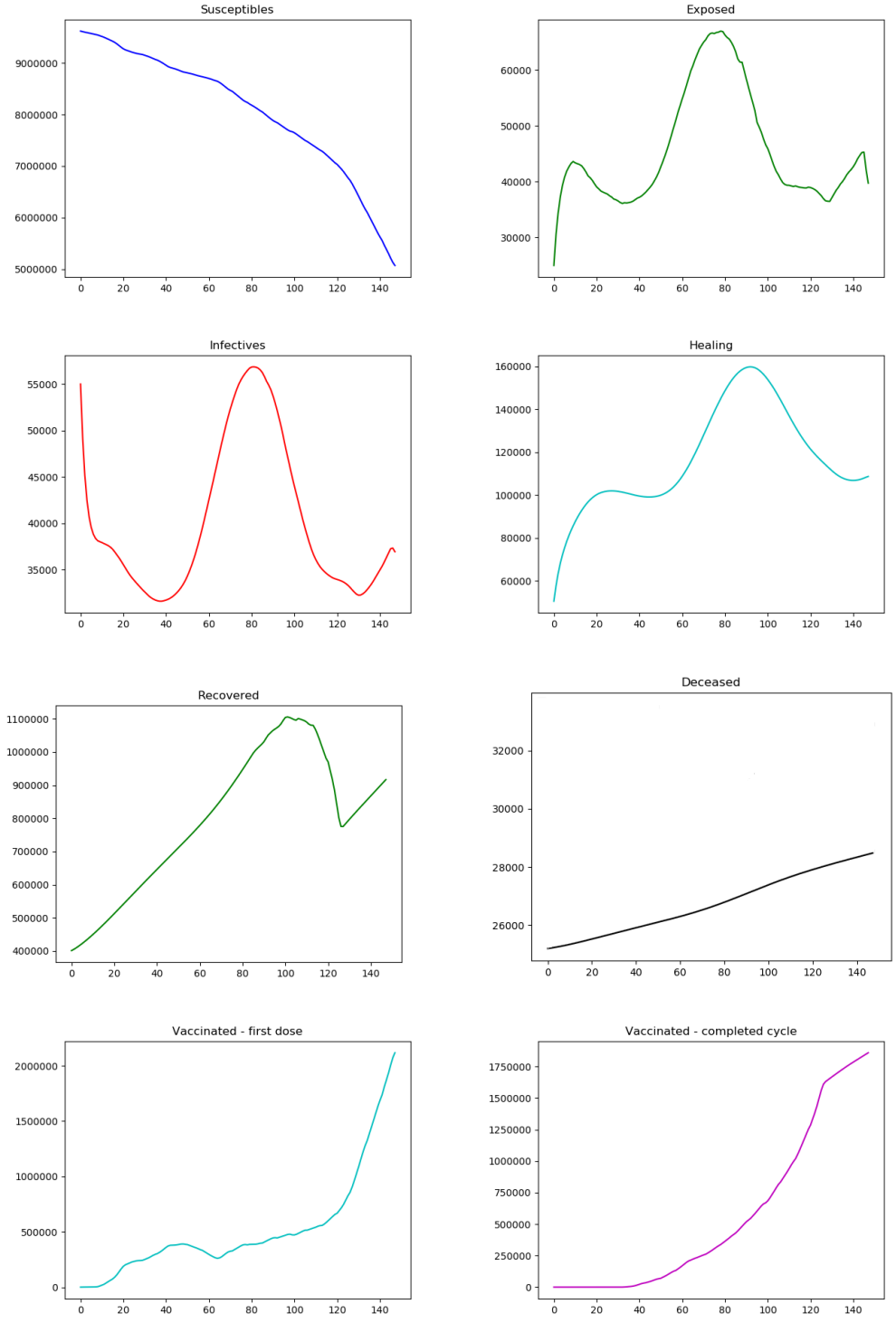


Figure 7.3: Evolution of each state of the `SEIHRDVW2` model. Case: $\int_0^{T_f} I(t)^2 \, \mathrm{d}t$.

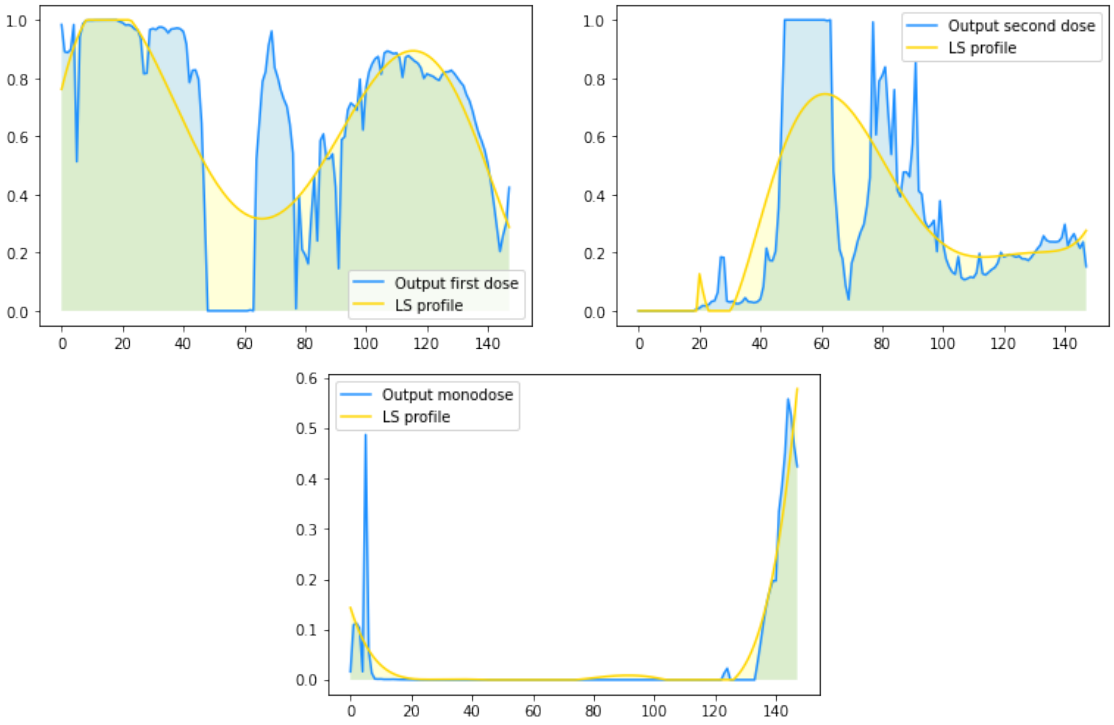


Figure 7.4: Percentages of first, second doses and doses administered to recovered individuals, overlapped with *Least Square* interpolation of order 6.
Case: $\int_0^{T_f} I(t)^2 \, \mathrm{d}t$.

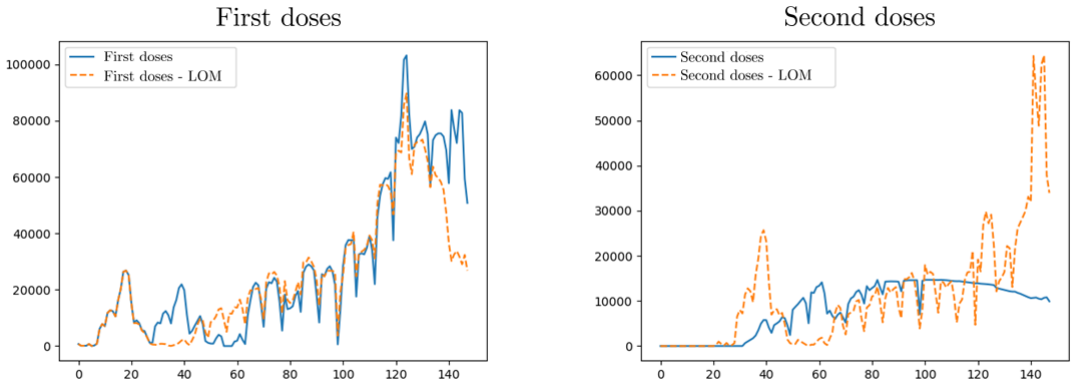


Figure 7.5: First and second doses comparison between actual ripartitions in Lombardy and optimal solution. In the first dose we also consider doses administered to recovered. Case: $\int_0^{T_f} I(t)^2 \, \mathrm{d}t$.

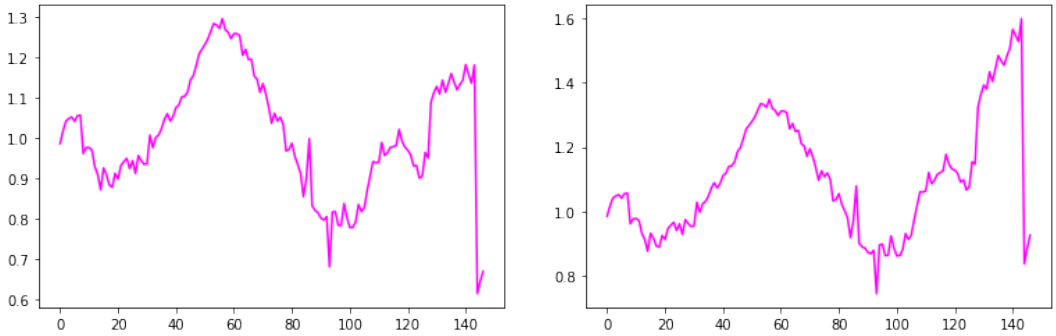


Figure 7.6: $\mathcal{R}_t$ comparison between the optimal solution (left figure) and DPL (right figure). Case: $\int_0^{T_f} I(t)^2 \, \mathrm{d}t$.

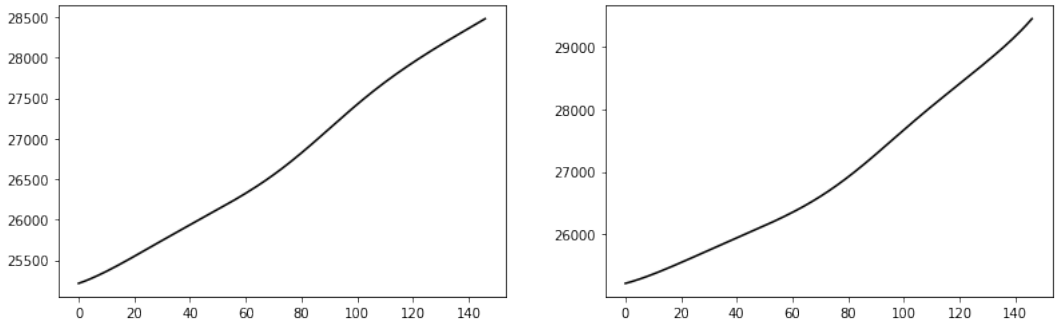


Figure 7.7: Deceased comparison between the optimal solution (left figure) and DPL (right figure). Case: $\int_0^{T_f} I(t)^2 \, \mathrm{d}t$.

Case 2

We present results obtained via optimization of the cost functional

$$J = \int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \, \mathrm{d}t,$$

where the $\dot{E}$ component is a regularization term. We set C_1 at $1e4$ to have comparable orders of magnitude between the two addenda. The cost functional achieves its minimum value ($1.80e12$) at iteration 100, without reaching the desired tolerance. In Figure 7.8 we report the evolution of the single compartments. We note that, except for the last 20 days, the recovered curve is always increasing. This means that administrations to recovered individuals are quite low until 120 days from the beginning of the campaign. Figure 7.9 shows percentage ripartitions of first doses, second doses and doses administered to recovered, together with a Least Square (LS) approximation with polynomials of order 6. Figure 7.10 compares first and second doses actually administered in Lombardy with the optimal solution. Figure 7.11 shows $\mathcal{R}_t$ indexes for the optimal strategy and the DPL. Moreover, for both cases we report the evolution of the deceased variable in Figure 7.12.

Case 2: Comments The main difference between the dashed (actual policy) and the continuous curve (optimal policy) in Figure 7.10 is the delay in second doses administrations until the 42nd day as in the case with minimization of the infectious curve. Furthermore, administrations of second doses are higher than in the previous case: the component of the cost functional related to variations in the exposed curve tends to completely immunize more people with respect to the previous case (Case 1).

We note that exposed curve in Figure 7.8 is flatter than the same picture in Figure 7.3. This is related to the presence of the $\dot{E}$ component which reinforces stabilization of the infected-classes curves. The level of deceased reached in this case is similar to the former one amounting at almost 28500. As for Case 1, we achieve better perfomances in terms of deceased individuals with respect to the DPL (see Figure 7.12). The $\dot{E}$-component stabilizes the exposed curve even though the infectious curve achieves higher values in the midterm period in this Case with respect to the previous one. This is only an apparent contradiction: cost functionals having a more complex structure need more iterations in order to display valuable enhancements.

Doses ripartitions in Figure 7.10 follow qualitatively the behaviour of the implemented campaign in Lombardy, except for the delay in the administrations of second doses we have previously underlined in Case 1. Even in this case when β increases second doses administrations increase, first doses decrease. Similarly to the previous case, $\mathcal{R}_t$ index is higher in the DPL than in the optimal strategy, especially in the long term where the maximum achieved value is 1.1 for the optimal policy, 1.6 for the direct problem.

We note unnatural peaks in the behaviour of $\mathcal{R}_t$, in the vaccination ripartition among doses and consequently in the evolution of the recovered variable at the end of the simulation time. Actually, evolution of the transmission rate β undergoes a deep discontinuity during the last 10 days of simulation, causing variations on each of the aforementioned quantities.

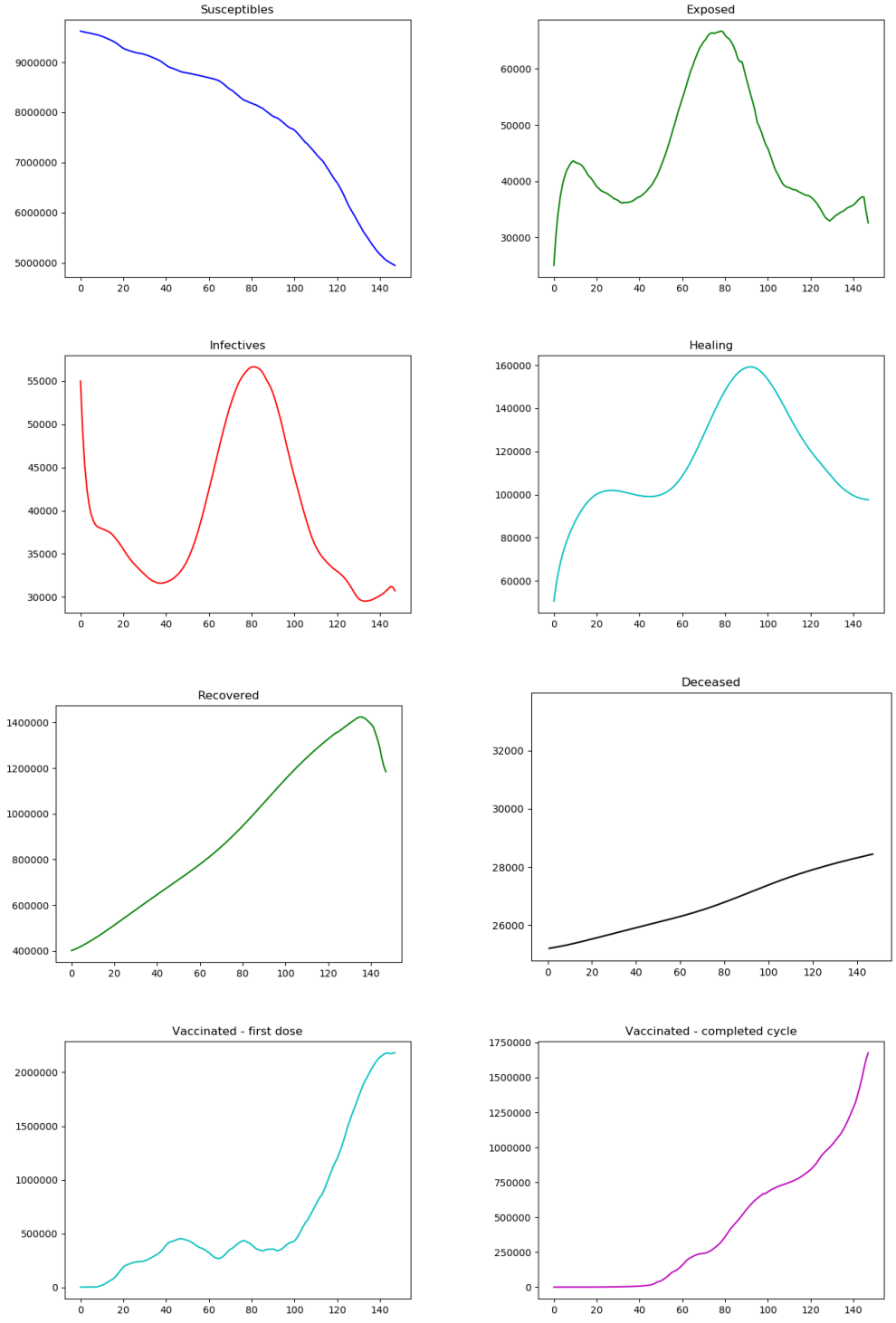


Figure 7.8: Evolution of each state of the `SEIHRDVW2` model.
Case: $J = \int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \, \mathrm{d}t$.

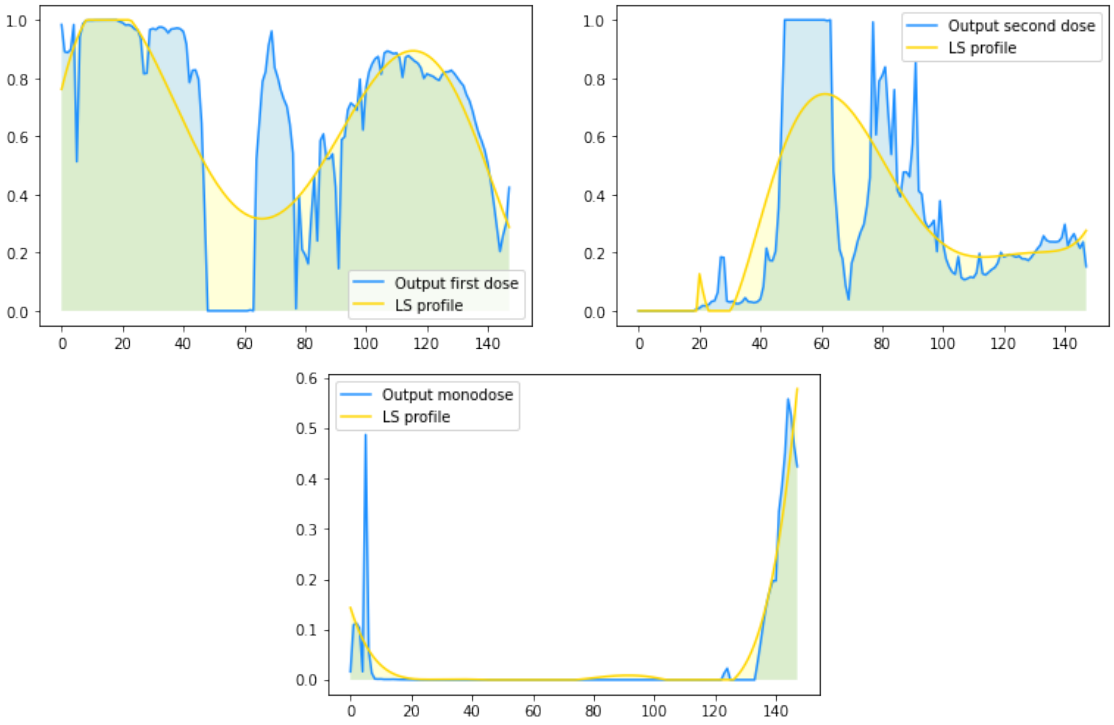


Figure 7.9: Percentages of first, second doses and doses administered to recovered individuals, together with *Least Square* interpolation of order 6. Case: $\int_0^{T_f} I(t)^2 + C_1\dot{E}(t)^2 \, \mathrm{d}t$.

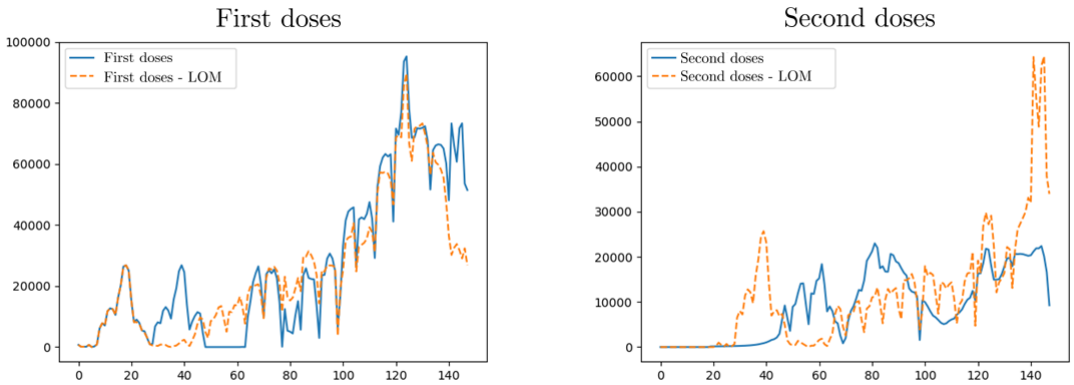


Figure 7.10: First and second doses comparison between actual ripartitions in Lombardy and optimal solution. In the first dose we also consider doses administered to recovered. Case: $\int_0^{T_f} I(t)^2 + C_1\dot{E}(t)^2 \, \mathrm{d}t$.

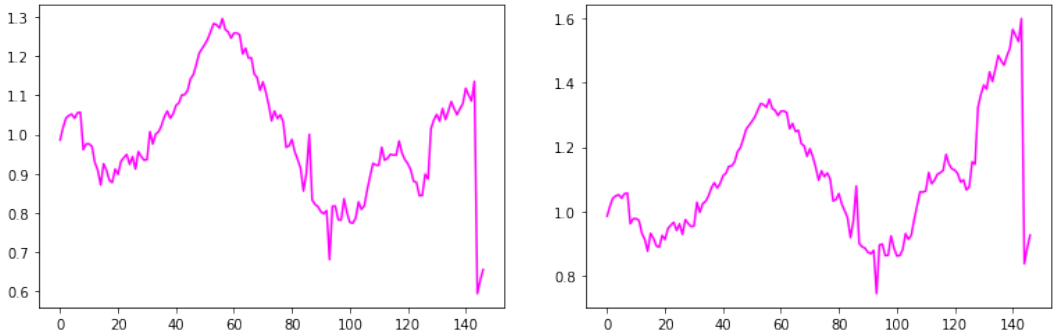


Figure 7.11: $\mathcal{R}_t$ comparison between the optimal solution (left figure) and DPL (right figure). Case: $\int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \, \mathrm{d}t$.

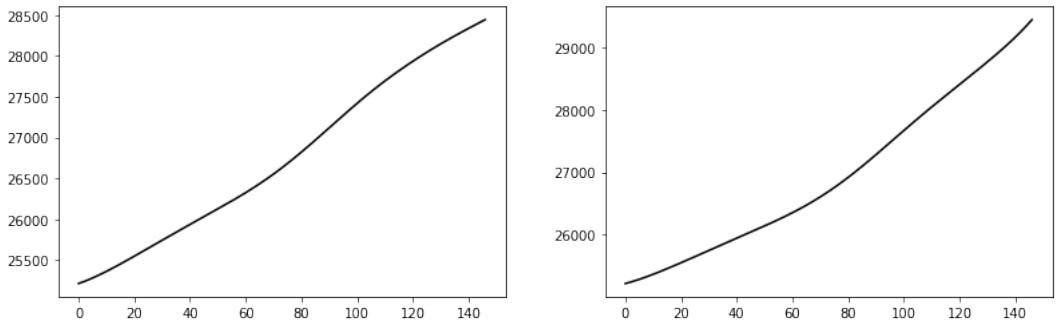


Figure 7.12: Deceased comparison between the optimal solution (left figure) and DPL (right figure). Case: $\int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 \, \mathrm{d}t$.

Case 3

We consider as third case the minimization of deceased individuals at the final time, namely

$$J = D(T_f)^2.$$

Simulation stops after 93 iterations, reaching the desired tolerance, and the final value of the cost functional is $8.30e8$, corresponding to almost 28800 deads.

In Figure 7.13 we introduce the evolution of each state variable involved in the `SEIHRDVW` model. The curve associated with recovered individuals is mildly increasing until the 120th day of simulation, when it rapidly decreases. We verify that, during the last period, a significant amount of recovered has been vaccinated. Figure 7.14 shows percentages of first, second doses and doses administered to recovered, together with the Least Square approximation with polynomials of order 6. Figure 7.15 compares actual administrations in Lombardy with first and second doses ripartition of the optimal solution. Figure 7.16 represents $\mathcal{R}_t$ index of the optimal solution versus the same index relative to the DPL. Figure 7.17 compares deceased evolution for the same two cases.

Case 3: Comments The distribution of first doses to recovered people is higher than in the previous cases standing at around 40% during the intermediate period. It implies a reduction in administered second doses with respect to the actual policy, see Figure 7.15, and an almost flat and then decreasing behaviour of the curve of recovered, see Figure 7.13.

The class of those vaccinated with the first dose is less fluctuating than in the previous case, and has a deep-sloped increasing during the last 27 days of simulation. Deceased people at the final time seem to be in line with previous results. Actually, we obtain a greater amount of deceased individuals in this case with respect to the ones previously analyzed.

After the inital transient stages of the vaccination campaign, doses ripartition reaches equilibrium for both three components: indeed, first doses to susceptibles stabilize at around 35-40%, first doses to recovered at 35-40%, and second doses at 20-30% until the 120th day of simulation (note that daily administrations are not constant). We remark that first doses administrations reach higher values than the actual policy implemented in Lombardy within the same period.

Vaccinations to recovered stop on the 120th day of simulation, with almost $1.5e6$ individuals belonging to this compartment. On the same day, the basic reproduction number $\mathcal{R}_t$ is decaying under the threshold value 1, and completely vaccinated individuals exceed $2e6$ individuals.

In this scenario, during the last simulation days exposed and infectious curves reach higher values than in the cases with other cost functionals.

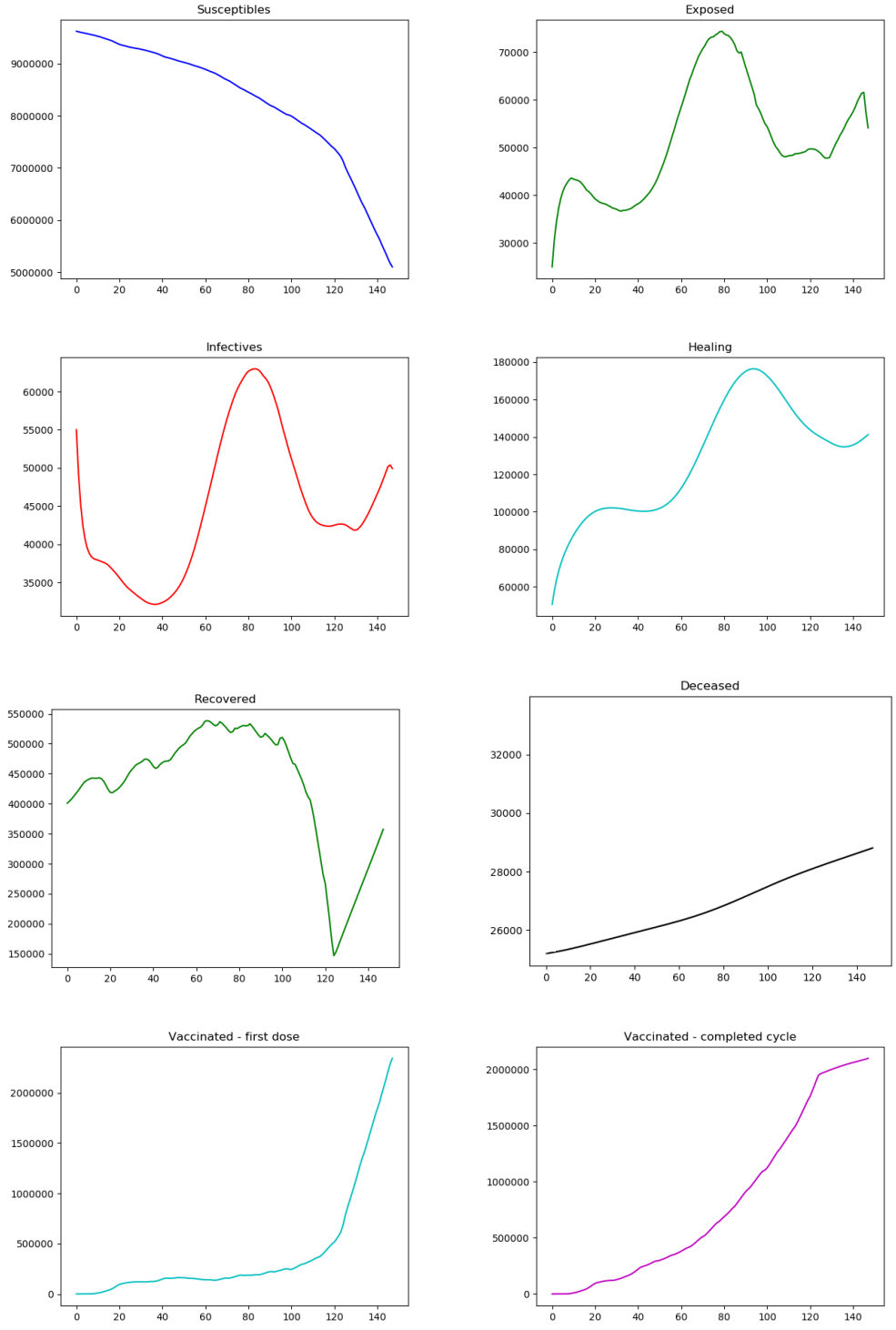


Figure 7.13: Evolution of each state of the `SEIHRDVW2` model. Case: $J = D(T_f)^2$.

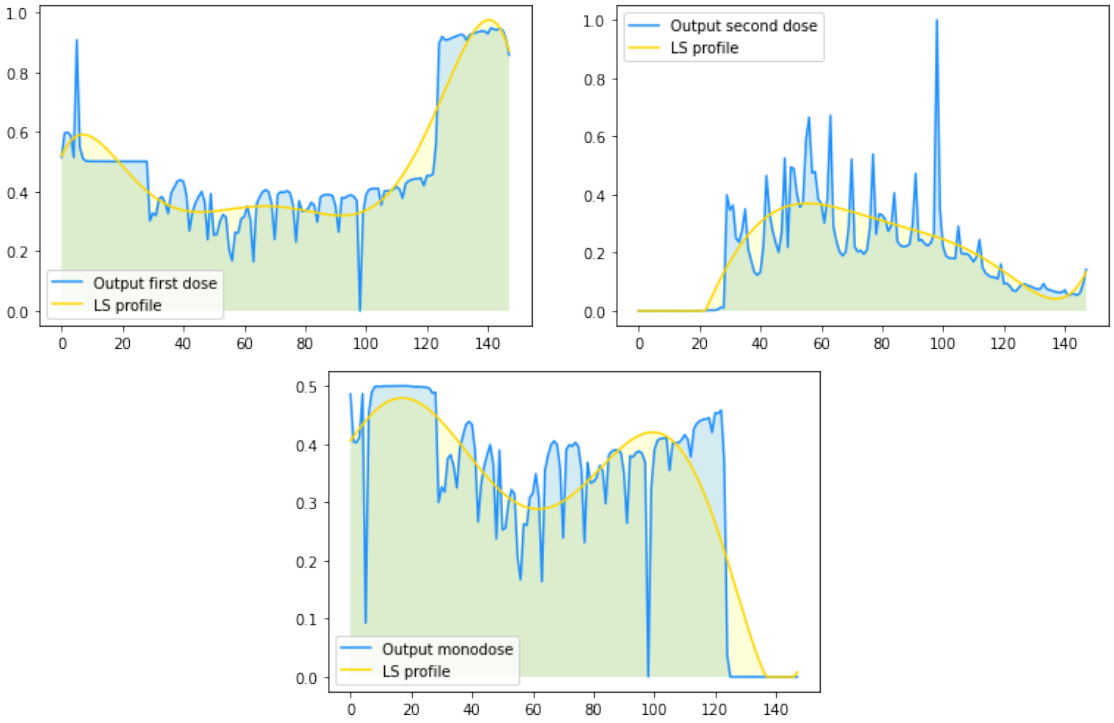


Figure 7.14: Percentages of first, second doses and doses administered to recovered individuals, overlapped with *Least Square* interpolation of order 6.
Case: $D(T_f)^2$.

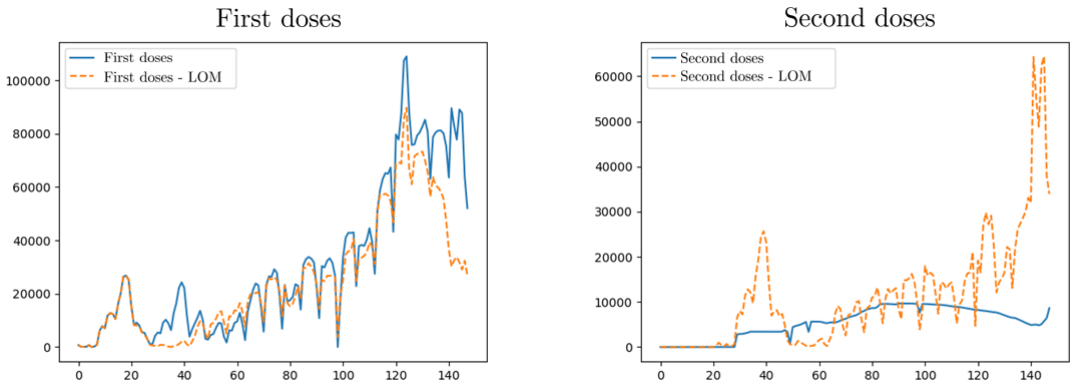


Figure 7.15: First and second doses comparison between actual ripartitions in Lombardy and optimal solution. In the first dose we also consider doses administered to recovered. Case: $D(T_f)^2$.

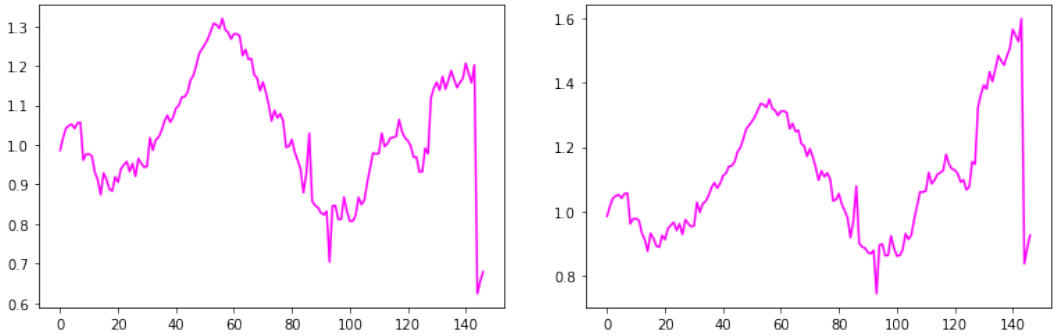


Figure 7.16: $\mathcal{R}_t$ comparison between the optimal solution (left figure) and DPL (right figure). Case: $D(T_f)^2$.

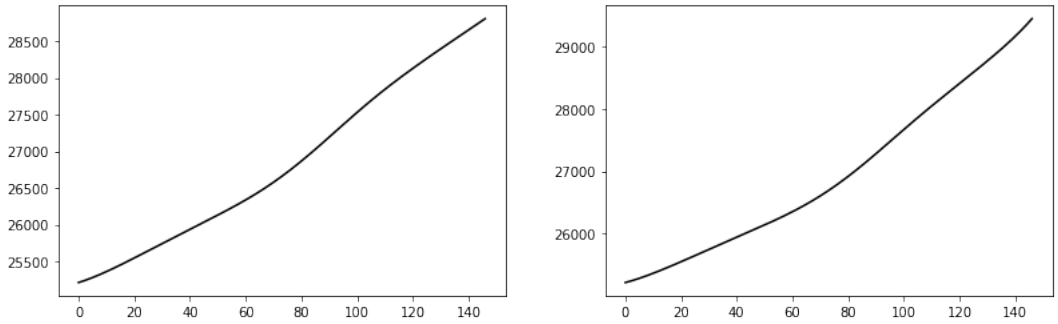


Figure 7.17: Deceased comparison between the optimal solution (left figure) and DPL (right figure). Case: $D(T_f)^2$.

Case 4

This paragraph deals with optimization of the more complex scenario where cost functional is constituted by a linear combination of all possible components which have been previously analyzed case by case, namely

$$J = \int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 dt + C_2 D^2(T_f),$$

where constants C_1 and C_2 have been fixed at $1e4$ and $1e3$ respectively. In this way we have compatibility among orders of magnitude of every addendum of the cost functional.
We aim at comparing solutions coming from Optimal Control Problem 1 and 2, meaning that we optimize the aforementioned cost functional fixing the daily amount of administrations or the weekly amount of administrations. For recovering a feasible amount of weekly delivered doses to administer, we extract from data available by the Italian department of Protezione Civile [92] the amount of daily delivered doses in

Lombardy in the period of interest, and then we sum these values for weeks[1]. In the case corresponding to OC Problem 2, we fix the maximum administration capacity C at 140000, corresponding to the actual highest value of administrations in Lombardy until June 2021. If some doses are delivered but not administered in a certain week, the algorithm considers the exceeding quantity as stocks for the following week. Figures 7.18-7.22 represent states evolutions, doses distributions and $\mathcal{R}_t$ indexes for the two cases.

Case 4: Comments Solutions related to the case of optimization of OC Problem 1 are qualitatively and quantitatively similar to those of Case 2 ($\int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 dt$), meaning that with 100 iterations and tolerance fixed at $1e-6$, it is not possible to achieve better performances than those obtained in Case 2. Hence, the final time functional term related to deaths does not play any role in the minimization process with this parameters setting.

The simulation referring to Problem 2 returns different solutions with respect to the other cases. At first, from the state analysis we observe that at final time suceptibles are higher than the other case (respectively almost 6 and 5 millions). Even the curve representing individuals which have been completely immunized (violet curve) is higher than in the other case. Less people have been infected in the second case with respect to the first one. Even the infected peak is lower when we consider OC Problem 2, namely 53000 infectives versus 55000 of OC Problem 1. Deceased individuals reach instead almost the same value in both simulations.

In Figure 7.20, we compare doses distribution percentages among the two cases. We notice that, in the second case, the average ripartition of doses is half-and-half between first and second administrations to susceptibles. Furthermore, the optimal solution accounts for days (depicted in red) where no doses are administered since available doses in the same week have already run out. From Figure 7.21 we deduce that second doses administrations are by far bigger in the second case than the same administrations in the first case, meaning that OC Problem 2 intents to completely immunize as many people as possible.

In conclusion, $\mathcal{R}_t$ indexes are comparable without evident differences between the two cases.

[1]We remark that during 140 days starting from 01-01-2021 not all delivered doses in Lombardy have been administered, *e.g.* on 17th of June 2021 90.9% of total delivered doses have been adminstered.

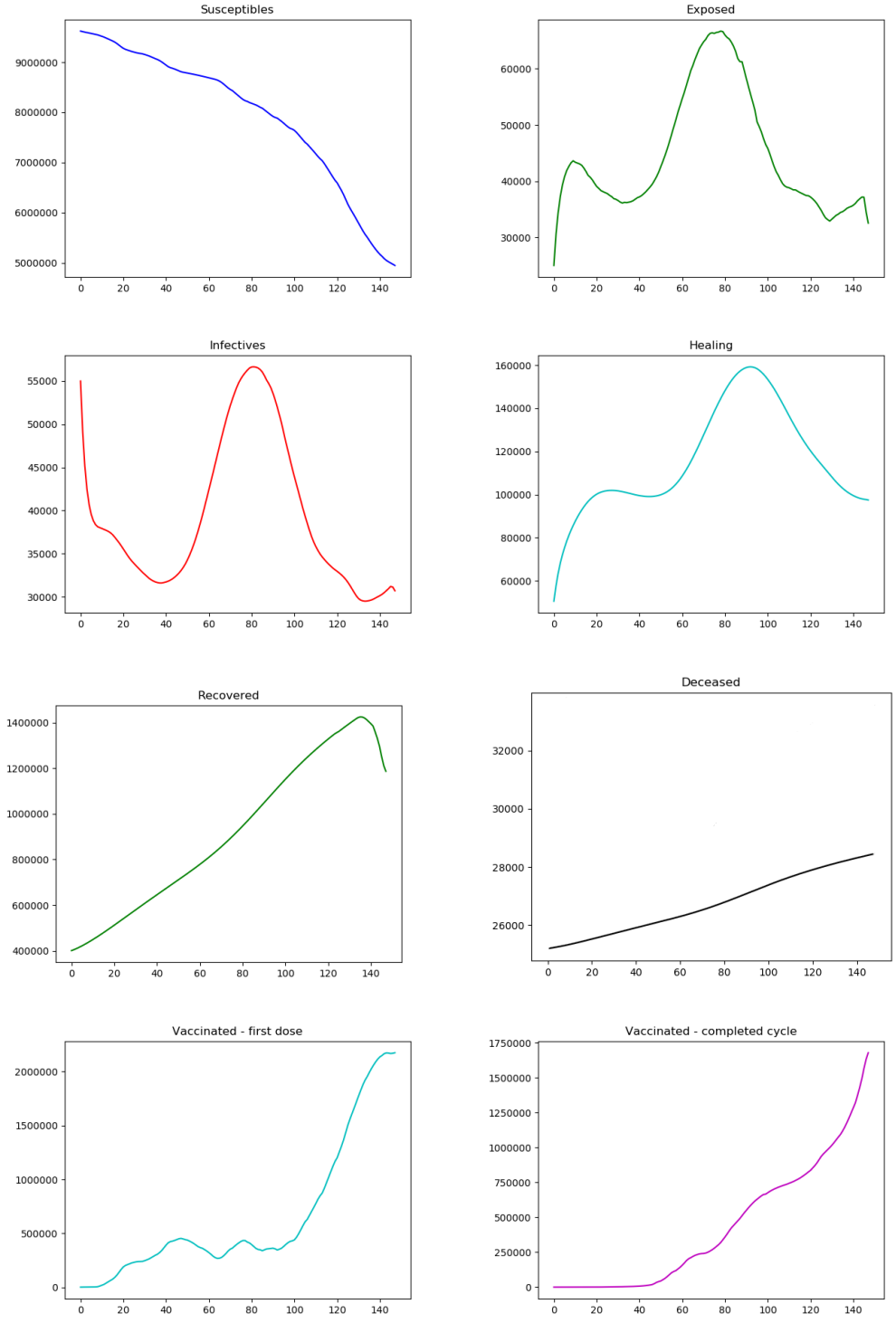


Figure 7.18: Evolution of each state of the `SEIHRDVW2` model (OC Problem 1). Case: $J = \int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 dt + C_2 D^2(T_f)$.

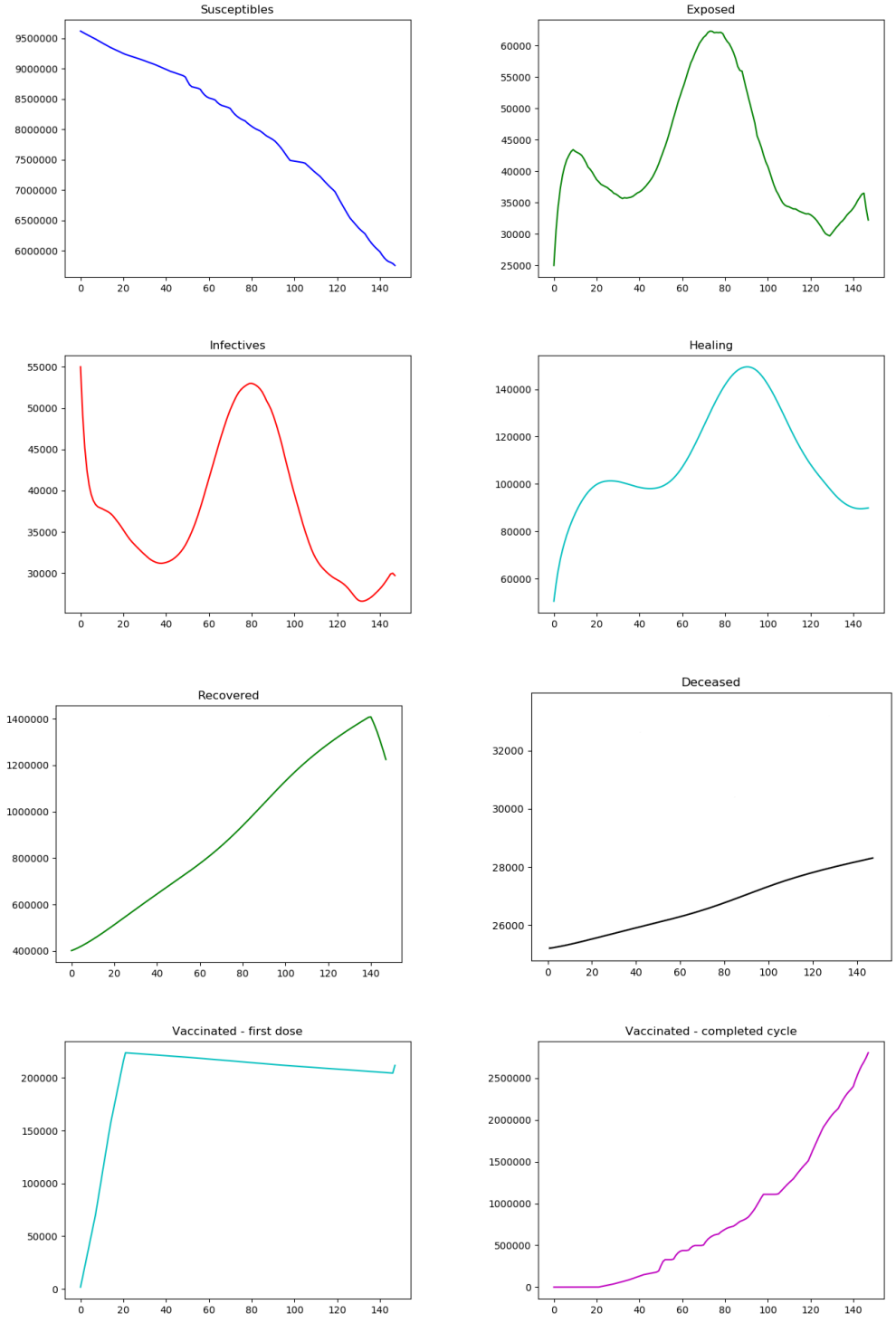


Figure 7.19: Evolution of each state of the `SEIHRDVW2` model (OC Problem 2). Case: $J = \int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 dt + C_2 D^2(T_f)$.

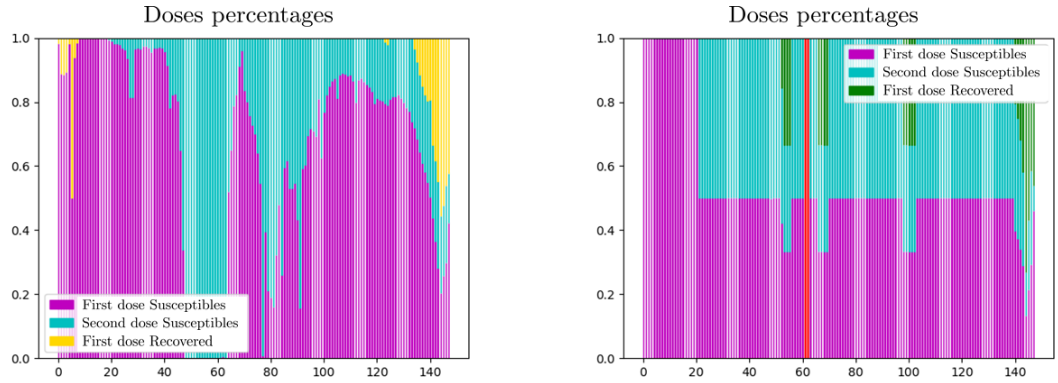


Figure 7.20: Comparison of histograms representing daily percentages of doses in the case of OC Problem 1 (left figure) and OC Problem 2 (right figure). In the left figure, days colored in red stand for days without administrations.
Case: $\int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 dt + C_2 D^2(T_f)$.

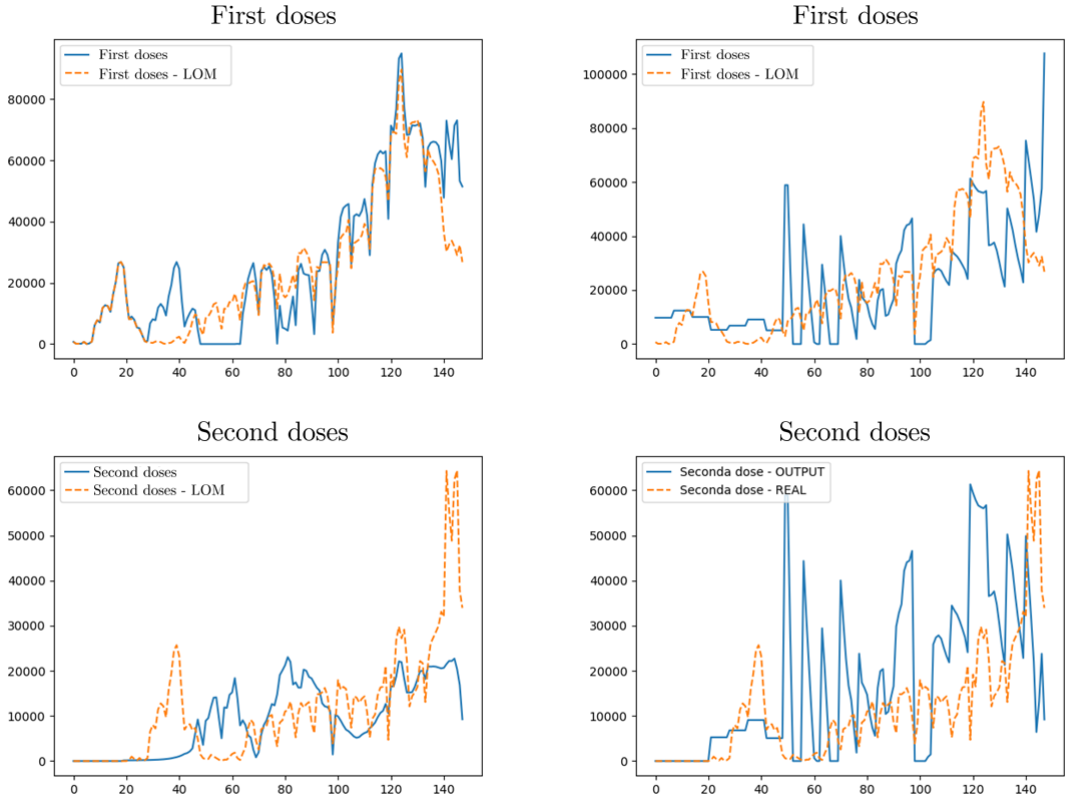


Figure 7.21: First and second doses comparison between actual ripartitions in Lombardy and optimal solution for both formulations of OC Problem 1 (left figures) and OC Problem 2 (right figures). In the first dose we also consider doses administered to recovered. Case: $\int_0^{T_f} I(t)^2 + C_1 \dot{E}(t)^2 dt + C_2 D^2(T_f)$.

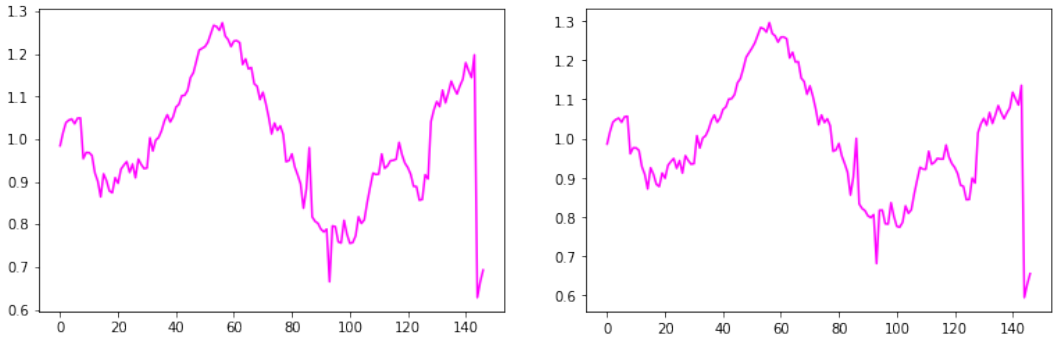


Figure 7.22: $\mathcal{R}_t$ comparison between OC Problem 1 (left figure) and OC Problem 2 (right figure). Cost functional: $\int_0^{T_f} I(t)^2 + C_1\dot{E}(t)^2 dt + C_2 D^2(T_f)$.

7.2 Optimal strategies for two vaccines against SARS-CoV-2

In this section, we aim at investigating optimal vaccination policies obtained taking into account for vaccine parameters, *i.e.* vaccine effectivenesses and t_{min}, t_{max} elapsing times among doses, compatible with two vaccines for SARS-CoV-2 disease: the **Comirnaty (BNT162b2)**, produced by the pharmaceutical firm BioNTech/Pfizer, and the **Vaxzevria (ChAdOx1-S)**, developed jointly by Oxford University and AstraZeneca.

Many studies, most of which published on the scientific journal *The Lancet*[2], are devoted to explore peculiar efficacy properties of these authorized vaccines. Since neither of the two guarantee complete immunity after the cycle is completed, the `SEIHRDVW2` model has to be modified as follows:

$$\begin{cases} \dot{S} = -\beta \dfrac{SI}{N} - U_1 + \mu_R R + \mu_V V \\ \dot{E} = \beta \dfrac{(S + \sigma V + \sigma_W W) I}{N} - \alpha E \\ \dot{I} = \alpha E - \gamma I \\ \dot{H} = \gamma I - \omega H \\ \dot{R} = (1 - f(S,V,W))\, \omega H - \mu_R R - U_2 \\ \dot{D} = f(S,V,W)\, \omega H \\ \dot{V} = -\beta \dfrac{\sigma V I}{N} + U_1 - U_3 - \mu_V V \\ \dot{W} = U_2 + U_3 - \beta \dfrac{\sigma_W W I}{N}, \end{cases} \tag{7.1}$$

where the fatality function reads as

$$f(S,V,W) = \bar{f} \frac{S(t-15) + \theta\sigma V(t-15) + \theta_W \sigma_W W(t-15)}{S(t-15) + \sigma V(t-15) + \sigma_W W(t-15)}.$$

Through the introduction of parameters σ_W and θ_W, representing vaccine efficacies for transmission and mortality after the second dose, the model is closer to the real behaviour of the adopted vaccines. However, a multivaccine model contempling all vaccines that have actually been authorized by EMA would perform better for the purpose of simulating the complete realistic scenario.

We solve Optimal Control Problem 2 for both vaccines. PGD parameters are set as follows: tolerance $1e-6$, maximum number of iterations 100, step $1e-2$. We

[2] `https://www.thelancet.com/coronavirus`

fix $N_{DosesDeliveredPerWeek}$ at 385000, the maximum daily administration capacity C at 140000 and the transmission rate β at 0.26172 as in Chapter 6. We refer to Table 6.1 for the Initial Conditions of the state problem. Except for vaccine efficacies, we fix other parameters as in Table 6.2.

7.2.1 Comirnaty (BioNTech/Pfizer)

We assume to deal with BioNTech/Pfizer vaccine. This vaccine is mRNA-based, *i.e* it contains messenger RNA (mRNA) molecules that have inside of them the building blocks for the SARS-CoV-2 Spike proteins. In the vaccine, the mRNA molecules are placed in a microscopic lipid vescicle, a *bubble* that protects the mRNA from being lost and destroyed by the immune system's defences as a foreign component, so that it can enter cells. Once the vaccine has been injected, the mRNA is absorbed into the cytoplasm of the cells and triggers the synthesis of Spike proteins. Their presence thus stimulates the immune system's production of specific antibodies. Vaccination also activates T-cells that prepare the immune system to respond to further exposure to the SARS-CoV-2 virus.

We fix $t_{min} = 21$ days, $t_{max} = 42$ days and vaccine effectivenesses as in Table 7.1. Those values have been extracted from [110] and [25], clinical trials conducted in order to get evidence about vaccine's efficacy.

Parameter	**Value**
σ	0.20
θ	0.074
σ_W	0.10
θ_W	0.06

Table 7.1: Effectivenesses of the BioNTech/Pfizer vaccine after one and two administrations.

We solve the optimization Problem 2 with two different cost functionals:

- $J_1 = \int_0^{T_f} I(t)^2 \, \mathrm{d}t$ (left figures);
- $J_2 = D(T_f)^2$ (right figures).

Figure 7.23 represents time evolution of each state variable involved. Notable differences of the two solutions are represented by the evolution of variables V and W which are associated with the vaccinated individuals.

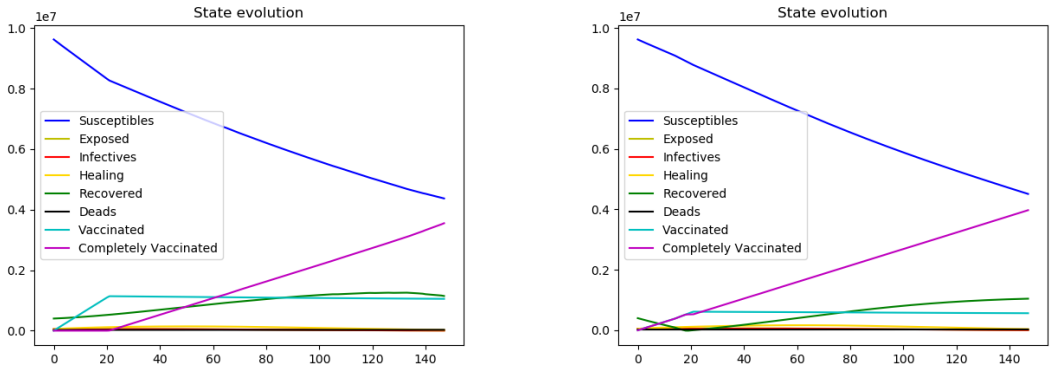


Figure 7.23: State variables evolution of the OC solutions for the BioNTech/Pfizer vaccine, obtained optimizing $J_1 = \int_0^{T_f} I(t)^2\,dt$ (left figure) and $J_2 = D(T_f)^2$ (right figure).

Figures 7.24 and 7.25 represent the evolution of partially and completely vaccinated individuals respectively. Left strategy does not assume to administer doses to recovered during the first 20 days from the beginning, while this is not true for the second strategy. Variable $V(t)$ is qualitatively similar in the two cases, but quantitatively distinct among the two. Indeed, minimizing infectious individuals imply to mantain in the V-class almost $1e6$ individuals, while minimizing deaths almost the half ($6e5$). The opposite occur with variable $W(t)$, which reaches higher values for the right case ($4e6$) than for the left case ($3.5e6$) in the same timeframe.

We intuitively deduce that completed vaccinated, that are associated with better vaccine effectivenesses ($\sigma_W < \sigma$ and $\theta_W < \theta$), are preferred to one-dose-vaccinated individuals for minimizing deaths, even though this implies that less people receive partial coverage through a single administration (in agreement with Section 6.1).

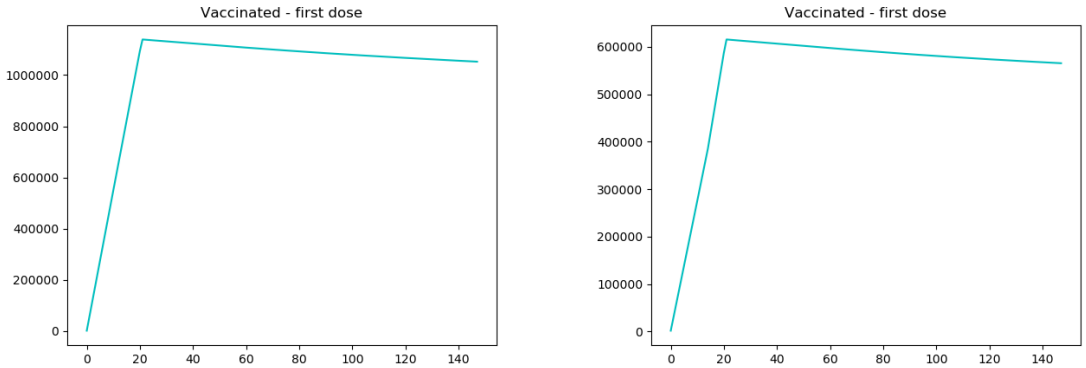


Figure 7.24: Evolution of single-dose vaccinated individuals with the BioNTech/Pfizer vaccine, obtained optimizing $J_1 = \int_0^{T_f} I(t)^2\,dt$ (left figure) and $J_2 = D(T_f)^2$ (right figure).

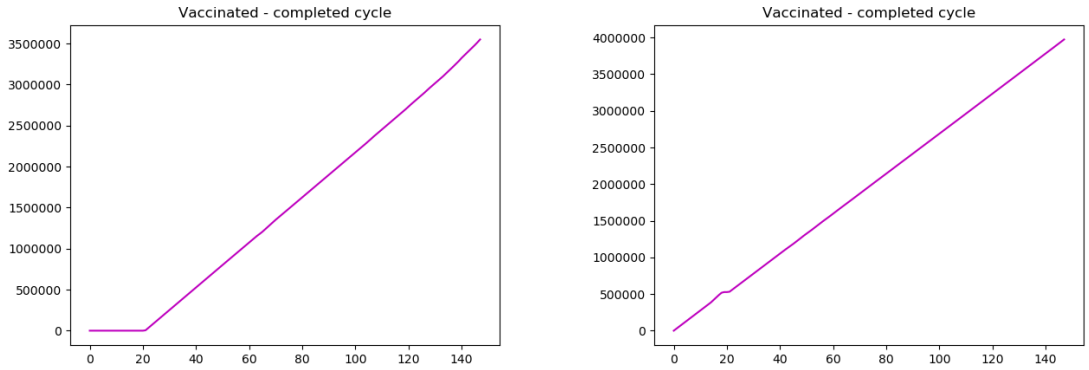


Figure 7.25: Evolution of completely vaccinated individuals with the BioNTech/Pfizer vaccine, obtained optimizing $J_1 = \int_0^{T_f} I(t)^2\,dt$ (left figure) and $J_2 = D(T_f)^2$ (right figure).

Figure 7.26 depicts the evolution of each of the three control variables. Apart from the initial 20 days of simulation, first and second doses administrations to susceptibles equilibrate to an half-and-half ripartition for almost all times. Instead, recovered individuals are vaccinated during the first 20 days of simulations just in the second strategy. The first strategy starts irregular vaccinations to recovered during the last 20 days of simulation. We obtain oscillations in the total amount of administered doses in both cases, which were not present in simulations with the standard `SEIHRDVW2` model.

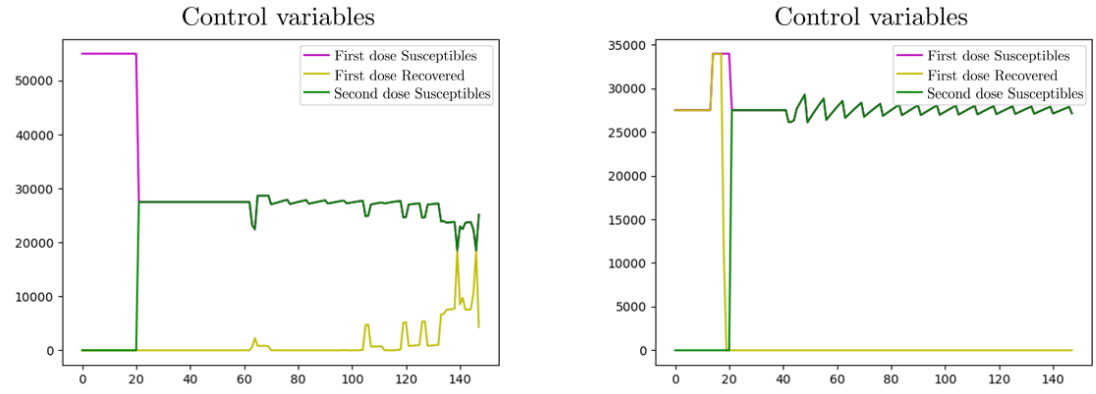


Figure 7.26: Control variables evolution obtained optimizing $J_1 = \int_0^{T_f} I(t)^2\,dt$ (left figure) and $J_2 = D(T_f)^2$ (right figure). We consider the BioNTech/Pfizer vaccine.

7.2.2 Vaxzevria (AstraZeneca)

This section is devoted to the analysis of the results of simulations run with parameters compatible with AstraZeneca vaccine. This vaccine is composed of a chimpanzee adenovirus unable to replicate (ChAdOx1 - Chimpanzee Adenovirus Oxford 1) and modified to carry the genetic information to produce the Spike protein of the SARS-CoV-2 virus.

We set vaccine-related parameters as follows: $t_{min} = 21$, $t_{max} = 84$ and vaccine efficacies as in Table 7.2. For the calibration of those parameters, we refer to [6] and [116].

Parameter	**Value**
σ	0.33
θ	0.24
σ_W	0.25
θ_W	0.18

Table 7.2: Effectivenesses of the AstraZeneca vaccine after one and two administrations.

We simulate the problem with two different cost functionals, as in the BioNTech/Pfizer case:

- $J_1 = \int_0^{T_f} I(t)^2 \, dt$ (left figures);
- $J_2 = D(T_f)^2$ (right figures).

Figures 7.27-7.29 show time evolutions of all states with a particular focus on classes regarding the vaccination process ($V(t)$, $W(t)$ respectively).

In both left and right cases, susceptibles decrease to almost $5e6$, whilst completed cycle vaccinated individuals achieve almost $4e6$. As we observe in Figure 7.28, $V(t)$ has a growing trend during the first 20 days from the beginning of the simulation. On the 20th day, second doses administrations to susceptibles start and $V(t)$ decreases reaching different values in the two cases (respectively $1e6$ and $5.5e5$ for left and right case).

Completed vaccinated people have a growing trend from the 20th to 147th day in both cases. While for the infectious case recovered individuals have not been vaccinated during the first period, this is not true for the other case (Figure 7.29).

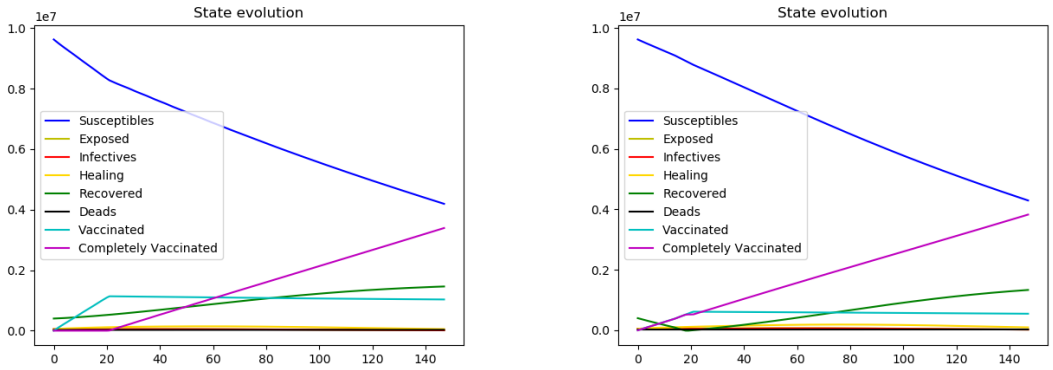


Figure 7.27: State variables evolution of the OC solutions for the AstraZeneca vaccine, obtained optimizing $J_1 = \int_0^{T_f} I(t)^2\,dt$ (left figure) and $J_2 = D(T_f)^2$ (right figure).

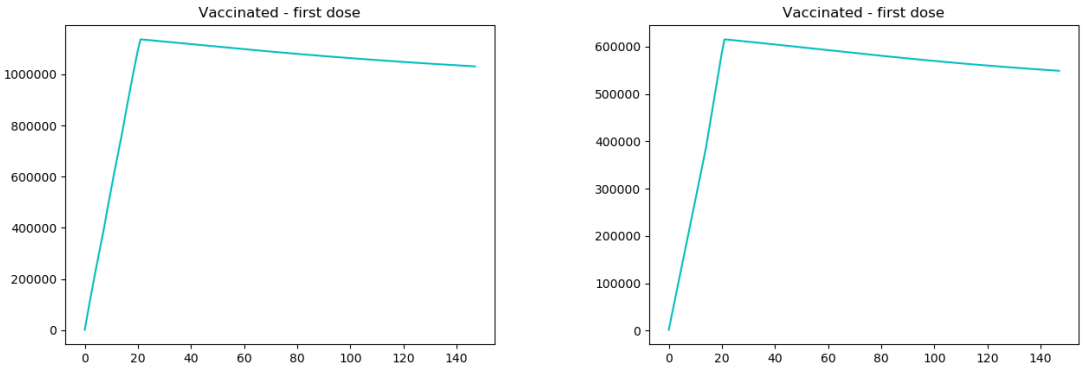


Figure 7.28: Evolution of single-dose vaccinated individuals with the AstraZeneca vaccine, obtained optimizing $J_1 = \int_0^{T_f} I(t)^2\,dt$ (left figure) and $J_2 = D(T_f)^2$ (right figure).

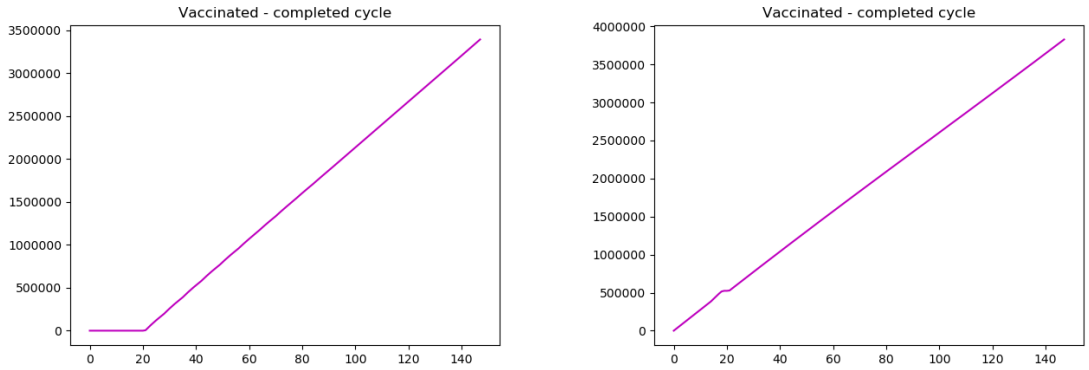


Figure 7.29: Evolution of completely vaccinated individuals with the AstraZeneca vaccine, obtained optimizing $J_1 = \int_0^{T_f} I(t)^2\,dt$ (left figure) and $J_2 = D(T_f)^2$ (right figure).

Figure 7.30 represents the evolution of each control variable for the two optimal solutions. We interpret these results in the following way: during the first 20 days, when infectious classes are not at their highest levels, the strategy minimizing deaths tries to create as much immunized people as possible, distributing half doses to recovered and half doses to susceptibles. For minimizing infectious, in the same period doses are administered exclusively to susceptibles. Then, both solutions agree in administering equally first and second doses to susceptibles, which are the most fragile individuals in the infectious process, delaying immunisation of recovered.

Finally, oscillations of first and second doses, that are present from the beginning of the simulation in the left case and from the 90th day in the right case, imply that the number of total daily administered doses is not constant. This is an unexpected result in a scenario where parameters have been set to constants. Indeed, oscillations are probably caused by the lack of convergence of the minimization algorithm, which has not reached the desired tolerance.

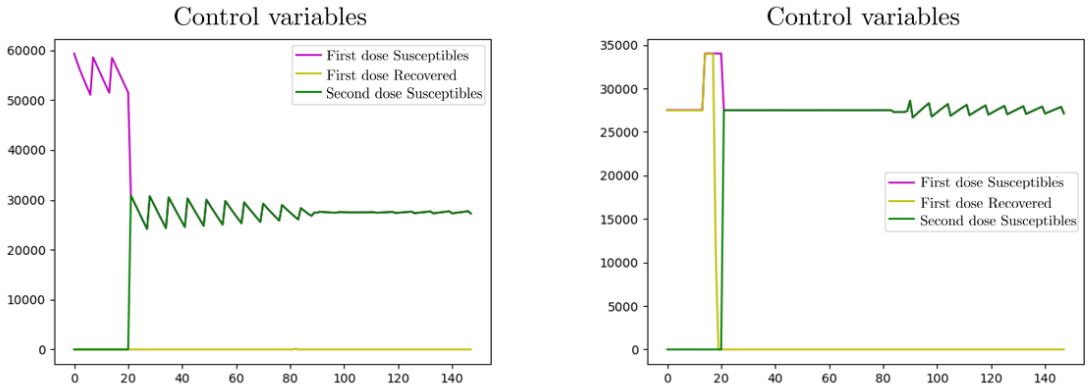


Figure 7.30: Control variables evolution obtained optimizing $J = \int_0^{T_f} I(t)^2\,dt$ (left figure) and $J = D(T_f)^2$ (right figure). We consider the AstraZeneca vaccine.

Although we considered two vaccines with different values of effectivenesses, we do not outline any significant discrepancies in the optimal vaccination strategies for Comirnaty and Vaxzevria vaccines: both solutions of OC Problem 2 minimizing J_1 assest to an equal ripartition of first and second doses, and a similar strategy, apart from the first 20 days of simulation, is optimal to minimize J_2, in agreement with Section 6.4.3.

Chapter 8

Conclusions

With the aim of contributing to the panorama of mathematical modeling for the current pandemic in Italy, in this work we investigated the numerical modelling of the optimal vaccination campaign for the SARS-CoV-2 epidemic disease minimizing the infectious spread or the number of deaths caused by the infection itself. We formulated a new compartmental epidemic model, named `SEIHRDVW`, incorporating the vaccination dynamics. We proposed and solved two different Optimal Control problems, based on the knowledge of available daily or weekly vaccines supply. We implemented an innovative optimisation algorithm, the Multi-Projected Gradient Descent, which returns the optimal vaccination campaign satisfying constraints imposed by pharmacological factors and availability of doses.

For our purposes, we ran several numerical experiments testing the model both in artificial and semi-realistic scenarios. In particular the semi-realistic scenario is related to the vaccination campaign in Lombardy beginning on the 1st January 2021. We considered the impact on the optimal vaccination policy of:

a. **Social Restrictions.** We determined plausible values for three different levels of social restrictions and solved optimal control problems for minimizing the infectious spread. We obtained that the optimal vaccination strategy assests to precise percentage ripartition among doses when restrictions become more prohibitive;

b. **Variants.** We underlined adjustments in the optimal vaccination strategy when more transmissible virus strains come out. In this case a fundamental role is played by administrations to recovered, that increase in the optimal solution with respect to the one referred to the base strain. We also accounted for variants affecting vaccine effectivenesses, without noting any evident discrepancy among optimal policies in each case of study;

c. **Time-dependant transmission rate and vaccinations supply.** From online available data for Lombardy region, we extracted approximations of the daily doses administrations, weekly delivered doses and a time-variable transmission

rate. We compared the optimal vaccination policy with the solution of the direct `SEIHRDVW` problem setting daily administrations as in the real vaccination campaign in Lombardy. The most relevant differences between the two solutions consisted in the delay in the beginning of administrations of second doses, which was a common feature of optimal solutions with different cost functionals. The heterogeneity of daily administrations made us difficult to extrapolate a concrete optimal vaccination strategy;

d. **Two vaccines for protecting against COVID-19.** We aimed at highlighting differences and similarities between optimal vaccination policies assuming to administer BioNTech/Pfizer and AstraZeneca vaccines. Actually, the solutions of the two optimal control problems fixing the amount of weekly delivered doses have shown no significant discrepancies with the chosen parameter setting.

Future Developments

The mathematical models and the algorithm we introduced in this dissertation are definitely beneficial for orienting qualitatively optimal vaccination campaigns. At the same time, they are useless for quantitative predictions without proceeding with the following improvements, which can be objectives of future researches:

a. **Calibration of the parameters.** For running realistic simulations, we need to extimate values of parameters and initial conditions with an high level of accuracy. We can address the calibration problem through a *Montecarlo Markov Chain* approach, as in [89], through bayesian statistics techniques. Another possibility is to solve the optimization Classical Inverse Problem (CIP), similar to the one introduced in Section 3.4, for each of the parameters involved in `SEIHRDVW`. Moreover, we can tackle the problem through the implementation of a Residual Neural Network (`ResNet`). Indeed, ResNets have been recently investigated as solutors of optimal control problems analogous to the CIP (see [10]);

b. **Extension of the model.** The `SEIHRDVW` model we considered is caged into some hypotheses that can be relaxed in order to collect more realistic and practicable vaccination policies. Here we focus on three possible directions:

   b1. with the purpose of capturing the variable response to infection by individuals of different ages, a *multi-age model* can be formulated, increasing the computational demand of each step of the Optimal Control algorithm;

   b2. since inner and outer fluxes across the domain of interest were not contemplated in the previous model, we can expand the `SEIHRDVW` to a *multi-city* formulation. For instance, the mobility component can be represented by a directed graph having cities as vertices and arcs for ingoing or outgoing travels, as in [4];

b3. until September 2021 four different vaccines against SARS-CoV-2 have been contemporarily administered in the Italian scenario. Hence, we can build a *multi-vaccine* model as the one proposed in Figure 8.1.

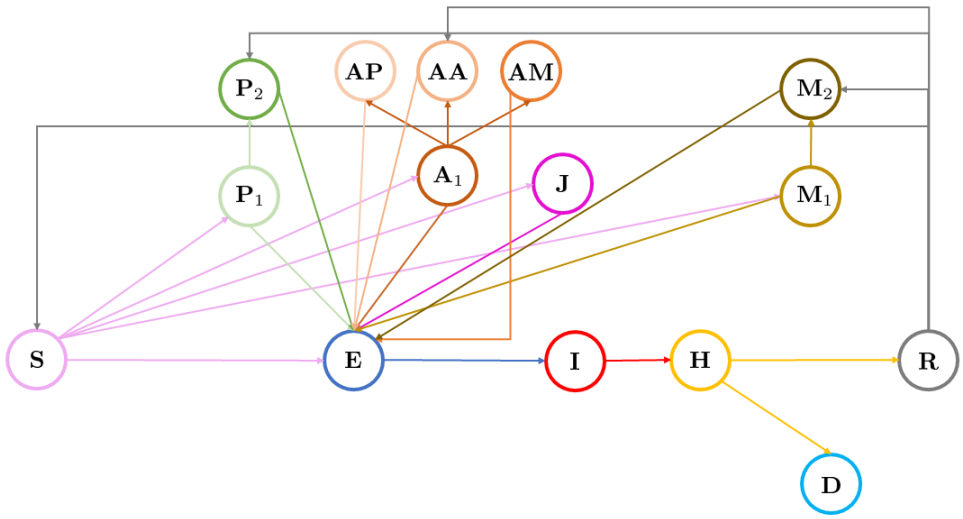


Figure 8.1: Multi-vaccine model englobing homologous and heterologous vaccination policies for SARS-CoV-2. Legend: **S**, Susceptibiles; **E**, Exposed; **I**, Infected; **H**, Healing; **R**, Recovered; **D**, Deceased; $\mathbf{P}_1$, Vaccinated with first dose BioNTech/Pfizer; $\mathbf{P}_2$, Vaccinated with two doses of BioNTech/Pfizer; $\mathbf{A}_1$, Vaccinated with first dose AstraZeneca; **AA**, Vaccinated with two doses of AstraZeneca; **AP**, Vaccinated with first dose of AstraZeneca and second dose of BioNTech/Pfizer; **AM**, Vaccinated with first dose of AstraZeneca and second dose of Moderna; **J**, Vaccinated with monodose Johnson & Johnson vaccine; $\mathbf{M}_1$, Vaccinated with first dose of Moderna dose 1; $\mathbf{M}_2$, Completely vaccinated with two doses of Moderna.

Appendix A

Attractivity and stability: an example

Consider $X = \mathbb{R}^2$ and the following system in polar coordinates

$$\begin{cases} \dot{r} = r(1-r), \\ \dot{\theta} = r(1-\cos\theta). \end{cases} \tag{A.1}$$

One equilibrium point for the system is $\bar{x} = (r = 1, \theta = 0)$. In the case $r(0) = 1, \theta(0) = \theta_0 \neq 0$ the solution is given by

$$\theta(t) = \pi + 2\arctan(\cot\frac{\theta_0}{2} + t),$$

therefore whatever neighbourhood of $\bar{x}$ we can choose, the orbit is not confined in it even though, as $\theta(t)$ approaches 2π, it slows down towards the equilibrium point.

Besides, integrating the evolution of $r(t)$ in the dynamical system we obtain,

$$r(t) = \frac{r_0 e^t}{1 - r_0 + r_0 e^t},$$

which is bounded and tends to 1 as $t \to +\infty$ for any r_0. Therefore, solutions tend to the equilibrium point, even though, taken an arbitrary small neighbourhood U, the solution with $\theta_0 > 0$ will not stay confined in the interval.

Appendix B

Functions with bounded integral and first derivative: Theorem

Theorem 12. *Let $f(t) : \mathbb{R}^+ \to \mathbb{R}^+$ a differentiable function such that:*

1. $\int_0^\infty f(t)\ \mathrm{d}t < \infty$;
2. $f'(t) < k < \infty\ \ \forall\, t \in \mathbb{R}^+$.

Then,

$$\lim_{t\to\infty} f(t) = 0.$$

Proof. Assume by contradiction that $f(t) \nrightarrow 0$. It implies that, fixing an arbitrary $\epsilon > 0$,

$$\exists \{t_n\}_{n\in\mathbb{N}} \to \infty,\ f(t_n) > \epsilon.$$

Hence, below the graph of f we can construct a sequence of triangles $\{T_n\}_{n\in\mathbb{N}}$ centered on each t_n such that

$$\mathrm{area}(T_n) \propto \frac{1}{k^2},$$

independently on n. Then,

$$\int_0^\infty f(t)\ \mathrm{d}t > \sum_{n=0}^{\infty} \mathrm{area}(T_n) \propto \sum_{n=0}^{\infty} \frac{1}{k^2} = \infty,$$

which contradicts hypothesis 1. □

Appendix C

Optimization Codes

We report below the `python3` code related to the solution of Optimal Control Problem 1 (with `SEIHRDVW2` model as state problem), where we fixed the daily amount of doses to administer.

```
import jax
import os
os.environ["XLA_FLAGS"]="--xla_gpu_cuda_data_dir=/usr/lib/cuda"
os.environ["CUDA_HOME"]="/usr/lib/cuda"
import sys
import jax.numpy as jnp
import numpy as np
import matplotlib.pyplot as plt
import math
import pandas as pd

#SEIHRDVW evolution function
def SEIHRDVW(t, state, params, U1, U2, U3, beta, state_15_0,
    state_15_1):

  # alpha: inverse of the incubation time
  # gamma: infectous traspassing rate
  # muR: re-infection rate from recovered class
  # muV: re-infection rate from vaccinated class
  # sigma: fraction of vaccine effectiveness on infection trasmission
  # omega: hospedalizing rate
  # theta: fraction for accounting for vaccine effectiveness on
    effects reduction
  # f: fatality rate
  # state_15:  contains S and V 15 days before

  alpha, gamma, muR, muV, sigma, omega, theta, f= params

  N = np.sum(state)
  S = state[0]
  E = state[1]
```

```
  I = state[2]
  H = state[3]
  R = state[4]
  D = state[5]
  V = state[6]
  W = state[7]

  dSdt = - beta * S * I / N - U1 + muR * R + muV * V
  dEdt = beta * (S + sigma * V) * I / N  - alpha * E
  dIdt = alpha * E - gamma * I
  dHdt = gamma * I - omega * H
  dRdt = (1.0 - f * (S + theta * sigma * V) / (S + sigma * V)) *
   omega * H - muR * R - U2
  dDdt = f * (state_15_0 + theta * sigma * state_15_1) / (state_15_0
   + sigma * state_15_1) * omega * H
  dVdt = - sigma * beta * V * I / N + U1 - muV * V - U3
  dWdt = U2 + U3

  return jnp.array([dSdt, dEdt, dIdt, dHdt, dRdt, dDdt, dVdt, dWdt])


# Runge-Kutta 4 method for the state SEIHRDVW problem
def solve_rk4(fun, t_span, y0, h, params, controls, states_15_start):
    nsteps = int((t_span[1]-t_span[0])/h)
    y = np.zeros((len(y0),nsteps+1))
    y[:,0] = y0
    U1, U2, U3, beta = control_at_time(controls, 0)

    for i in range(nsteps):
        t = int(t_span[0]+i*h)
        U1_1, U2_1, U3_1, beta_1 = control_at_time(controls, t+1)
        y0 = y[:,i]

        if i >= 15:

            k1=fun(t      , y0          , params, U1, U2, U3, beta, y
  [0,i-15], y[6, i-15])
            k2=fun(t+0.5*h, y0+0.5*h*k1, params, U1, U2, U3, beta, y
  [0,i-15], y[6, i-15])
            k3=fun(t+0.5*h, y0+0.5*h*k2, params, U1, U2, U3, beta, y
  [0,i-15], y[6, i-15])
            k4=fun(t+    h, y0+    h*k3, params, U1_1, U2_1, U3_1,
  beta_1, y[0,i-15], y[6, i-15])
            y[:,i+1] = y0 + h*(k1+2*(k2+k3)+k4)/6.0

        else:

            k1=fun(t      , y0          , params, U1, U2, U3, beta,
  states_15_start[0][i], states_15_start[1][i])
            k2=fun(t+0.5*h, y0+0.5*h*k1, params, U1, U2, U3, beta,
  states_15_start[0][i], states_15_start[1][i])
```

```
            k3=fun(t+0.5*h, y0+0.5*h*k2, params, U1, U2, U3, beta, states_15_start[0][i], states_15_start[1][i])
            k4=fun(t+    h, y0+    h*k3, params, U1_1, U2_1, U3_1, beta_1, states_15_start[0][i], states_15_start[1][i])
            y[:,i+1] = y0 + h*(k1+2*(k2+k3)+k4)/6.0

        U1 = U1_1
        U2 = U2_1
        U3 = U3_1
        beta = beta_1

    return y


# Function which returns tuple of controls and transmission rate at a specified time
def control_at_time(controls, t):

    if (isinstance(t,int)):
        t1 = t
    else:
        t1 = t.astype(int)

    return (controls[0][t1], controls[1][t1], controls[2][t1], controls[3][t1])


# Runge-Kutta 4 method applied for the costate (multipliers) problem
def solve_rk4_H(fun, t_span, y0, h, state, params, controls, multipliers, states_15_start):
    nsteps = int((t_span[1]-t_span[0])/h)
    y = np.zeros((len(y0),nsteps+1))
    y[:,nsteps] = y0
    U1, U2, U3, beta = control_at_time(controls, nsteps)

    for i in range(nsteps):

        t = t_span[0]+i*h

        y0 = y[:,nsteps - i]
        U1_1, U2_1, U3_1, beta_1 = control_at_time(controls, nsteps - i - 1)

        if t >= 15:

            k1=-fun(t      , params, state[:, nsteps - i], U1, U2, U3, beta, y0         , state[0, nsteps -i -15], state[6, nsteps- i -15])
            k2=-fun(t+0.5*h, params, state[:, nsteps - i],  U1, U2, U3, beta, y0+0.5*h*k1, state[0, nsteps -i -15], state[6, nsteps- i -15])
```

```
114             k3=-fun(t+0.5*h, params, state[:, nsteps - i],  U1, U2,
    U3, beta, y0+0.5*h*k2, state[0, nsteps -i -15], state[6, nsteps- i
     -15])
115             k4=-fun(t+    h, params, state[:, nsteps - i - 1],  U1_1,
     U2_1, U3_1, beta_1, y0+    h*k3, state[0, nsteps -i -15], state
    [6, nsteps- i -15])
116             y[:,nsteps - i - 1] = y0 + h*(k1+2*(k2+k3)+k4)/6.0
117
118         else:
119             k1=-fun(t      , params, state[:, nsteps - i], U1, U2, U3
    , beta, y0         , states_15_start[0][int(t)], states_15_start
    [1][int(t)])
120             k2=-fun(t+0.5*h, params, state[:, nsteps - i],  U1, U2,
    U3, beta, y0+0.5*h*k1, states_15_start[0][int(t)], states_15_start
    [1][int(t)])
121             k3=-fun(t+0.5*h, params, state[:, nsteps - i],  U1, U2,
    U3, beta, y0+0.5*h*k2, states_15_start[0][int(t)], states_15_start
    [1][int(t)])
122             k4=-fun(t+    h, params, state[:, nsteps - i - 1],  U1_1,
     U2_1, U3_1, beta_1, y0+    h*k3, states_15_start[0][int(t)],
    states_15_start[1][int(t)])
123             y[:,nsteps - i - 1] = y0 + h*(k1+2*(k2+k3)+k4)/6.0
124         U1 = U1_1
125         U2 = U2_1
126         U3 = U3_1
127         beta = beta_1
128
129     return y
130
131
132 # Definition of the Lagrangian function. Choose the Lagrangian to
    optimize by switching on/off returns (lines 149-152)
133 def Lagrangian(t, params, state, U1, U2, U3, beta):
134
135     alpha, gamma, muR, muV, sigma, omega, theta, f= params
136
137     N = np.sum(state)
138
139     S = state[0]
140     E = state[1]
141     I = state[2]
142     H = state[3]
143     R = state[4]
144     D = state[5]
145     V = state[6]
146     W = state[7]
147
148
149     #return 0.0 #For DFinal only
150     #return I**2
151     #return (beta * (S + sigma * V) * I / N  - alpha * E)**2 #E_dot
```

```
    return  I**2 + 1e4 * (beta * (S + sigma * V) * I / N  - alpha * E)**2 #I_squared + E_dot


# Definition of Hamiltonian Function
def Hamiltonian(t, params, state, U1, U2, U3, beta, multipliers, states_15_0, states_15_1):

    return Lagrangian(t, params, state, U1, U2, U3, beta) + jnp.dot(multipliers, SEIHRDVW(t, state, params, U1, U2, U3, beta, states_15_0, states_15_1))


# Function evaluating the cost functional to optimize
def cost_functional(params, state, controls, Tf):
    E = state[1,:]
    I = state[2,:]
    D = state[5,:]

    #cost_d = 1e3 * D[-1]**2 # Dfinal + infectious
    #cost_d = D[-1]**2 # Dfinal only
    cost_d = 0.0 # No final time control
    cost_i = 0.0

    for i in range(Tf + 1):
        U1, U2, U3, beta = control_at_time(controls, i)
        if i == 0 or i == Tf :
            cost_i += Lagrangian(i, params, state[:,i], U1, U2, U3, beta)/2
        else:
            cost_i += Lagrangian(i, params, state[:,i], U1, U2, U3, beta)
    cost = cost_i + cost_d

    return cost


# Function plotting states evolution
def plot(state, path):
    S = state[0]
    E = state[1]
    I = state[2]
    H = state[3]
    R = state[4]
    D = state[5]
    V = state[6]
    W = state[7]

    t = np.linspace(0, len(S), len(S))

    Sp, = plt.plot(S,'b', label = 'Susceptibles')
```

```
    Ep, = plt.plot(E,'y', label = 'Exposed')
    Ip, = plt.plot(I,'r', label = 'Infectives')
    Hp, = plt.plot(H, 'gold', label = 'Healing')
    Rp, = plt.plot(R,'g', label = 'Recovered')
    Dp, = plt.plot(D,'k', label = 'Deads')
    Vp, = plt.plot(V, 'c', label = 'Vaccinated')
    Wp, = plt.plot(W, 'm', label = 'Completely Vaccinated')

    plt.legend(handles = [Sp, Ep, Ip, Hp, Rp, Dp, Vp, Wp])
    plt.title('State evolution')
    plt.savefig(path + "state.png")
    plt.show()

    S2p, = plt.plot(S, 'b', label = 'Susceptibles')
    plt.title('Susceptibles')
    plt.savefig(path + "susceptibles.png")
    plt.show()

    I2p, = plt.plot(I, 'r', label = 'Infectives')
    plt.title('Infectives')
    plt.savefig(path + "infectives.png")
    plt.show()

    E2p, = plt.plot(E, 'g', label = 'Exposed')
    plt.title('Exposed')
    plt.savefig(path + "exposed.png")
    plt.show()

    H2p, = plt.plot(H, 'c', label = 'Healing')
    plt.title('Healing')
    plt.savefig(path + "healing.png")
    plt.show()

    R2p, = plt.plot(R, 'g', label = 'Recovered')
    plt.title('Recovered')
    plt.savefig(path + "recovered.png")
    plt.show()

    D2p, = plt.plot(D, 'k', label = 'Deceased')
    plt.title('Deceased')
    plt.savefig(path + "deceased.png")
    plt.show()

    V2p, = plt.plot(V, 'c', label = 'Vaccinated - first dose')
    plt.title('Vaccinated - first dose')
    plt.savefig(path + "vaccinated_V.png")
    plt.show()

    W2p, = plt.plot(W, 'm', label = 'Vaccinated - second dose')
    plt.title('Vaccinated - completed cycle')
    plt.savefig(path + "vaccinated_W.png")
```

```
248     plt.show()
249
250
251 # Function plotting multipliers evolution
252 def plot_multi(multipliers, path):
253     m_S = multipliers[0]
254     m_E = multipliers[1]
255     m_I = multipliers[2]
256     m_H = multipliers[3]
257     m_R = multipliers[4]
258     m_D = multipliers[5]
259     m_V = multipliers[6]
260     m_W = multipliers[7]
261
262     m_Sp, = plt.plot(m_S,'b', label = 'Susceptibles multiplier')
263     m_Ep, = plt.plot(m_E,'y', label = 'Exposed multiplier')
264     m_Ip, = plt.plot(m_I,'r', label = 'Infectives multiplier')
265     m_Hp, = plt.plot(m_H, 'gold', label = 'Healing multiplier')
266     m_Rp, = plt.plot(m_R,'g', label = 'Removed multiplier')
267     m_Dp, = plt.plot(m_D,'k', label = 'Deads multiplier')
268     m_Vp, = plt.plot(m_V, 'c', label = 'Vaccinated multiplier')
269     m_Wp, = plt.plot(m_W, 'm', label = 'Completely Vaccinated
    multiplier')
270
271     plt.legend(handles = [m_Sp, m_Ep, m_Ip, m_Hp, m_Rp, m_Dp, m_Vp,
    m_Wp])
272     plt.title('Multipliers evolution')
273     plt.savefig(path + "multipliers.png" )
274     plt.show()
275
276
277 # Function plotting controls evolution
278 def plot_controls(controls, vax_history, path, prima_history,
    seconda_history):
279
280     U1 = controls[0]
281     U2 = controls[1]
282     U3 = controls[2]
283     beta = controls[3]
284
285     U1p, = plt.plot(U1, label = 'First dose Susceptibles')
286     plt.title('Prima dose Susceptibles')
287     plt.savefig(path + "prima_dose_S.png")
288     plt.show()
289
290     U2p, = plt.plot(U2, label = 'First dose Recovered')
291     plt.title('Prima dose Recovered')
292     plt.savefig(path + "prima_dose_R.png")
293     plt.show()
294
295     U3p, = plt.plot(U3, label = 'Second doses')
```

```
296     secondaverap, = plt.plot(seconda_history, linestyle = 'dashed',
        label = 'Second doses - LOM')
297     plt.legend(handles = [U3p, secondaverap])
298     plt.title('Second doses')
299     plt.savefig(path + "seconda_dose.png")
300     plt.show()
301 
302     primadosep, = plt.plot(U1 + U2, label = 'First doses')
303     primadoseverap, = plt.plot(prima_history, linestyle = 'dashed',
        label = 'First doses - LOM')
304     plt.legend(handles = [primadosep, primadoseverap])
305     plt.title('First doses')
306     plt.savefig(path + "prime_dosi_comp.png")
307     plt.show()
308 
309     Usump, = plt.plot(U1 + U2 + U3, label = 'Total vaccinations')
310     vax_histp, = plt.plot(vax_history, label = 'Real Vaccinations')
311     plt.legend(handles = [Usump, vax_histp])
312     plt.title('Total Vaccinations')
313     plt.savefig(path + "vax_tot.png")
314     plt.show()
315 
316     betap, = plt.plot(beta, label = 'Beta')
317     plt.title('Beta')
318     plt.savefig(path + "beta.png")
319     plt.show()
320 
321     vaccination1Sp, = plt.plot(U1, 'm', label = 'First dose
        Susceptibles')
322     vaccination1Rp, = plt.plot(U2, 'y', label = 'First dose Recovered
        ')
323     vaccination2p, = plt.plot(U3, 'g', label = 'Second dose')
324 
325     plt.legend(handles = [vaccination1Sp, vaccination1Rp,
        vaccination2p])
326     plt.title('Vaccination summary')
327     plt.savefig(path + "vax_sum.png")
328     plt.show()
329 
330 
331 # Function plotting controls evolution without information about
        ripartition of first and second doses
332 def plot_controls_2(controls, vax_history, path):
333 
334     U1 = controls[0]
335     U2 = controls[1]
336     U3 = controls[2]
337     beta = controls[3]
338 
339     U1p, = plt.plot(U1, label = 'First dose Susceptibles')
340     plt.title('First dose Susceptibles')
```

```
    plt.savefig(path + "prima_dose_S.png")
    plt.show()

    U2p, = plt.plot(U2, label = 'First dose Recovered')
    plt.title('First dose Recovered')
    plt.savefig(path + "dose_R.png")
    plt.show()

    U3p, = plt.plot(U3, label = 'Second dose')
    plt.title('Second dose')
    plt.savefig(path + "seconda_dose.png")
    plt.show()

    Usump, = plt.plot(U1 + U2 + U3, label = 'Total vaccinations')
    vax_histp, = plt.plot(vax_history, label = 'Real vaccinations')

    plt.legend(handles = [Usump, vax_histp])
    plt.title('Total vaccinations')
    plt.savefig(path + "vax_tot.png")
    plt.show()

    betap, = plt.plot(beta, label = 'Transmission Rate')
    plt.title('Transmission Rate')
    plt.savefig(path + "beta.png")
    plt.show()

    vaccination1Sp, = plt.plot(U1, 'm', label = 'First dose
    Susceptibles')
    vaccination1Rp, = plt.plot(U2, 'y', label = 'First dose Recovered
    ')
    vaccination2p, = plt.plot(U3, 'g', label = 'Second dose')

    plt.legend(handles = [vaccination1Sp, vaccination1Rp,
    vaccination2p])
    plt.title('Vaccination summary')
    plt.savefig(path + "vax_sum.png")
    plt.show()


# Function plotting the reproduction number
def plot_Rt(state, controls, params, Tf, path):

    S = state[0]
    E = state[1]
    I = state[2]
    H = state[3]
    R = state[4]
    D = state[5]
    V = state[6]
    W = state[7]

```

```
    alpha, gamma, muR, muV, sigma, omega, theta, f= params

    t = np.linspace(0, len(S), len(S))

    N = 10103969
    Rt = np.zeros(Tf+1)

    gamma_2 = 1/9

    for i in range(Tf+1):

        Rt[i] = controls[3][i] / gamma * (S[i] + sigma * V[i]) / N

    Rtp, = plt.plot(Rt, 'm', label = 'Rt')
    plt.title('Rt')
    plt.savefig(path + "Rt.png")
    plt.show()
    np.savetxt(path + 'Rt.csv', Rt, delimiter = ',')


# Function plotting percentages of first, second doses and doses to recovered
def plot_histogram(controls, z1, Tf, path):
    prima_dose_perc = (controls[0] ) / z1
    dose_r_perc = controls[1] / z1
    seconda_dose_perc = controls[2] / z1
    prima_list = list(prima_dose_perc)
    rec_list = list(dose_r_perc)
    seconda_list = list(seconda_dose_perc)

    t = list(range(Tf+1))

    plt.bar(t, prima_list, color = 'm', width = 0.25, edgecolor = 'm', label = 'Percentage - first doses')
    plt.bar(t, seconda_list, color = 'c', width = 0.25, edgecolor = 'c', bottom = prima_dose_perc, label = 'Percentage - second doses')
    plt.bar(t, rec_list, color = 'g', width = 0.25, edgecolor = 'g', bottom = prima_dose_perc + seconda_dose_perc, label = 'Percentage - recovered doses')

    plt.legend(['Percentage of first doses', 'Percentage of second doses', 'Percentage of doses to recovered'])
    plt.title('Doses percentages')
    plt.savefig(path + "hist.png")
    plt.show()


# Checking validity of the constraint about maximum daily doses to administer
def check1(U1, U2, U3, U_ub): #CONSTRAINT 1
    return (U1 + U2 + U3 <= U_ub)
```

```
433
434 # Checking validity of the constraint about minimum elapsing time
    among doses
435 def check2(U1_vec, U2_vec, U3_vec): #CONSTRAINT 2
436     return (np.sum(U3_vec) <= np.sum(U1_vec))
437
438 # Checking validity of the constraint about maximum elapsing time
    among doses
439 def check3(U1_vec, U2_vec, U3_vec): #CONSTRAINT 3
440     return (np.sum(U3_vec) >= np.sum(U1_vec))
441
442
443 # Algorithm projecting vector v on the simplex \sum_{n=0}^{N} v_n = z
444 def projection_simplex_sort(v, z=1):
445
446     n_features = v.shape[0]
447     u = np.sort(v)[::-1]
448     cssv = np.cumsum(u) - z
449     ind = np.arange(n_features) + 1
450     cond = u - cssv / ind > 0
451
452     rho = ind[cond][-1]
453     theta = cssv[cond][-1] / float(rho)
454     w = np.maximum(v - theta, 0)
455     return w
456
457
458 # MULTI-PROJECTION ALGORITHM
459 def proj_complete(controls, z1, state, Tf): #U1[0] U2[0] U3[0] fixed
460
461     k = 30 #maximum number of iterations of the MP Algorithm
462
463     # 1st projection on the simplex with z = z1[j]
464     for j in range(Tf + 1):
465         if j >= 19 :
466             vec = np.array([controls[0][j], controls[1][j], controls
    [2][j]])
467             vec = projection_simplex_sort(vec, z1[j])
468             controls[0][j] = max(min(vec[0], state[0,j-1]),0)
469             controls[1][j] = max(min(vec[1], state[4,j-1]),0)
470             controls[2][j] = max(min(vec[2], state[6,j-1]),0)
471
472         else:
473             controls[2][j] =  0.0
474             vec = np.array([controls[0][j], controls[1][j]])
475             vec = projection_simplex_sort(vec, z1[j])
476             if j == 0 :
477                 controls[0][j] = max(min(vec[0], state[0,j]),0)
478                 controls[1][j] = max(min(vec[1], state[4,j]),0)
479             else:
480                 controls[0][j] = max(min(vec[0], state[0,j-1]),0)
```

```
481                 controls[1][j] = max(min(vec[1], state[4,j-1]),0)
482 
483     # MULTI-PROJECTION CYCLE
484     for i in range(0,k):
485 
486         #STEP1: Projection on maximum elapsing time constraint
487         for j in range(Tf+1):
488             if ((j>=42) & (np.sum(controls[0][:j-42+1] ) -  np.sum(
    controls[2][:j+1]) > 0)):
489 
490                 vec2 = controls[2][:j+1]
491                 vec2 = projection_simplex_sort(vec2, np.sum(controls
    [0][:j-42+1]))
492                 controls[2][:j+1] = np.minimum(vec2, z1[:j+1])
493                 for s in range(j+1):
494                     vec4 = np.array([controls[0][s], controls[1][s]])
495 
496                     if (z1[s] - controls[2][s] <= 0):
497                         vec4 = np.zeros(len(vec4))
498                     else:
499                         vec4 = projection_simplex_sort(vec4, max(z1[s
    ] - controls[2][s],0))
500                     if s == 0 :
501                         controls[0][s] = max(min(vec4[0], state[0,s])
    ,0)
502                         controls[1][s] = max(min(vec4[1], state[4,s])
    ,0)
503                     else:
504                         controls[0][s] = max(min(vec4[0], state[0,s
    -1]),0)
505                         controls[1][s] = max(min(vec4[1], state[4,s
    -1]),0)
506 
507         #STEP2: Projection on maximum daily administrations
    constraint
508         for j in range(Tf + 1):
509 
510             if j >= 19 :
511                 vec3 = np.array([controls[0][j], controls[1][j],
    controls[2][j]])
512                 vec3 = projection_simplex_sort(vec3, z1[j])
513                 controls[0][j] = max(min(vec3[0], state[0,j-1]),0)
514                 controls[1][j] = max(min(vec3[1], state[4,j-1]),0)
515                 controls[2][j] = max(min(vec3[2], state[6,j-1]),0)
516 
517 
518             else:
519                 controls[2][j] =  0.0
520                 vec3 = np.array([controls[0][j], controls[1][j]])
521                 vec3 = projection_simplex_sort(vec3, z1[j])
522                 if j == 0 :
```

```
                    controls[0][j] = max(min(vec3[0], state[0,j]),0)
                    controls[1][j] = max(min(vec3[1], state[4,j]),0)
                else:
                    controls[0][j] = max(min(vec3[0], state[0,j-1])
,0)
                    controls[1][j] = max(min(vec3[1], state[4,j-1])
,0)

        #STEP3: Projection on minimum elapsing time constraint
        for j in range(Tf+1):
            if ((j>=19) & (np.sum(controls[0][:j-19+1]) -  np.sum(
controls[2][:j+1]) < 0)):
                vec2 = controls[2][:j+1]
                vec2 = projection_simplex_sort(vec2, np.sum(controls
[0][:j-19+1]))
                controls[2][:j+1] = np.minimum(vec2, z1[:j+1])
                for s in range(j+1):
                    vec4 = np.array([controls[0][s], controls[1][s]])

                    if (z1[s] - controls[2][s] <= 0):
                        vec4 = np.zeros(len(vec4))
                    else:
                        vec4 = projection_simplex_sort(vec4, max(z1[s
] - controls[2][s],0))
                    if s == 0 :
                        controls[0][s] = max(min(vec4[0], state[0,s])
,0)
                        controls[1][s] = max(min(vec4[1], state[4,s])
,0)
                    else:
                        controls[0][s] = max(min(vec4[0], state[0,s
-1]),0)
                        controls[1][s] = max(min(vec4[1], state[4,s
-1]),0)

        #STEP2bis: Projection on maximum daily administrations
constraint
        for j in range(Tf + 1):

            if j >= 19 :
                vec3 = np.array([controls[0][j], controls[1][j],
controls[2][j]])
                vec3 = projection_simplex_sort(vec3, z1[j])
                controls[0][j] = max(min(vec3[0], state[0,j-1]),0)
                controls[1][j] = max(min(vec3[1], state[4,j-1]),0)
                controls[2][j] = max(min(vec3[2], state[6,j-1]),0)


            else:
                controls[2][j] =  0.0
                vec3 = np.array([controls[0][j], controls[1][j]])
```

```
                vec3 = projection_simplex_sort(vec3, z1[j])
                if j == 0 :
                    controls[0][j] = max(min(vec3[0], state[0,j]),0)
                    controls[1][j] = max(min(vec3[1], state[4,j]),0)
                else:
                    controls[0][j] = max(min(vec3[0], state[0,j-1])
    ,0)
                    controls[1][j] = max(min(vec3[1], state[4,j-1])
    ,0)

    return controls


def main(argv):
    Tf = int(argv[0])

    dt = 1
    print('Initializing...')
    print('
    ------------------------------------------------------------')

    # alpha: inverse of the incubation time
    # gamma: infectous spassing rate
    # muR: re-infection rate from recovered class
    # muV: re-infection rate from vaccinated class
    # sigma: fraction of vaccine effectiveness on infection
    trasmission
    # omega: hospedalizing rate
    # theta: fraction for accounting for vaccine effectiveness on
    effects reduction
    # f: fatality rate


    #vax_history = np.array(75e3 * np.ones(Tf + 1))
    #vaccine_dataframe = pd.read_csv('vaccini_grouped_lombardia.csv',
    sep =',')
    #vax_history = vaccine_dataframe['totale'].to_numpy()

    #vax_history = np.array(50e3 * np.ones(Tf + 1))

    #vax_history = np.array(35e3 * np.ones(Tf+1))
    #vax_history[0:60] = 10e3 * np.ones(60)
    #vax_history[60:101] = 25e3 * np.ones(41)

    # Definition of daily amount of doses
    vaccine_dataframe = pd.read_csv('
    vaccini_grouped_lom_ALL_29_maggio.csv', sep =',')
    vax_history = vaccine_dataframe['totale'].to_numpy()
    prima_history = vaccine_dataframe['prima_dose'].to_numpy()
    seconda_history = vaccine_dataframe['seconda_dose'].to_numpy()

```

```
607
608     # Rt* computation
609     Infectives_dataframe = pd.read_csv('Lombardia_infetti_147.csv',
    sep = ',')
610     Suscettibili_147 = Infectives_dataframe['suscettibili'].to_numpy
    ()
611     Vaccinated_first_147 = Infectives_dataframe['vaccinati_prima_dose
    '].to_numpy()
612     Infectives_147 = Infectives_dataframe['totale_positivi'].to_numpy
    ()
613     Deceased_147 = Infectives_dataframe['deceduti'].to_numpy()
614
615     gamma_2 = 1/9
616     logI=np.log(Infectives_147)
617     Rt=(logI[7:]-logI[:-7])/7/gamma_2+1
618
619
620     # Parameters definition
621     params = [0.182, 0.211, 4.76e-4, 4.76e-4, 0.25, 0.0690, 0.15,
    2.794e-3] # LOMBARDIA
622     N = 10103969
623
624
625     # \beta definition
626     beta = Rt * params[1] * N /(Suscettibili_147[:-7] + params[4] *
    Vaccinated_first_147[:-7])
627
628
629     # History of Susceptibles and Vaccinationed with first dose 15
    days before the beginning of the simulation (to use in the
    SEIHRDVW computation)
630
631     states_15_start = list()
632
633     #susc_dataframe = pd.read_csv('', sep = ',')
634     #susc_15 = susc_dataframe.to_numpy()
635     #states_15_start.append(susc_15)
636     #states_15_start.append(9620987 * np.ones(15))
637
638     states_15_start.append(np.array([9651234, 9648490, 9646546,
    9644751, 9643801, 9641523, 9639370, 9636714, 9634086, 9632480,
    9632014, 9631441, 9630598, 9628931, 9624866]))
639     states_15_start.append(np.zeros(15))
640
641     states_15_start[1][-1] = 714
642     states_15_start[1][-2] = 114
643     states_15_start[1][-3] = 104
644     states_15_start[1][-4] = 107
645     states_15_start[1][-5] = 717
646
647
```

```
    # INITIAL CONDITIONS
    #IC = [57000000, 10800, 569896, 1463111, 74159,0,2000000,0]
    #IC = [57000000, 10800, 569896, 1463111, 74159,0,4e5,0]
    #IC = [48000000, 10680, 563479, 28949, 2933757, 109847, 10324127,
    3237582] # 1st April 2021
    IC = [9620987, 2.5e4, 5.5e4, 50520, 4.01e5, 25203, 1779, 0]
    #IC = [48000000, 10680, 563479, 28949, 2933757, 109847, 0, 0]


    # META ALGORITHM: STEP 1 -> Define initial Guess and parameters
    # Define initial guess for vaccinations
    controls = list()

    #controls.append(25e3 * np.ones(Tf+1))
    #controls.append(5e5 * np.ones(Tf+1))
    #controls.append(25e3 * np.ones(Tf+1))
    #controls.append(5e5 * np.ones(Tf+1))

    controls.append(np.zeros(Tf+1))
    controls.append(np.zeros(Tf+1))
    controls.append(np.zeros(Tf+1))

    #controls.append(np.zeros(Tf+1))
    #controls.append(0.1007 * np.ones(int(Tf/7)+1))
    #controls.append(np.array([0.23225, 0.23225, 0.23225, 0.23225,
    0.23225, 0.26315, 0.26315, 0.26315, 0.26315, 0.26172, 0.26172,
    0.26172, 0.26172, 0.26172, 0.26172, 0.19872]))
    #controls.append(0.26172 * np.ones(int(Tf/7)+1))

    #T_f = 147 only
    #controls.append(np.array([0.23225, 0.23225, 0.23225, 0.23225,
    0.23225, 0.26315, 0.26315, 0.26315, 0.26315, 0.26172, 0.26172,
    0.26172, 0.26172, 0.26172, 0.26172, 0.19872, 0.19872, 0.19872,
    0.19872, 0.19872, 0.19872, 0.19872]))

    controls.append(beta)


    # Path for saving data and figures
    path = "/home/giovanni/Desktop/OC_LOM/FINAL_LOM/15_GIUGNO/
    I_squared_2/"

    # Maximum steps and tolerance for the PGD Algorithm
    nsteps = 100 #100#20
    tolerance = 1e-6

    # Step length PGD Algorithm
    #learning_rate_vax = 1e-5
    learning_rate_vax = 1e-2
    #learning_rate_vax = 1e-5

```

```
    ep = 1
    err_min = tolerance + 1
    err_old = 1e-7
    state = np.zeros((len(IC),Tf+1))
    multipliers = np.zeros((len(IC),Tf+1))

    #vaccine_dataframe = pd.read_csv('vaccini_grouped_lombardia.csv', sep =',')
    #vax_history = vaccine_dataframe['totale'].to_numpy()

    # Functions for gradients of the Hamiltonian
    grad_x = jax.grad(Hamiltonian, 2)
    grad_u1 = jax.grad(Hamiltonian, 3)
    grad_u2 = jax.grad(Hamiltonian, 4)
    grad_u3 = jax.grad(Hamiltonian, 5)

    # Just in time compilation Gradients
    grad_x_jit = jax.jit(grad_x)
    grad_u1_jit = jax.jit(grad_u1)
    grad_u2_jit = jax.jit(grad_u2)
    grad_u3_jit = jax.jit(grad_u3)

    sum_gradient = 0.0
    sum_old = 1.0
    history = list()

    print('Gradient Loop...')
    print('-----------------------------------------------------------------')

    while (ep <= nsteps and err_min > tolerance):

        #Final time control only
        #lambda_fin = [0 , 0, 0, 0, 0,  state[5,Tf], 0,0] #1e3 *

        lambda_fin = [0 , 0, 0, 0, 0, 0, 0,0]
        sum_gradient = 0.0

        # META ALGORITHM: STEP 2a -> Solution of the state problem
        state = solve_rk4(SEIHRDVW, [0, Tf], IC, 1.0, params, controls, states_15_start)
        print('OK STATE')


        # META ALGORITHM: STEP 2b -> Solution of the costate problem
        multipliers = solve_rk4_H(grad_x_jit, [Tf, 0], lambda_fin, -1.0, state, params, controls, multipliers, states_15_start)
        print('OK MULTI')
```

```
738         # META ALGORITHM: STEPS 2c -> Computation of the gradient,
    Gradient Descent step and Multi-Projection
739         for j in range(Tf+1):
740             U1, U2, U3, beta = control_at_time(controls, j)
741             grad_u1_vec = 0.0
742             grad_u2_vec = 0.0
743             grad_u3_vec = 0.0
744 
745             # Computation of the gradients
746             if j >= 15:
747                 grad_u1_vec = grad_u1_jit(j, params, state[:,j], U1,
    U2, U3, beta, multipliers[:, j], state[0, j-15], state[6, j-15])
748                 grad_u2_vec = grad_u2_jit(j, params, state[:,j], U1,
    U2, U3, beta, multipliers[:, j], state[0, j-15], state[6, j-15])
749                 grad_u3_vec = grad_u3_jit(j, params, state[:,j], U1,
    U2, U3, beta, multipliers[:, j], state[0, j-15], state[6, j-15])
750             else:
751                 grad_u1_vec = grad_u1_jit(j, params, state[:,j], U1,
    U2, U3, beta, multipliers[:, j], states_15_start[0][j],
    states_15_start[1][j])
752                 grad_u2_vec = grad_u2_jit(j, params, state[:,j], U1,
    U2, U3, beta, multipliers[:, j], states_15_start[0][j],
    states_15_start[1][j])
753                 grad_u3_vec = grad_u3_jit(j, params, state[:,j], U1,
    U2, U3, beta, multipliers[:, j], states_15_start[0][j],
    states_15_start[1][j])
754 
755 
756             # Gradient Descent step
757             controls[0][j] -=  learning_rate_vax * (1 - ep / nsteps)
    * grad_u1_vec #U1
758             controls[1][j] -=  learning_rate_vax * (1 - ep / nsteps)
    * grad_u2_vec #U2
759             controls[2][j] -=  learning_rate_vax * (1 - ep / nsteps)
    * grad_u3_vec #U3
760 
761             sum_gradient += abs(grad_u1_vec) + abs(grad_u2_vec) + abs
    (grad_u3_vec)
762 
763         # Multi-Projection of the control variables
764         controls = proj_complete(controls, vax_history, state, Tf)
765 
766 
767         print('OK GRAD')
768         print('Gradient: ', sum_gradient)
769 
770         # Saving cost functional values
771         history.append(cost_functional(params, state, controls, Tf))
772 
773         print('Cost Functional: ', cost_functional(params, state,
    controls, Tf))
```

```
        print('Iteration', ep, 'Done')
        print('Error: ',err_min)

        # META ALGORITHM: STEP 2d -> Stopping criterium verification
        err_min = abs(sum_old - sum_gradient)/abs(sum_old)
        sum_old = sum_gradient

        ep += 1

        print('First check: ', ep <= nsteps)
        print('Second check: ', err_min > tolerance)

        print('
    ------------------------------------------------------------')

    print('End Loop...')
    print('
    ------------------------------------------------------------')

    #Checking constraints are fulfilled
    for j in range(Tf+1):
        if not check1(controls[0][j], controls[1][j], controls[2][j],
    vax_history[j]):
            print('Check 1 fails at time', j)
        if j >= 19:
            if not check2(controls[0][:j+1-19], controls[1][:j+1-19],
    controls[2][:j+1]):
                print('Check 2 fails at time', j)
        if j >= 42:
            if not check3(controls[0][:j+1-42], controls[1][:j+1-42],
    controls[2][:j+1]):
                print('Check 3 fails at time', j)


    # Deleting days without imposed constraints
    state_fin = state[:,:-42]
    multipliers_fin = multipliers[:,:-42]
    controls_fin = list()
    controls_fin.append(controls[0][:-42])
    controls_fin.append(controls[1][:-42])
    controls_fin.append(controls[2][:-42])
    controls_fin.append(controls[3][:-42])


    # Plots

    plot(state_fin, path)

    plot_multi(multipliers_fin, path)

```

```
819     plot_controls(controls_fin, vax_history, path, prima_history,
        seconda_history, Tf - 42)
820
821     plot_Rt(state_fin, controls_fin, params, Tf - 42, path)
822
823     plot_histogram(controls_fin, Tf - 42, path)
824
825     # Saving procedure
826     np.savetxt(path + 'state.csv', state_fin, delimiter = ',')
827     np.savetxt(path + 'S.csv', state_fin[0,:], delimiter = ',')
828     np.savetxt(path + 'E.csv', state_fin[1,:], delimiter = ',')
829     np.savetxt(path + 'I.csv', state_fin[2,:], delimiter = ',')
830     np.savetxt(path + 'H.csv', state_fin[3,:], delimiter = ',')
831     np.savetxt(path + 'R.csv', state_fin[4,:], delimiter = ',')
832     np.savetxt(path + 'D.csv', state_fin[5,:], delimiter = ',')
833     np.savetxt(path + 'V.csv', state_fin[6,:], delimiter = ',')
834     np.savetxt(path + 'W.csv', state_fin[7,:], delimiter = ',')
835
836
837     np.savetxt(path + 'prima_dose_S.csv', controls_fin[0], delimiter
        = ',')
838     np.savetxt(path + 'dose_R.csv', controls_fin[1], delimiter = ',')
839     np.savetxt(path + 'seconda_dose.csv', controls_fin[2], delimiter
        = ',')
840     np.savetxt(path + 'beta.csv', controls_fin[3], delimiter = ',')
841
842     plt.loglog(history)
843     plt.title('Cost history')
844     plt.savefig(path + "cost_hist.png")
845     plt.show()
846
847     return 0
848
849 if __name__ == "__main__":
850     main(sys.argv[1:])
```

The following `python3` code refers to the solution of Optimal Control Problem 2 (with `SEIHRDVW2` model as state problem), where we fixed the weekly amount of delivered doses and the maximum daily administration capacity.

```
import jax
import os
os.environ["XLA_FLAGS"]="--xla_gpu_cuda_data_dir=/usr/lib/cuda"
os.environ["CUDA_HOME"]="/usr/lib/cuda"
import sys
import jax.numpy as jnp
import numpy as np
import matplotlib.pyplot as plt
import math
import pandas as pd

#SEIHRDVW evolution function
def SEIHRDVW(t, state, params, U1, U2, U3, beta, state_15_0,
    state_15_1):

  # alpha: inverse of the incubation time
  # gamma: infectIous spassing rate
  # muR: re-infection rate from recovered class
  # muV: re-infection rate from vaccinated class
  # sigma: fraction of vaccine effectiveness on infection trasmission
  # omega: hospedalizing rate
  # theta: fraction for accounting for vaccine effectiveness on
    effects reduction
  # f: fatality rate
  # state_15:  contains S and V 15 days before

  alpha, gamma, muR, muV, sigma, omega, theta, f= params

  N = np.sum(state)
  S = state[0]
  E = state[1]
  I = state[2]
  H = state[3]
  R = state[4]
  D = state[5]
  V = state[6]
  W = state[7]

  dSdt = - beta * S * I / N - U1 + muR * R + muV * V
  dEdt = beta * (S + sigma * V) * I / N  - alpha * E
  dIdt = alpha * E - gamma * I
  dHdt = gamma * I - omega * H
  dRdt = (1.0 - f *  (state_15_0 + theta * sigma * state_15_1) / (
    state_15_0 + sigma * state_15_1)) * omega * H - muR * R - U2
  dDdt = f * (state_15_0 + theta * sigma * state_15_1) / (state_15_0
    + sigma * state_15_1) * omega * H
```

```
  dVdt = - sigma * beta * V * I / N + U1 - muV * V - U3
  dWdt = U2 + U3

  return jnp.array([dSdt, dEdt, dIdt, dHdt, dRdt, dDdt, dVdt, dWdt])


# Runge-Kutta 4 method for the state SEIHRDVW problem
def solve_rk4(fun, t_span, y0, h, params, controls, states_15_start):
    nsteps = int((t_span[1]-t_span[0])/h)
    y = np.zeros((len(y0),nsteps+1))
    y[:,0] = y0
    U1, U2, U3, beta = control_at_time(controls, 0)

    for i in range(nsteps):
        t = int(t_span[0]+i*h)
        U1_1, U2_1, U3_1, beta_1 = control_at_time(controls, t+1)
        y0 = y[:,i]

        if i >= 15:

            k1=fun(t      , y0         , params, U1, U2, U3, beta, y
    [0,i-15], y[6, i-15])
            k2=fun(t+0.5*h, y0+0.5*h*k1, params, U1, U2, U3, beta, y
    [0,i-15], y[6, i-15])
            k3=fun(t+0.5*h, y0+0.5*h*k2, params, U1, U2, U3, beta, y
    [0,i-15], y[6, i-15])
            k4=fun(t+    h, y0+    h*k3, params, U1_1, U2_1, U3_1,
    beta_1, y[0,i-15], y[6, i-15])
            y[:,i+1] = y0 + h*(k1+2*(k2+k3)+k4)/6.0

        else:

            k1=fun(t      , y0         , params, U1, U2, U3, beta,
    states_15_start[0][i], states_15_start[1][i])
            k2=fun(t+0.5*h, y0+0.5*h*k1, params, U1, U2, U3, beta,
    states_15_start[0][i], states_15_start[1][i])
            k3=fun(t+0.5*h, y0+0.5*h*k2, params, U1, U2, U3, beta,
    states_15_start[0][i], states_15_start[1][i])
            k4=fun(t+    h, y0+    h*k3, params, U1_1, U2_1, U3_1,
    beta_1, states_15_start[0][i], states_15_start[1][i])
            y[:,i+1] = y0 + h*(k1+2*(k2+k3)+k4)/6.0

        U1 = U1_1
        U2 = U2_1
        U3 = U3_1
        beta = beta_1

    return y


# Function which returns tuple of controls and transmission rate at a
```

```
    specified time
def control_at_time(controls, t):

    if (isinstance(t,int)):
        t1 = t
    else:
        t1 = t.astype(int)

    return (controls[0][t1], controls[1][t1], controls[2][t1],
    controls[3][t1])


# Runge-Kutta 4 method applied for the costate (multipliers) problem
def solve_rk4_H(fun, t_span, y0, h, state, params, controls,
    multipliers, states_15_start):
    nsteps = int((t_span[1]-t_span[0])/h)
    y = np.zeros((len(y0),nsteps+1))
    y[:,nsteps] = y0
    U1, U2, U3, beta = control_at_time(controls, nsteps)

    for i in range(nsteps):

        t = t_span[0]+i*h

        y0 = y[:,nsteps - i]
        U1_1, U2_1, U3_1, beta_1 = control_at_time(controls, nsteps -
    i - 1)

        if t >= 15:

            k1=-fun(t      , params, state[:, nsteps - i], U1, U2, U3
    , beta, y0         , state[0, nsteps -i -15], state[6, nsteps- i
    -15])
            k2=-fun(t+0.5*h, params, state[:, nsteps - i],  U1, U2,
    U3, beta, y0+0.5*h*k1, state[0, nsteps -i -15], state[6, nsteps- i
     -15])
            k3=-fun(t+0.5*h, params, state[:, nsteps - i],  U1, U2,
    U3, beta, y0+0.5*h*k2, state[0, nsteps -i -15], state[6, nsteps- i
     -15])
            k4=-fun(t+    h, params, state[:, nsteps - i - 1],  U1_1,
     U2_1, U3_1, beta_1, y0+    h*k3, state[0, nsteps -i -15], state
    [6, nsteps- i -15])
            y[:,nsteps - i - 1] = y0 + h*(k1+2*(k2+k3)+k4)/6.0

        else:
            k1=-fun(t      , params, state[:, nsteps - i], U1, U2, U3
    , beta, y0         , states_15_start[0][int(t)], states_15_start
    [1][int(t)])
            k2=-fun(t+0.5*h, params, state[:, nsteps - i],  U1, U2,
    U3, beta, y0+0.5*h*k1, states_15_start[0][int(t)], states_15_start
    [1][int(t)])
```

```
            k3=-fun(t+0.5*h, params, state[:, nsteps - i],  U1, U2, U3, beta, y0+0.5*h*k2, states_15_start[0][int(t)], states_15_start[1][int(t)])
            k4=-fun(t+    h, params, state[:, nsteps - i - 1],  U1_1, U2_1, U3_1, beta_1, y0+    h*k3, states_15_start[0][int(t)], states_15_start[1][int(t)])
            y[:,nsteps - i - 1] = y0 + h*(k1+2*(k2+k3)+k4)/6.0
        U1 = U1_1
        U2 = U2_1
        U3 = U3_1
        beta = beta_1

    return y


# Definition of the Lagrangian function. Choose the Lagrangian to optimize by switching on/off returns (lines 149-152)
def Lagrangian(t, params, state, U1, U2, U3, beta):

    alpha, gamma, muR, muV, sigma, omega, theta, f= params

    N = np.sum(state)

    S = state[0]
    E = state[1]
    I = state[2]
    H = state[3]
    R = state[4]
    D = state[5]
    V = state[6]
    W = state[7]


    #return 0.0 #For DFinal only
    #return I**2
    #return (beta * (S + sigma * V) * I / N  - alpha * E)**2 #E_dot
    return  I**2 + 1e4 * (beta * (S + sigma * V) * I / N  - alpha * E)**2 #I_squared + E_dot


# Definition of Hamiltonian Function
def Hamiltonian(t, params, state, U1, U2, U3, beta, multipliers, states_15_0, states_15_1):

    return Lagrangian(t, params, state, U1, U2, U3, beta) + jnp.dot(multipliers, SEIHRDVW(t, state, params, U1, U2, U3, beta, states_15_0, states_15_1))


# Function evaluating the cost functional to optimize
def cost_functional(params, state, controls, Tf):
```

```
    E = state[1,:]
    I = state[2,:]
    D = state[5,:]

    #cost_d = 1e3 * D[-1]**2 # Dfinal + infectious
    #cost_d = D[-1]**2 # Dfinal only
    cost_d = 0.0 # No final time control
    cost_i = 0.0

    for i in range(Tf + 1):
        U1, U2, U3, Rt = control_at_time(controls, i)
        if i == 0 or i == Tf :
            cost_i += Lagrangian(i, params, state[:,i], U1, U2, U3,
    Rt)/2
        else:
            cost_i += Lagrangian(i, params, state[:,i], U1, U2, U3,
    Rt)
    cost = cost_i + cost_d

    return cost


# Function plotting states evolution
def plot(state, path):
    S = state[0]
    E = state[1]
    I = state[2]
    H = state[3]
    R = state[4]
    D = state[5]
    V = state[6]
    W = state[7]

    t = np.linspace(0, len(S), len(S))

    Sp, = plt.plot(S,'b', label = 'Susceptibles')
    Ep, = plt.plot(E,'y', label = 'Exposed')
    Ip, = plt.plot(I,'r', label = 'Infectives')
    Hp, = plt.plot(H, 'gold', label = 'Healing')
    Rp, = plt.plot(R,'g', label = 'Recovered')
    Dp, = plt.plot(D,'k', label = 'Deads')
    Vp, = plt.plot(V, 'c', label = 'Vaccinated')
    Wp, = plt.plot(W, 'm', label = 'Completely Vaccinated')

    plt.legend(handles = [Sp, Ep, Ip, Hp, Rp, Dp, Vp, Wp])
    plt.title('State evolution')
    plt.savefig(path + "state.png")
    plt.show()

    S2p, = plt.plot(S, 'b', label = 'Susceptibles')
    plt.title('Susceptibles')
```

```
    plt.savefig(path + "susceptibles.png")
    plt.show()

    I2p, = plt.plot(I, 'r', label = 'Infectives')
    plt.title('Infectives')
    plt.savefig(path + "infectives.png")
    plt.show()

    E2p, = plt.plot(E, 'g', label = 'Exposed')
    plt.title('Exposed')
    plt.savefig(path + "exposed.png")
    plt.show()

    H2p, = plt.plot(H, 'c', label = 'Healing')
    plt.title('Healing')
    plt.savefig(path + "healing.png")
    plt.show()

    R2p, = plt.plot(R, 'g', label = 'Recovered')
    plt.title('Recovered')
    plt.savefig(path + "recovered.png")
    plt.show()

    D2p, = plt.plot(D, 'k', label = 'Deceased')
    plt.title('Deceased')
    plt.savefig(path + "deceased.png")
    plt.show()

    V2p, = plt.plot(V, 'c', label = 'Vaccinated - first dose')
    plt.title('Vaccinated - first dose')
    plt.savefig(path + "vaccinated_V.png")
    plt.show()

    W2p, = plt.plot(W, 'm', label = 'Vaccinated - second dose')
    plt.title('Vaccinated - completed cycle')
    plt.savefig(path + "vaccinated_W.png")
    plt.show()


# Function plotting multipliers evolution
def plot_multi(multipliers, path):
    m_S = multipliers[0]
    m_E = multipliers[1]
    m_I = multipliers[2]
    m_H = multipliers[3]
    m_R = multipliers[4]
    m_D = multipliers[5]
    m_V = multipliers[6]
    m_W = multipliers[7]

    m_Sp, = plt.plot(m_S,'b', label = 'Susceptibles multiplier')
```

```
263     m_Ep, = plt.plot(m_E,'y', label = 'Exposed multiplier')
264     m_Ip, = plt.plot(m_I,'r', label = 'Infectives multiplier')
265     m_Hp, = plt.plot(m_H, 'gold', label = 'Healing multiplier')
266     m_Rp, = plt.plot(m_R,'g', label = 'Removed multiplier')
267     m_Dp, = plt.plot(m_D,'k', label = 'Deads multiplier')
268     m_Vp, = plt.plot(m_V, 'c', label = 'Vaccinated multiplier')
269     m_Wp, = plt.plot(m_W, 'm', label = 'Completely Vaccinated
    multiplier')
270
271     plt.legend(handles = [m_Sp, m_Ep, m_Ip, m_Hp, m_Rp, m_Dp, m_Vp,
    m_Wp])
272     plt.title('Multipliers evolution')
273     plt.savefig(path + "multipliers.png" )
274     plt.show()
275
276
277 # Function plotting controls evolution
278 def plot_controls(controls, vax_history, consegne, path,
    prima_history, seconda_history, Tf):
279
280     U1 = controls[0]
281     U2 = controls[1]
282     U3 = controls[2]
283     beta = controls[3]
284
285     U1p, = plt.plot(U1, label = 'First dose Susceptibles')
286     plt.title('Prima dose Susceptibles')
287     plt.savefig(path + "prima_dose_S.png")
288     plt.show()
289
290     U2p, = plt.plot(U2, label = 'First dose Recovered')
291     plt.title('Prima dose Recovered')
292     plt.savefig(path + "prima_dose_R.png")
293     plt.show()
294
295     U3p, = plt.plot(U3, label = 'Second dose - OUTPUT')
296     secondaverap, = plt.plot(seconda_history, linestyle = 'dashed',
    label = 'Second dose - REAL')
297     plt.legend(handles = [U3p, secondaverap])
298     plt.title('Second dose - Comparison')
299     plt.savefig(path + "seconda_dose.png")
300     plt.show()
301
302     primadosep, = plt.plot(U1 + U2, label = 'First dose - OUTPUT')
303     primadoseverap, = plt.plot(prima_history, linestyle = 'dashed',
    label = 'First dose - REAL')
304     plt.legend(handles = [primadosep, primadoseverap])
305     plt.title('First dose - Comparison')
306     plt.savefig(path + "prime_dosi_comp.png")
307     plt.show()
308
```

```
    Usump, = plt.plot(U1 + U2 + U3, label = 'Total vaccinations')
    vax_histp, = plt.plot(vax_history, label = 'Real Vaccinations')
    plt.legend(handles = [Usump, vax_histp])
    plt.title('Total Vaccinations')
    plt.savefig(path + "vax_tot.png")
    plt.show()

    consegne_vec = np.zeros(Tf +1)
    for i in range(len(consegne)):
        consegne_vec[7*i] = consegne[i]
    cumsumVaxp, = plt.plot(np.cumsum(U1 + U2 + U3), label = '
    Cumulative sum optimal vaccinations')
    cumsumconsegnep, = plt.plot(np.cumsum(consegne_vec), label = '
    Cumulative sum delivered doses')
    plt.legend(handles = [cumsumVaxp, cumsumconsegnep])
    plt.title('Cumulative doses administrated')
    plt.savefig(path + "cumulative_administration.png")
    plt.show()

    betap, = plt.plot(beta, label = 'Beta')
    plt.title('Beta')
    plt.savefig(path + "beta.png")
    plt.show()

    vaccination1Sp, = plt.plot(U1, 'm', label = 'First dose
    Susceptibles')
    vaccination1Rp, = plt.plot(U2, 'y', label = 'First dose Recovered
    ')
    vaccination2p, = plt.plot(U3, 'g', label = 'Second dose')

    plt.legend(handles = [vaccination1Sp, vaccination1Rp,
    vaccination2p])
    plt.title('Vaccination summary')
    plt.savefig(path + "vax_sum.png")
    plt.show()


# Function plotting controls evolution without information about
    ripartition of first and second doses
def plot_controls_2(controls, vax_history, path):

    U1 = controls[0]
    U2 = controls[1]
    U3 = controls[2]
    beta = controls[3]

    U1p, = plt.plot(U1, label = 'First dose Susceptibles')
    plt.title('First dose Susceptibles')
    plt.savefig(path + "prima_dose_S.png")
    plt.show()

```

```
    U2p, = plt.plot(U2, label = 'First dose Recovered')
    plt.title('First dose Recovered')
    plt.savefig(path + "dose_R.png")
    plt.show()

    U3p, = plt.plot(U3, label = 'Second dose')
    plt.title('Second dose')
    plt.savefig(path + "seconda_dose.png")
    plt.show()

    Usump, = plt.plot(U1 + U2 + U3, label = 'Total vaccinations')
    vax_histp, = plt.plot(vax_history, label = 'Real vaccinations')

    plt.legend(handles = [Usump, vax_histp])
    plt.title('Total vaccinations')
    plt.savefig(path + "vax_tot.png")
    plt.show()

    betap, = plt.plot(beta, label = 'Transmission Rate')
    plt.title('Transmission Rate')
    plt.savefig(path + "beta.png")
    plt.show()

    vaccination1Sp, = plt.plot(U1, 'm', label = 'First dose
    Susceptibles')
    vaccination1Rp, = plt.plot(U2, 'y', label = 'First dose Recovered
    ')
    vaccination2p, = plt.plot(U3, 'g', label = 'Second dose')

    plt.legend(handles = [vaccination1Sp, vaccination1Rp,
    vaccination2p])
    plt.title('Vaccination summary')
    plt.savefig(path + "vax_sum.png")
    plt.show()


# Function plotting the reproduction number
def plot_Rt(state, controls, params, Tf, path):

    S = state[0]
    E = state[1]
    I = state[2]
    H = state[3]
    R = state[4]
    D = state[5]
    V = state[6]
    W = state[7]

    alpha, gamma, muR, muV, sigma, omega, theta, f= params

    t = np.linspace(0, len(S), len(S))
```

```
402
403     N = 10103969
404     Rt = np.zeros(Tf+1)
405
406     gamma_2 = 1/9
407
408     for i in range(Tf+1):
409
410         Rt[i] = controls[3][i] / gamma * (S[i] + sigma * V[i]) / N
411
412     Rtp, = plt.plot(Rt, 'm', label = 'Rt')
413     plt.title('Rt')
414     plt.savefig(path + "Rt.png")
415     plt.show()
416     np.savetxt(path + 'Rt.csv', Rt, delimiter = ',')
417
418
419 # Function plotting percentages of first, second doses and doses to
    recovered
420 def plot_histogram(controls, Tf, path):
421     z1 = controls[0] + controls[1] + controls[2]
422     zero_adm = np.zeros(Tf+1)
423     zero_adm[z1 == 0] = 1
424     prima_dose_perc = (controls[0] ) / z1
425     dose_r_perc = controls[1] / z1
426     seconda_dose_perc = controls[2] / z1
427     prima_list = list(prima_dose_perc)
428     rec_list = list(dose_r_perc)
429     seconda_list = list(seconda_dose_perc)
430
431     t = list(range(Tf+1))
432
433     plt.bar(t, prima_list, color = 'm', width = 0.25, edgecolor = 'm'
    , label = 'Percentage - first doses')
434     plt.bar(t, seconda_list, color = 'c', width = 0.25, edgecolor = '
    c', bottom = prima_dose_perc, label = 'Percentage - second doses')
435     plt.bar(t, rec_list, color = 'g', width = 0.25, edgecolor = 'g',
    bottom = prima_dose_perc + seconda_dose_perc, label = 'Percentage
    - recovered doses')
436     plt.bar(t, zero_adm, color = 'red', width = 0.25, edgecolor = '
    red')
437
438     plt.legend(['Percentage of first doses', 'Percentage of second
    doses', 'Percentage of doses to recovered'])
439     plt.title('Doses percentages')
440     plt.savefig(path + "hist.png")
441     plt.show()
442
443
444 # Checking validity of the constraint about maximum daily doses to
    administer weekly
```

```
445 def check1(U1_vec, U2_vec, U3_vec, U_ub): #CONSTRAINT 1
446     return (np.sum(U1_vec + U2_vec + U3_vec) <= U_ub)
447 
448 # Checking validity of the constraint about minimum elapsing time
    among doses
449 def check2(U1_vec, U2_vec, U3_vec): #CONSTRAINT 2
450     return (np.sum(U3_vec) <= np.sum(U1_vec))
451 
452 # Checking validity of the constraint about maximum elapsing time
    among doses
453 def check3(U1_vec, U2_vec, U3_vec): #CONSTRAINT 3
454     return (np.sum(U3_vec) >= np.sum(U1_vec))
455 
456 
457 # Algorithm projecting vector v on the simplex \sum_{n=0}^{N} v_n = z
458 def projection_simplex_sort(v, z=1):
459 
460     n_features = v.shape[0]
461     u = np.sort(v)[::-1]
462 
463     cssv = np.cumsum(u) - z
464     ind = np.arange(n_features) + 1
465 
466     cond = u - cssv / ind > 0
467 
468     rho = ind[cond][-1]
469     theta = cssv[cond][-1] / float(rho)
470     w = np.maximum(v - theta, 0)
471     return w
472 
473 
474 # MULTI-PROJECTION ALGORITHM
475 def proj_complete(controls, z1, vax_max, state, Tf): #U1[0] U2[0] U3
    [0] fixed
476 
477     k = 30 #maximum number of iterations of the MP Algorithm
478 
479     # 1st projection on the simplex with z = z1[j]
480     for j in range(int(Tf/7)):
481         if j > 2:
482             vec = np.concatenate((controls[0][7*j:7*(j+1)], controls
    [1][7*j:7*(j+1)], controls[0][7*j:7*(j+1)]))
483             vec = projection_simplex_sort(vec, z1[j])
484             for i in range(7*j, 7*(j+1)):
485                 controls[0][i] = max(min(vec[i - 7*j], state[0, i-1])
    , 0)
486                 controls[1][i] = max(min(vec[i - 7*j +7], state[4, i
    -1]), 0)
487                 controls[2][i] = max(min(vec[i - 7*j +14], state[6, i
    -1]), 0)
488         else:
```

```
            controls[2][7*j:7*(j+1)] = np.zeros(7)
            vec = np.concatenate((controls[0][7*j:7*(j+1)], controls
    [1][7*j:7*(j+1)]))
            vec = projection_simplex_sort(vec, z1[j])
            for i in range(7*j, 7*(j+1)):
                if (i > 0):
                    controls[0][i] = max(min(vec[i - 7*j], state[0, i
    -1]), 0)
                    controls[1][i] = max(min(vec[i - 7*j +7], state
    [4, i-1]), 0)
                else:
                    controls[0][i] = max(min(vec[i - 7*j], state[0, i
    ]), 0)
                    controls[1][i] = max(min(vec[i - 7*j +7], state
    [4, i]), 0)

     # MULTI-PROJECTION CYCLE
     for i in range(0,k):

         #STEP1: Projection on maximum elapsing time constraint
         for j in range(Tf+1):
             if ((j>=42) & (np.sum(controls[0][:j-42+1] ) -  np.sum(
    controls[2][:j+1]) > 0)):
                 vec2 = controls[2][:j+1]
                 vec2 = projection_simplex_sort(vec2, np.sum(controls
    [0][:j-42+1]))
                 controls[2][:j+1] = np.minimum(vec2, vax_max[:j+1])
                 for s in range(j+1):
                     vec4 = np.array([controls[0][s], controls[1][s]])

                     if (vax_max[s] - controls[2][s] <= 0):
                         vec4 = np.zeros(len(vec4))
                     else:
                         vec4 = projection_simplex_sort(vec4, max(
    vax_max[s] - controls[2][s],0))
                     if s == 0 :
                         controls[0][s] = max(min(vec4[0], state[0,s])
    ,0)
                         controls[1][s] = max(min(vec4[1], state[4,s])
    ,0)
                     else:
                         controls[0][s] = max(min(vec4[0], state[0,s
    -1]),0)
                         controls[1][s] = max(min(vec4[1], state[4,s
    -1]),0)

         #STEP2: Projection on maximum weekly administrations
    constraint and maximum daily capacity
         for j in range(int(Tf/7)):
             if j > 2:
                 vec = np.concatenate((controls[0][7*j:7*(j+1)],
```

```
    controls[1][7*j:7*(j+1)], controls[0][7*j:7*(j+1)]))
                vec = projection_simplex_sort(vec, z1[j])
                for i in range(7*j, 7*(j+1)):
                    controls[0][i] = max(min(vec[i - 7*j], state[0, i
-1]), 0)
                    controls[1][i] = max(min(vec[i - 7*j +7], state
[4, i-1]), 0)
                    controls[2][i] = max(min(vec[i - 7*j +14], state
[6, i-1]), 0)
            else:
                controls[2][7*j:7*(j+1)] = np.zeros(7)
                vec = np.concatenate((controls[0][7*j:7*(j+1)],
controls[1][7*j:7*(j+1)]))
                vec = projection_simplex_sort(vec, z1[j])
                for d in range(7*j, 7*(j+1)):
                    if (d > 0):
                        controls[0][d] = max(min(vec[d - 7*j], state
[0, d-1]), 0)
                        controls[1][d] = max(min(vec[d - 7*j +7],
state[4, d-1]), 0)
                    else:
                        controls[0][d] = max(min(vec[d - 7*j], state
[0, d]), 0)
                        controls[1][d] = max(min(vec[d - 7*j +7],
state[4, d]), 0)

        #STEP3: Projection on minimum elapsing time constraint
        for j in range(Tf+1):
            if ((j>=21) & (np.sum(controls[0][:j-21+1]) -  np.sum(
controls[2][:j+1]) < 0)):

                vec2 = controls[2][:j+1]
                vec2 = projection_simplex_sort(vec2, np.sum(controls
[0][:j-21+1]))
                controls[2][:j+1] = np.minimum(vec2, vax_max[:j+1])
                for s in range(j+1):
                    vec4 = np.array([controls[0][s], controls[1][s]])

                    if (vax_max[s] - controls[2][s] <= 0):
                        vec4 = np.zeros(len(vec4))
                    else:
                        vec4 = projection_simplex_sort(vec4, max(
vax_max[s] - controls[2][s],0))
                    if s == 0 :
                        controls[0][s] = max(min(vec4[0], state[0,s])
,0)
                        controls[1][s] = max(min(vec4[1], state[4,s])
,0)
                    else:
                        controls[0][s] = max(min(vec4[0], state[0,s
-1]),0)
```

```
                        controls[1][s] = max(min(vec4[1], state[4,s
    -1]),0)

        #STEP2bis: Projection on maximum weekly administrations
    constraint and maximum daily capacity
        for j in range(int(Tf/7)):
            if j > 2:
                vec = np.concatenate((controls[0][7*j:7*(j+1)],
    controls[1][7*j:7*(j+1)], controls[0][7*j:7*(j+1)]))
                vec = projection_simplex_sort(vec, z1[j])
                for i in range(7*j, 7*(j+1)):
                    controls[0][i] = max(min(vec[i - 7*j], state[0, i
    -1]), 0)
                    controls[1][i] = max(min(vec[i - 7*j +7], state
    [4, i-1]), 0)
                    controls[2][i] = max(min(vec[i - 7*j +14], state
    [6, i-1]), 0)
            else:
                controls[2][7*j:7*(j+1)] = np.zeros(7)
                vec = np.concatenate((controls[0][7*j:7*(j+1)],
    controls[1][7*j:7*(j+1)]))
                vec = projection_simplex_sort(vec, z1[j])
                for d in range(7*j, 7*(j+1)):
                    if (d > 0):
                        controls[0][d] = max(min(vec[d - 7*j], state
    [0, d-1]), 0)
                        controls[1][d] = max(min(vec[d - 7*j +7],
    state[4, d-1]), 0)
                    else:
                        controls[0][d] = max(min(vec[d - 7*j], state
    [0, d]), 0)
                        controls[1][d] = max(min(vec[d - 7*j +7],
    state[4, d]), 0)

    return controls


def main(argv):
    Tf = int(argv[0])

    dt = 1
    print('Initializing...')
    print('
    ----------------------------------------------------------------')

    # alpha: inverse of the incubation time
    # gamma: infectous spassing rate
    # muR: re-infection rate from recovered class
    # muV: re-infection rate from vaccinated class
    # sigma: fraction of vaccine effectiveness on infection
    trasmission
```

```
    # omega: hospedalizing rate
    # theta: fraction for accounting for vaccine effectiveness on effects reduction
    # f: fatality rate


    #vax_history = np.array(75e3 * np.ones(Tf + 1))
    #vaccine_dataframe = pd.read_csv('vaccini_grouped_lombardia.csv', sep =',')
    #vax_history = vaccine_dataframe['totale'].to_numpy()

    #vax_history = np.array(50e3 * np.ones(Tf + 1))

    #vax_history = np.array(35e3 * np.ones(Tf+1))
    #vax_history[0:60] = 10e3 * np.ones(60)
    #vax_history[60:101] = 25e3 * np.ones(41)

    # Definition of daily amount of doses
    vaccine_dataframe = pd.read_csv('vaccini_grouped_lom_ALL_29_maggio.csv', sep =',')
    vax_history = vaccine_dataframe['totale'].to_numpy()
    prima_history = vaccine_dataframe['prima_dose'].to_numpy()
    seconda_history = vaccine_dataframe['seconda_dose'].to_numpy()

    #consegne_dataframe = pd.read_csv('vaccini_consegne_grouped_lom_ALL_29_maggio.csv', sep = ',')
    #consegne_history = consegne_dataframe['dosi_settimanali'].to_numpy()[:21]

    consegne_history = 385000 * np.ones(21 + 6)


    # Rt* computation
    Infectives_dataframe = pd.read_csv('Lombardia_infetti_147.csv', sep = ',')
    Suscettibili_147 = Infectives_dataframe['suscettibili'].to_numpy()
    Vaccinated_first_147 = Infectives_dataframe['vaccinati_prima_dose'].to_numpy()
    Infectives_147 = Infectives_dataframe['totale_positivi'].to_numpy()
    Deceased_147 = Infectives_dataframe['deceduti'].to_numpy()

    gamma_2 = 1/9
    logI=np.log(Infectives_147)
    Rt=(logI[7:]-logI[:-7])/7/gamma_2+1

    # Parameters setting

    #params = [0.182, 0.211, 4.76e-4, 4.76e-4, 0.25, 0.0690, 0.15, 2.794e-3] # LOMBARDIA
```

```
    #params = [0.182, 0.211, 4.76e-4, 4.76e-4, 0.40, 0.0690, 0.15, 2.794e-3] # sigma = 0.40 -> vaccine effectiveness 0.6
    #params = [0.182, 0.211, 4.76e-4, 4.76e-4, 0.10, 0.0690, 0.15, 2.794e-3] # sigma = 0.10 -> vaccine effectiveness 0.9
    #params = [0.182, 0.211, 4.76e-4, 4.76e-4, 0.25, 0.0690, 0.30, 2.794e-3] # theta = 0.30 -> vaccine effectiveness 0.7
    params = [0.182, 0.211, 4.76e-4, 4.76e-4, 0.25, 0.0690, 0.40, 2.794e-3] # theta = 0.40 -> vaccine effectiveness 0.6

    N = 10103969

    # \beta definition
    beta_2 = Rt * params[1] * N /(Suscettibili_147[:-7] + params[4] * Vaccinated_first_147[:-7])

    beta = np.mean(beta_2) * np.ones(Tf+1)


    # History of Susceptibles and Vaccinationed with first dose 15 days before the beginning of the simulation (to use in the SEIHRDVW computation)

    states_15_start = list()

    #susc_dataframe = pd.read_csv('', sep = ',')
    #susc_15 = susc_dataframe.to_numpy()
    #states_15_start.append(susc_15)
    #states_15_start.append(9620987 * np.ones(15))

    # Susceptible_history
    states_15_start.append(np.array([9651234, 9648490, 9646546, 9644751, 9643801, 9641523, 9639370, 9636714, 9634086, 9632480, 9632014, 9631441, 9630598, 9628931, 9624866]))
    states_15_start.append(np.zeros(15))

    states_15_start[1][-1] = 714
    states_15_start[1][-2] = 114
    states_15_start[1][-3] = 104
    states_15_start[1][-4] = 107
    states_15_start[1][-5] = 717


    # INITIAL CONDITIONS
    #IC = [57000000, 10800, 569896, 1463111, 74159,0,2000000,0]
    #IC = [57000000, 10800, 569896, 1463111, 74159,0,4e5,0]
    #IC = [48000000, 10680, 563479, 28949, 2933757, 109847, 10324127, 3237582] # 1st APRIL 2021
    IC = [9620987, 2.5e4, 5.5e4, 50520, 4.01e5, 25203, 1779, 0]
    #IC = [48000000, 10680, 563479, 28949, 2933757, 109847, 0, 0]
```

```
683
684     # META ALGORITHM: STEP 1 -> Define initial Guess and parameters
685     # Define initial guess for vaccinations
686     controls = list()
687     #controls.append(25e3 * np.ones(Tf+1))
688     #controls.append(5e5 * np.ones(Tf+1))
689     #controls.append(25e3 * np.ones(Tf+1))
690     #controls.append(5e5 * np.ones(Tf+1))
691
692     controls.append(np.zeros(Tf+1))
693     controls.append(np.zeros(Tf+1))
694     controls.append(np.zeros(Tf+1))
695
696     #controls.append(np.zeros(Tf+1))
697     #controls.append(0.1007 * np.ones(int(Tf/7)+1))
698     #controls.append(np.array([0.23225, 0.23225, 0.23225, 0.23225,
    0.23225, 0.26315, 0.26315, 0.26315, 0.26315, 0.26172, 0.26172,
    0.26172, 0.26172, 0.26172, 0.26172, 0.19872]))
699     #controls.append(0.26172 * np.ones(int(Tf/7)+1))
700
701     #T_f = 147 only
702     #controls.append(np.array([0.23225, 0.23225, 0.23225, 0.23225,
    0.23225, 0.26315, 0.26315, 0.26315, 0.26315, 0.26172, 0.26172,
    0.26172, 0.26172, 0.26172, 0.26172, 0.19872, 0.19872, 0.19872,
    0.19872, 0.19872, 0.19872, 0.19872]))
703
704     controls.append(beta)
705     # Path for saving data and figures
706     path = "/home/giovanni/Desktop/OC_LOM/confronto_varianti_vaccini/
    beta_026617/theta/060/I_squared_E_dot/"
707
708     # Maximum steps and tolerance for the PGD Algorithm
709     nsteps = 100 #20
710     tolerance = 1e-7
711
712     # Step length PGD Algorithm
713     #learning_rate_vax = 1e-5
714     learning_rate_vax = 1e-2
715     #learning_rate_vax = 1e-5
716
717     ep = 1
718     err_min = tolerance + 1
719     err_old = 1e-7
720     state = np.zeros((len(IC),Tf+1))
721     multipliers = np.zeros((len(IC),Tf+1))
722     vax_max = max(vax_history) * np.ones(Tf+1)
723
724     #vaccine_dataframe = pd.read_csv('vaccini_grouped_lombardia.csv',
    sep =',')
725     #vax_history = vaccine_dataframe['totale'].to_numpy()
726
```

```
    # Function for gradients of the Hamiltonian
    grad_x = jax.grad(Hamiltonian, 2)
    grad_u1 = jax.grad(Hamiltonian, 3)
    grad_u2 = jax.grad(Hamiltonian, 4)
    grad_u3 = jax.grad(Hamiltonian, 5)

    # Just in time compilation gradients
    grad_x_jit = jax.jit(grad_x)
    grad_u1_jit = jax.jit(grad_u1)
    grad_u2_jit = jax.jit(grad_u2)
    grad_u3_jit = jax.jit(grad_u3)

    sum_gradient = 0.0
    sum_old = 1.0
    history = list()

    print('Gradient Loop...')
    print('
    ----------------------------------------------------------------')

    while (ep <= nsteps and err_min > tolerance):

        #Final time control only
        #lambda_fin = [0 , 0, 0, 0, 0,  state[5,Tf], 0,0] #1e3 *

        lambda_fin = [0 , 0, 0, 0, 0, 0, 0,0]
        sum_gradient = 0.0

        # META ALGORITHM: STEP 2a -> Solution of the state problem
        state = solve_rk4(SEIHRDVW, [0, Tf], IC, 1.0, params,
    controls, states_15_start)
        print('OK STATE')


        # META ALGORITHM: STEP 2b -> Solution of the costate problem
        multipliers = solve_rk4_H(grad_x_jit, [Tf, 0], lambda_fin,
    -1.0, state, params, controls, multipliers, states_15_start)
        print('OK MULTI')


        # META ALGORITHM: STEPS 2c -> Computation of the gradient,
    Gradient Descent step and Multi-Projection
        for j in range(Tf+1):
            U1, U2, U3, beta = control_at_time(controls, j)
            grad_u1_vec = 0.0
            grad_u2_vec = 0.0
            grad_u3_vec = 0.0

            # Computation of the gradients
            if j >= 15:
                grad_u1_vec = grad_u1_jit(j, params, state[:,j], U1,
```

```
    U2, U3, beta, multipliers[:, j], state[0, j-15], state[6, j-15])
774                 grad_u2_vec = grad_u2_jit(j, params, state[:,j], U1,
    U2, U3, beta, multipliers[:, j], state[0, j-15], state[6, j-15])
775                 grad_u3_vec = grad_u3_jit(j, params, state[:,j], U1,
    U2, U3, beta, multipliers[:, j], state[0, j-15], state[6, j-15])
776             else:
777                 grad_u1_vec = grad_u1_jit(j, params, state[:,j], U1,
    U2, U3, beta, multipliers[:, j], states_15_start[0][j],
    states_15_start[1][j])
778                 grad_u2_vec = grad_u2_jit(j, params, state[:,j], U1,
    U2, U3, beta, multipliers[:, j], states_15_start[0][j],
    states_15_start[1][j])
779                 grad_u3_vec = grad_u3_jit(j, params, state[:,j], U1,
    U2, U3, beta, multipliers[:, j], states_15_start[0][j],
    states_15_start[1][j])
780
781
782             # Gradient Descent step
783             controls[0][j] -=  learning_rate_vax * (1 - ep / nsteps)
    * grad_u1_vec #U1
784             controls[1][j] -=  learning_rate_vax * (1 - ep / nsteps)
    * grad_u2_vec #U2
785             controls[2][j] -=  learning_rate_vax * (1 - ep / nsteps)
    * grad_u3_vec #U3
786
787             sum_gradient += abs(grad_u1_vec) + abs(grad_u2_vec) + abs
    (grad_u3_vec)
788
789
790         # Multi-Projection of the control variables
791         controls = proj_complete(controls, consegne_history, vax_max,
     state, Tf)
792
793
794         print('OK GRAD')
795         print('Gradient: ', sum_gradient)
796
797         # Saving cost functional values
798         history.append(cost_functional(params, state, controls, Tf))
799
800         print('Cost Functional: ', cost_functional(params, state,
    controls, Tf))
801         print('Iteration', ep, 'Done')
802         print('Error: ',err_min)
803
804
805         # META ALGORITHM: STEP 2d -> Stopping criterium verification
806         err_min = abs(sum_old - sum_gradient)/abs(sum_old)
807         sum_old = sum_gradient
808
809         ep += 1
```

```
        print('First check: ', ep <= nsteps)
        print('Second check: ', err_min > tolerance)

        print('
    ----------------------------------------------------------------')

    print('End Loop...')
    print('
    ----------------------------------------------------------------')

    #Checking constraints are fulfilled
    for j in range(Tf+1):
        if (j%7 == 0 and j<Tf):
            if not check1(controls[0][j:j+7], controls[1][j:j+7],
    controls[2][j:j+7], consegne_history[math.floor(j/7)]):
                print('Check 1 fails at week', int(j/7) + 1)
        if j >= 21:
            if not check2(controls[0][:j+1-21], controls[1][:j+1-21],
     controls[2][:j+1]):
                print('Check 2 fails at time', j)

        if j >= 42:
            if not check3(controls[0][:j+1-42], controls[1][:j+1-42],
     controls[2][:j+1]):
                print('Check 3 fails at time', j)

    # Deleting days without imposed constraints
    state_fin = state[:,:-42]
    multipliers_fin = multipliers[:,:-42]
    controls_fin = list()
    controls_fin.append(controls[0][:-42])
    controls_fin.append(controls[1][:-42])
    controls_fin.append(controls[2][:-42])
    controls_fin.append(controls[3][:-42])


    # Plots

    plot(state_fin, path)

    plot_multi(multipliers_fin, path)

    plot_controls(controls_fin, vax_history, consegne_history[:21],
    path, prima_history, seconda_history, Tf - 42)

    plot_Rt(state_fin, controls_fin, params, Tf - 42, path)

    plot_histogram(controls_fin, Tf - 42, path)

    # Saving procedure
```

```
    np.savetxt(path + 'state.csv', state_fin, delimiter = ',')
    np.savetxt(path + 'S.csv', state_fin[0,:], delimiter = ',')
    np.savetxt(path + 'E.csv', state_fin[1,:], delimiter = ',')
    np.savetxt(path + 'I.csv', state_fin[2,:], delimiter = ',')
    np.savetxt(path + 'H.csv', state_fin[3,:], delimiter = ',')
    np.savetxt(path + 'R.csv', state_fin[4,:], delimiter = ',')
    np.savetxt(path + 'D.csv', state_fin[5,:], delimiter = ',')
    np.savetxt(path + 'V.csv', state_fin[6,:], delimiter = ',')
    np.savetxt(path + 'W.csv', state_fin[7,:], delimiter = ',')


    np.savetxt(path + 'prima_dose_S.csv', controls_fin[0], delimiter
    = ',')
    np.savetxt(path + 'dose_R.csv', controls_fin[1], delimiter = ',')
    np.savetxt(path + 'seconda_dose.csv', controls_fin[2], delimiter
    = ',')
    np.savetxt(path + 'beta.csv', controls_fin[3], delimiter = ',')


    plt.loglog(history)
    plt.title('Cost history')
    plt.savefig(path + "cost_hist.png")
    plt.show()

    return 0

if __name__ == "__main__":
    main(sys.argv[1:])
```

Bibliography

[1] Hakimeh Ameri and Kathryn Cooper. A compartmental network model for the spread of whooping cough. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2224–2225. IEEE, 2017.

[2] Suwardi Annas, Muh Isbar Pratama, Muh Rifandi, Wahidah Sanusi, and Syafruddin Side. Stability analysis and numerical simulation of SEIR model for pandemic COVID-19 spread in Indonesia. *Chaos, Solitons & Fractals*, 139:110072, 2020.

[3] Sina Ardabili, Amir Mosavi, Shahab S. Band, and Annamaria R. Varkonyi-Koczy. Coronavirus disease (COVID-19) global prediction using hybrid artificial intelligence method of ann trained with grey wolf optimizer. In *2020 IEEE 3rd International Conference and Workshop in Óbuda on Electrical and Power Engineering (CANDO-EPE)*, pages 000251–000254. IEEE, 2020.

[4] Julien Arino and Paul Van den Driessche. A multi-city epidemic model. *Mathematical Population Studies*, 10(3):175–193, 2003.

[5] Rudianto Artiono, Budi Priyo Prawoto, et al. An epidemic model with age structured of rubella virus: threshold and stability. In *Journal of Physics: Conference Series*, volume 1821, page 012042. IOP Publishing, 2021.

[6] PLC AstraZeneca. COVID-19 vaccine AstraZeneca confirms 100% protection against severe disease, hospitalisation and death in the primary analysis of Phase III trials, 2021.

[7] Emmanuel Afolabi Bakare, Y.A. Adekunle, and K.O. Kadiri. Modelling and simulation of the dynamics of the transmission of measles. *International Jounal of Computer Trends and Technology*, 3:174–178, 2012.

[8] Emmanuel Afolabi Bakare, A. Nwagwo, and E. Danso-Addo. Optimal control analysis of an SIR epidemic model with constant recruitment. *International Journal of Applied Mathematics Research*, 3(3):273, 2014.

[9] Roberto Battiston. Un modo semplice per calcolare $\mathcal{R}_t$. *Scienza in rete*, 2020. https://www.scienzainrete.it/articolo/modo-semplice-calcolare-rt/roberto-battiston/2020-11-20.

[10] Martin Benning, Elena Celledoni, Matthias J. Ehrhardt, Brynjulf Owren, and Carola-Bibiane Schönlieb. Deep learning as optimal control problems: Models and numerical methods. *arXiv preprint arXiv:1904.05657*, 2019.

[11] Giulia Bertaglia and Lorenzo Pareschi. Hyperbolic compartmental models for epidemic spread on networks with uncertain data: application to the emergence of COVID-19 in Italy, 2021.

[12] Mariano Bizzarri, Mario Di Traglia, Alessandro Giuliani, Annarita Vestri, Valeria Fedeli, and Alberto Prestininzi. New statistical RI index allow to better track the dynamics of COVID-19 outbreak in Italy. *Scientific Reports*, 10(1):1–13, 2020.

[13] Mathieu Blondel, Akinori Fujino, and Naonori Ueda. Large-scale multiclass support vector machine training via euclidean projection onto the simplex. In *2014 22nd International Conference on Pattern Recognition*, pages 1289–1294. IEEE, 2014.

[14] Joseph-Frédéric Bonnans, Jean Charles Gilbert, Claude Lemaréchal, and Claudia A Sagastizábal. *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media, 2006.

[15] Stephen Boyd, Jon Dattorro, et al. Alternating projections. *EE392o, Stanford University*, 2003.

[16] Marco Bramanti, Carlo Domenico Pagani, and Sandro Salsa. *Analisi matematica 2*. Zanichelli, 2008.

[17] Michael Brin and Garrett Stuck. *Introduction to dynamical systems*. Cambridge university press, 2002.

[18] Arthur E. Bryson and Yu-Chi Ho. *Applied optimal control: optimization, estimation, and control*. Routledge, 2018.

[19] Svetlana Bunimovich-Mendrazitsky and Lewi Stone. Modeling polio as a disease of development. *Journal of theoretical biology*, 237(3):302–315, 2005.

[20] Giuseppe C. Calafiore, Carlo Novara, and Corrado Possieri. A modified SIR model for the COVID-19 contagion in Italy. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3889–3894. IEEE, 2020.

[21] Finlay Campbell, Brett Archer, Henry Laurenson-Schafer, Yuka Jinnai, Franck Konings, Neale Batra, Boris Pavlin, Katelijn Vandemaele, Maria D. Van Kerkhove, Thibaut Jombart, et al. Increased transmissibility and global spread of SARS-CoV-2 variants of concern as at June 2021. *Eurosurveillance*, 26(24):2100509, 2021.

[22] Robert Stephen Cantrell and Chris Cosner. *Spatial ecology via reaction-diffusion equations*. John Wiley & Sons, 2004.

[23] Francisco Caramelo, Nuno Ferreira, and Barbara Oliveiros. Estimation of risk factors for COVID-19 mortality-preliminary results. *MedRxiv*, 2020.

[24] José M Carcione, Juan E Santos, Claudio Bagaini, and Jing Ba. A simulation of a COVID-19 epidemic based on a deterministic SEIR model. *Frontiers in public health*, 8:230, 2020.

[25] Zain Chagla. The BNT162b2 (BioNTech/Pfizer) vaccine had 95% efficacy against COVID-19 $\geq$ 7 days after the 2nd dose. *Annals of Internal Medicine*, 174(2):JC15, 2021.

[26] Yan Chen, Peilong Lu, and C. Chang. A time-dependent SIR model for COVID-19. *IEEE Transactions on Network Science and Engineering*, 7(4):3279–3294, 2020.

[27] Ward Cheney and Allen A Goldstein. Proximity maps for convex sets. *Proceedings of the American Mathematical Society*, 10(3):448–450, 1959.

[28] Carmen Chicone. *Ordinary differential equations with applications*, volume 34. Springer Science & Business Media, 2006.

[29] Sebastian Contreras and Viola Priesemann. Risking further COVID-19 waves despite vaccination. *The Lancet Infectious Diseases*, 21(6):745–746, 2021.

[30] Ian Cooper, Argha Mondal, and Chris G. Antonopoulos. A SIR model assumption for the spread of COVID-19 in different communities. *Chaos, Solitons & Fractals*, 139:110057, 2020.

[31] Ana Corberán-Vallet, Francisco-José Santonja, Marc Jornet-Sanz, and R-J. Villanueva. Modeling chickenpox dynamics with a discrete time bayesian stochastic compartmental model. *Complexity*, 2018, 2018.

[32] Nicola Decaro and Alessio Lorusso. Novel human coronavirus (SARS-CoV-2): A lesson from animal coronaviruses. *Veterinary microbiology*, 244:108693, 2020.

[33] Edmund X. DeJesus and Charles Kaufman. Routh-Hurwitz criterion in the examination of eigenvalues of a system of nonlinear ordinary differential equations. *Physical Review A*, 35(12):5288, 1987.

[34] Gruppo di lavoro ISS Prevenzione e controllo delle Infezioni. Indicazioni ad interim sulle misure di prevenzione e controllo delle infezioni da SARS-CoV-2 in tema di varianti e vaccinazione anti COVID-19. Versione del 13 marzo 2021 (rapporto ISS COVID-19 n. 4/ 2021).

[35] Odo Diekmann, Johan Andre Peter Heesterbeek, and Johan AJ Metz. On the definition and the computation of the basic reproduction ratio $\mathcal{R}_0$ in models for infectious diseases in heterogeneous populations. *Journal of mathematical biology*, 28(4):365–382, 1990.

[36] Nigel J. Dimmock, Andrew J. Easton, and Keith N. Leppard. *Introduction to modern virology*. John Wiley & Sons, 2015.

[37] Tong-Ren Ding. *Approaches to the qualitative theory of ordinary differential equations: dynamical systems and nonlinear oscillations*, volume 3. World Scientific Publishing Company, 2007.

[38] Zhilan Feng, Carlos Castillo-Chavez, and Angel F. Capurro. A model for tuberculosis with exogenous reinfection. *Theoretical population biology*, 57(3):235–247, 2000.

[39] Zhilan Feng and Horst R. Thieme. Recurrent outbreaks of childhood diseases revisited: the impact of isolation. *Mathematical biosciences*, 128(1-2):93–130, 1995.

[40] Athanasios S. Fokas, Nikolaos Dikaios, and George A. Kastis. COVID-19: predictive mathematical models for the number of deaths in South Korea, Italy, Spain, France, UK, Germany, and USA. *MedRxiv*, 2020.

[41] Lauren Forchette, William Sebastian, and Tuoen Liu. A comprehensive review of COVID-19 virology, vaccines, variants, and therapeutics. *Current Medical Science*, pages 1–15, 2021.

[42] Holly Gaff and Elsa Schaefer. Optimal control applied to vaccination and treatment strategies for various epidemiological models. *Mathematical Biosciences & Engineering*, 6(3):469, 2009.

[43] Holly D. Gaff, David M. Hartley, and Nicole P. Leahy. An epidemiological model of Rift Valley fever. *Electronic Journal of Differential Equations*, 2007(115), 2007.

[44] Marino Gatto, Enrico Bertuzzo, Lorenzo Mari, Stefano Miccoli, Luca Carraro, Renato Casagrandi, and Andrea Rinaldo. Spread and dynamics of the COVID-19 epidemic in Italy: Effects of emergency containment measures. *Proceedings of the National Academy of Sciences*, 117(19):10484–10491, 2020.

[45] Nigel J Gay. Modeling measles, mumps, and rubella: implications for the design of vaccination programs. *Infection Control & Hospital Epidemiology*, 19(8):570–573, 1998.

[46] Nigel J. Gay, Louise M. Hesketh, Paul Morgan-Capner, and Elizabeth Miller. Interpretation of serological surveillance data for measles using mathematical

models: implications for vaccine strategy. *Epidemiology & Infection*, 115(1):139–156, 1995.

[47] Giulia Giordano, Franco Blanchini, Raffaele Bruno, Patrizio Colaneri, Alessandro Di Filippo, Angela Di Matteo, and Marta Colaneri. Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy. *Nature medicine*, 26(6):855–860, 2020.

[48] Federico Gobbi, Dora Buonfrate, Lucia Moro, Paola Rodari, Chiara Piubelli, Sara Caldrer, Silvia Riccetti, Alessandro Sinigaglia, and Luisa Barzon. Antibody response to the BNT162b2 mRNA COVID-19 vaccine in subjects with prior SARS-CoV-2 infection. *Viruses*, 13(3):422, 2021.

[49] Alberto Godio, Francesca Pace, and Andrea Vergnano. SEIR modeling of the Italian epidemic of SARS-CoV-2 using computational swarm intelligence. *International Journal of Environmental Research and Public Health*, 17(10):3535, 2020.

[50] Andrew Golub, Wilpen L. Gorr, and Peter R. Gould. Spatial diffusion of the HIV/AIDS epidemic: modeling implications and case study of AIDS incidence in Ohio. *Geographical analysis*, 25(2):85–100, 1993.

[51] Elena Gubar and Quanyan Zhu. Optimal control of influenza epidemic model with virus mutations. In *2013 European Control Conference (ECC)*, pages 3125–3130. IEEE, 2013.

[52] Morton E. Gurtin and Richard C. MacCamy. Non-linear age-dependent population dynamics. *Archive for Rational Mechanics and Analysis*, 54(3):281–300, 1974.

[53] Jack K. Hale and Hüseyin Koçak. *Dynamics and bifurcations*, volume 3. Springer Science & Business Media, 2012.

[54] Shaobo He, Yuexi Peng, and Kehui Sun. SEIR modeling of the COVID-19 and its dynamics. *Nonlinear dynamics*, 101(3):1667–1680, 2020.

[55] Herbert W. Hethcote. Three basic epidemiological models. In *Applied mathematical ecology*, pages 119–144. Springer, 1989.

[56] Andrew N. Hill and Ira M. Longini Jr. The critical vaccination fraction for heterogeneous epidemic models. *Mathematical biosciences*, 181(1):85–106, 2003.

[57] Alexandra Hogan, Peter Winskill, Oliver Watson, Patrick Walker, Charles Whittaker, Marc Baguelin, David Haw, Alessandra Lochen, and Kat yand others Gaythorpe. Report 33: modelling the allocation and impact of a COVID-19 vaccine. *Imperial College of London Reports*, 2020.

[58] Juan Hou and Zhidong Teng. Continuous and impulsive vaccination of SEIR epidemic models with saturation incidence rates. *Mathematics and Computers in Simulation*, 79(10):3038–3054, 2009.

[59] Qingwen Hu and Xingfu Zou. Optimal vaccination strategies for an influenza epidemic model. *Journal of Biological Systems*, 21(04):1340006, 2013.

[60] Zixin Hu, Qiyang Ge, Shudi Li, Li Jin, and Momiao Xiong. Artificial intelligence forecasting of COVID-19 in China. *arXiv preprint arXiv:2002.07112*, 2020.

[61] Enahoro A. Iboi, Calistus N. Ngonghala, and Abba B. Gumel. Will an imperfect vaccine curtail the COVID-19 pandemic in the us? *Infectious Disease Modelling*, 5:510–524, 2020.

[62] Hisashi Inaba. Mathematical analysis of an age-structured SIR epidemic model with vertical transmission. *Discrete & Continuous Dynamical Systems-B*, 6(1):69, 2006.

[63] Alfredo N. Iusem. On the convergence properties of the projected gradient method for convex optimization. *Computational & Applied Mathematics*, 22:37–52, 2003.

[64] William Ogilvy Kermack and Anderson G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

[65] William Ogilvy Kermack and Anderson G. McKendrick. Contributions to the mathematical theory of epidemics. II.—The problem of endemicity. *Proceedings of the Royal Society of London. Series A, containing papers of a mathematical and physical character*, 138(834):55–83, 1932.

[66] William Ogilvy Kermack and Anderson G. McKendrick. Contributions to the mathematical theory of epidemics. III.—Further studies of the problem of endemicity. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 141(843):94–122, 1933.

[67] Bouchaib Khajji, Abdelfatah Kouidere, Mohamed Elhia, Omar Balatif, and Mostafa Rachik. Fractional optimal control problem for an age-structured model of COVID-19 transmission. *Chaos, Solitons & Fractals*, 143:110625, 2021.

[68] Donald E. Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.

[69] Denise Kirschner, Suzanne Lenhart, and Steve Serbin. Optimal control of the chemotherapy of HIV. *Journal of mathematical biology*, 35(7):775–792, 1997.

[70] Pratima Kumari and Durga Toshniwal. Real-time estimation of COVID-19 cases using machine learning and mathematical models-the case of India. In *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)*, pages 369–374. IEEE, 2020.

[71] Yuliya N. Kyrychko, Konstantin B. Blyuss, and Igor Brovchenko. Mathematical modelling of the dynamics and containment of COVID-19 in Ukraine. *Scientific reports*, 10(1):1–11, 2020.

[72] Gabrièle Laborde-Balen, Bernard Taverne, Cheikh Tidiane Ndour, Charles Kouanfack, Martine Peeters, Ibra Ndoye, and Eric Delaporte. The fourth HIV epidemic. *The Lancet Infectious Diseases*, 18(4):379–380, 2018.

[73] John Denholm Lambert et al. *Numerical methods for ordinary differential systems*, volume 146. Wiley New York, 1991.

[74] Joseph Chadi Lemaitre, Damiano Pasetto, Mario Zanon, Enrico Bertuzzo, Lorenzo Mari, Stefano Miccoli, Renato Casagrandi, Marino Gatto, and Andrea Rinaldo. Optimizing the spatio-temporal allocation of COVID-19 vaccines: Italy as a case study. *medRxiv*, 2021.

[75] Jianquan Li and Yali Yang. SIR-SVS epidemic models with continuous and impulsive vaccination strategies. *Journal of theoretical biology*, 280(1):108–116, 2011.

[76] Michael Y. Li and James S. Muldowney. Global stability for the SEIR model in epidemiology. *Mathematical biosciences*, 125(2):155–164, 1995.

[77] Xianning Liu, Yasuhiro Takeuchi, and Shingo Iwami. SVIR epidemic models with vaccination strategies. *Journal of Theoretical Biology*, 253(1):1–11, 2008.

[78] Ira M. Longini Jr. and M. Elizabeth Halloran. Strategy for distribution of influenza vaccine to high-risk groups and children. *American journal of epidemiology*, 161(4):303–306, 2005.

[79] Faray Majid, Michael Gray, Aditya M. Deshpande, Subramanian Ramakrishnan, Manish Kumar, and Shelley Ehrlich. Non-Pharmaceutical Interventions as Controls to mitigate the spread of epidemics: An analysis using a spatiotemporal PDE model and COVID–19 data. *ISA transactions*, 2021.

[80] Lorenzo Mari, Renato Casagrandi, Enrico Bertuzzo, Damiano Pasetto, Stefano Miccoli, Andrea Rinaldo, and Marino Gatto. The epidemicity index of recurrent SARS-CoV-2 infections. *Nature communications*, 12(1):1–12, 2021.

[81] Maia Martcheva. *An introduction to mathematical epidemiology*, volume 61. Springer, 2015.

[82] Laura Matrajt, Julia Eaton, Tiffany Leung, and Elizabeth R. Brown. Vaccine optimization for COVID-19: Who to vaccinate first? *Science Advances*, 7(6):eabf1374, 2020.

[83] Sam Moore, Edward M. Hill, Michael J. Tildesley, Louise Dyson, and Matt J. Keeling. Vaccination and non-pharmaceutical interventions for COVID-19: a mathematical modelling study. *The Lancet Infectious Diseases*, 21(6):793–802, 2021.

[84] David M. Morens, Gregory K. Folkers, and Anthony S. Fauci. What is a pandemic?, 2009.

[85] Bachir Nail, Abdelaziz Rabehi, Belkacem Bekhiti, and Taha Arbaoui. A new design of an adaptive model of infectious diseases based on artificial intelligence approach: monitoring and forecasting of COVID-19 epidemic cases. *medRxiv*, 2020.

[86] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[87] Nick H. Ogden, Aamir Fazil, Julien Arino, Philippe Berthiaume, David N. Fisman, Amy L. Greer, Antoinette Ludwig, Victoria Ng, Ashleigh R. Tuite, Patricia Turgeon, et al. Artificial intelligence in public health: Modelling scenarios of the epidemic of COVID-19 in Canada. *Canada Communicable Disease Report*, 46(8):198, 2020.

[88] Nicola Parolini, Giovanni Ardenghi, Luca Dede', and Alfio Quarteroni. A mathematical dashboard for the analysis of italian COVID-19 epidemic data, 2021.

[89] Nicola Parolini, Luca Dede', Paola Francesca Antonietti, Giovanni Ardenghi, Andrea Manzoni, Edie Miglio, Andrea Pugliese, Marco Verani, and Alfio Quarteroni. SUIHTER: A new mathematical model for COVID-19. Application to the analysis of the second epidemic outbreak in Italy. *arXiv preprint arXiv:2101.03369*, 2021.

[90] Abhilash Perisetti, Hemant Goyal, Mahesh Gajendran, Umesha Boregowda, Rupinder Mann, and Neil Sharma. Prevalence, mechanisms, and implications of gastrointestinal symptoms in COVID-19. *Frontiers in Medicine*, 7:741, 2020.

[91] Presidenza del Consiglio dei Ministri and Dipartimento di Protezione Civile. Dati COVID-19 Italia. *GitHub repository*, 2020.

[92] Presidenza del Consiglio dei Ministri and Dipartimento di Protezione Civile. Dati COVID-19 Vaccini Italia. *GitHub repository*, 2021.

[93] Narinder Singh Punn, Sanjay Kumar Sonbhadra, and Sonali Agarwal. COVID-19 epidemic analysis using machine learning and deep learning algorithms. *MedRxiv*, 2020.

[94] Jing Qin, Chong You, Qiushi Lin, Taojun Hu, Shicheng Yu, and Xiao-Hua Zhou. Estimation of incubation period distribution of COVID-19 using disease onset forward time: a novel cross-sectional and forward follow-up study. *Science advances*, 6(33):eabc1202, 2020.

[95] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical mathematics*, volume 37. Springer Science & Business Media, 2010.

[96] Chiara Reno, Jacopo Lenzi, Antonio Navarra, Eleonora Barelli, Davide Gori, Alessandro Lanza, Riccardo Valentini, Biao Tang, and Maria Pia Fantini. Forecasting COVID-19-associated hospitalizations under different levels of social distancing in Lombardy and Emilia-Romagna, Northern Italy: results from an extended SEIR compartmental model. *Journal of clinical medicine*, 9(5):1492, 2020.

[97] Hannah Ritchie, Esteban Ortiz-Ospina, Diana Beltekian, Edouard Mathieu, Joe Hasell, Bobbie Macdonald, Charlie Giattino, Cameron Appel, Lucas Rodés-Guirao, and Max Roser. Coronavirus pandemic (COVID-19). *Our World in Data*, 2020. https://ourworldindata.org/coronavirus.

[98] Sandro Salsa and Annamaria Squellati. *Dynamical Systems and Optimal Control*. Bocconi University Press, 2018.

[99] Reza Sameni. Mathematical modeling of epidemic diseases; a case study of the COVID-19 coronavirus. *arXiv preprint arXiv:2003.11371*, 2020.

[100] Almut Scherer and Angela McLean. Mathematical models of vaccination. *British Medical Bulletin*, 62(1):187–199, 2002.

[101] Shai Shalev-Shwartz and Yoram Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 7:1567–1599, 2006.

[102] Afroza Shirin, Yen Ting Lin, and Francesco Sorrentino. Data-driven optimized control of the COVID-19 epidemics. *Scientific reports*, 11(1):1–16, 2021.

[103] Boris Shulgin, Lewi Stone, and Zvia Agur. Pulse vaccination strategy in the SIR epidemic model. *Bulletin of mathematical biology*, 60(6):1123–1148, 1998.

[104] Cristiana J. Silva and Delfim F.M. Torres. A SICA compartmental model in epidemiology with application to HIV/AIDS in Cape Verde. *Ecological complexity*, 30:70–75, 2017.

[105] Sadhika Sood. Psychological effects of the Coronavirus disease-2019 pandemic. *Research & Humanities in Medical Education*, 7(11):23–26, 2020.

[106] Ian H. Spicknall, Katharine J. Looker, Sami L. Gottlieb, Harrell W. Chesson, Joshua T. Schiffer, Jocelyn Elmes, and Marie-Claude Boily. Review of mathematical models of HSV-2 vaccination: Implications for vaccine development. *Vaccine*, 37(50):7396–7407, 2019.

[107] Robert F. Stengel. *Optimal control and estimation*. Courier Corporation, 1994.

[108] Ne-Zheng Sun and Alexander Sun. *Model calibration and parameter estimation: for environmental and water resource systems*. Springer, 2015.

[109] Zhou Tang, Xianbin Li, and Houqiang Li. Prediction of new coronavirus infection based on a modified SEIR model. *medRxiv*, 2020.

[110] Mark G. Thompson, Jefferey L. Burgess, Allison L. Naleway, Harmony L. Tyner, Sarang K. Yoon, Jennifer Meece, Lauren E.W. Olsho, Alberto J. Caban-Martinez, Ashley Fowlkes, Karen Lutrick, et al. Interim estimates of vaccine effectiveness of BNT162b2 and mRNA-1273 COVID-19 vaccines in preventing SARS-CoV-2 infection among health care personnel, first responders, and other essential and frontline workers–eight US locations, December 2020–March 2021. *Morbidity and Mortality Weekly Report*, 70(13):495, 2021.

[111] O. Torrealba-Rodriguez, R.A. Conde-Gutiérrez, and A.L. Hernández-Javier. Modeling and prediction of COVID-19 in Mexico applying mathematical and computational models. *Chaos, Solitons & Fractals*, 138:109946, 2020.

[112] Fredi Tröltzsch. *Optimal control of partial differential equations: theory, methods, and applications*, volume 112. American Mathematical Soc., 2010.

[113] Paul Van den Driessche and James Watmough. Further notes on the basic reproduction number. In *Mathematical epidemiology*, pages 159–178. Springer, 2008.

[114] Jan-Diederik Van Wees, Sander Osinga, Martijn Van der Kuip, Michael Tanck, Maurice Hanegraaf, Marten Pluymaekers, Olwijn Leeuwenburgh, Lonneke Van Bijsterveldt, Jaap Zindler, and M.T. Van Furth. Forecasting hospitalization and ICU rates of the COVID-19 outbreak: An efficient SEIR model. *Bull World Health Organ*, 2020.

[115] Alex Viguerie, Alessandro Veneziani, Guillermo Lorenzo, Davide Baroli, Nicole Aretz-Nellesen, Alessia Patton, Thomas E. Yankeelov, Alessandro Reali, Thomas Jr. Hughes, and Ferdinando Auricchio. Diffusion–reaction compartmental models formulated in a continuum mechanics framework: application to COVID-19, mathematical analysis, and numerical study. *Computational Mechanics*, 66(5):1131–1152, 2020.

[116] Merryn Voysey, Sue Ann Costa Clemens, Shabir A. Madhi, Lily Y. Weckx, Pedro M. Folegatti, Parvinder K. Aley, Brian Angus, Vicky L. Baillie, Shaun L. Barnabas, Qasim E. Bhorat, et al. Single-dose administration and the influence of the timing of the booster dose on immunogenicity and efficacy of ChAdOx1 nCoV-19 (AZD1222) vaccine: a pooled analysis of four randomised trials. *The Lancet*, 397(10277):881–891, 2021.

[117] Xinwei Wang, Haijun Peng, Boyang Shi, Dianheng Jiang, Sheng Zhang, and Biaosong Chen. Optimal vaccination strategy of a constrained time-varying SEIR epidemic model. *Communications in Nonlinear Science and Numerical Simulation*, 67:37–48, 2019.

[118] Lilith K. Whittles and Xavier Didelot. Epidemiological analysis of the Eyam plague outbreak of 1665–1666. *Proceedings of the Royal Society B: Biological Sciences*, 283(1830):20160618, 2016.

[119] Gul Zaman, Yong Han Kang, and Il Hyo Jung. Stability analysis and optimal vaccination of an SIR epidemic model. *BioSystems*, 93(3):240–249, 2008.

[120] Shilei Zhao and Hua Chen. Modeling the epidemic dynamics and control of COVID-19 outbreak in China. *Quantitative biology (Beijing, China)*, page 1, 2020.

[121] Ze-Yu Zhao, Yuan-Zhao Zhu, Jing-Wen Xu, Shi-Xiong Hu, Qing-Qing Hu, Zhao Lei, Jia Rui, Xing-Chun Liu, Yao Wang, Meng Yang, et al. A five-compartment model of age-specific transmissibility of SARS-CoV-2. *Infectious diseases of poverty*, 9(1):1–15, 2020.

[122] Zili Zhou, Ning Zhao, Yan Shu, Shengbo Han, Bin Chen, and Xiaogang Shu. Effect of gastrointestinal symptoms in patients with COVID-19. *Gastroenterology*, 158(8):2294, 2020.