



**HAL**  
open science

# Application of optimal control techniques to natural systems management

Emilio Molina

► **To cite this version:**

Emilio Molina. Application of optimal control techniques to natural systems management. Mathematics [math]. Sorbonne University UPMC; Universidad de Chile. Facultad de ciencias físicas y matemáticas, 2022. English. NNT: . tel-03936581

**HAL Id: tel-03936581**

**<https://theses.hal.science/tel-03936581v1>**

Submitted on 12 Jan 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

**SORBONNE UNIVERSITÉ**  
**LJLL**

Doctoral School **École Doctorale Sciences Mathématiques de Paris Centre**

University Department **Laboratoire Jacques-Louis Lions**

Thesis defended by **Emilio Jesús MOLINA OLIVARES**

Defended on **September 14, 2022**

In order to become Doctor from Sorbonne Université

Academic Field **Applied mathematics**

Speciality **Optimal control**

# **Application of optimal control techniques to natural systems management**

**Thesis supervised by** Héctor RAMÍREZ    Supervisor  
Mario SIGALOTTI    Supervisor  
Pierre MARTINON    Co-Supervisor

**Committee members**

<i>Referees</i>	María Soledad ARONNA	Professor at FGV EMap	
	Jean-Baptiste CAILLAU	Professor at Université Côte d'Azur	
<i>Examiners</i>	Emmanuel TRELAT	Professor at Sorbonne Université	Committee President
	Alain RAPAPORT	Senior Researcher at INRAE	
	Jaime ORTEGA	Professor at Universidad de Chile	
<i>Supervisors</i>	Héctor RAMÍREZ	Professor at Universidad de Chile	
	Mario SIGALOTTI	Senior Researcher at INRIA	
	Pierre MARTINON	Junior Researcher at INRIA	

## COLOPHON

Doctoral dissertation entitled “Application of optimal control techniques to natural systems management”, written by Emilio Jesús MOLINA OLIVARES, completed on January 12, 2023, typeset with the document preparation system  $\text{\LaTeX}$  and the yathesis class dedicated to theses prepared in France.

**Keywords:** optimal control, natural system management, mining, epidemiology

**Mots clés :** contrôle optimal, gestion du système naturel, exploitation minière, épidémiologie



This thesis has been prepared at

**Laboratoire Jacques-Louis Lions**

Sorbonne Université  
Campus Pierre et Marie Curie  
4 place Jussieu  
75005 Paris  
France

☎ +33 1 44 27 42 98  
Web Site <https://ljl.math.upmc.fr/>





*A mis padres.*

*A mi familia quienes son los que más creen en mí*





ooox!

---

Nicolás Torres



# Funding

This thesis was partially supported by ANID-PFCHA/Doctorado Nacional/2018-21180348, FONDECYT grant 1201982 and Centro de Modelamiento Matemático (CMM), ACE210010 and FB210005, BASAL funds for center of excellence, all of them from ANID (Chile), and Inria Cage team.



# Acknowledgements

Muchas cosas han pasado en este tiempo mientras hacía la tesis y también hay muchos a quienes agradecer.

Quisiera partir agradeciendo a mis profesores guías por toda la ayuda brindada para realizar este trabajo, muchas gracias a Héctor Remírez por aceptar trabajar conmigo y por su ayuda en concretar una cotutela como era mi idea desde que ingresé al doctorado. *Many thanks to Pierre Martinon for all his help and for sharing his knowledge and expertise on Bocop and numerical methods with me. Thank you, also, for all the help with paperwork at the beginning of the cotutelle. We pioneered the distance meetings since we started with this before Covid-19. Thanks also to Mario Sigalotti for his help in administrative matters and managing the help I received from Inria. During my PhD I had a very pleasant stay in the MISTEA laboratory where Alain Rapaport received me. Thank you very much, Alain, for all the hours of work together, your always good disposition and the advice. I learned a lot from your experience in academia. Many thanks to those who reviewed and made a report of my thesis, María Soledad Aronna and Jean-Baptiste Caillau. Thank you for accepting this work and for the quality comments and feedback you gave me. Thanks to Jaime Ortega and Emmanuel Trelat for being part of my jury. In particular thanks to Emmanuel for making the link between Pierre and me, helping to start the cotutelle.* No quiero dejar de agradecer en esta parte a Jorge Amaya y Cristopher Hermosilla con quienes realizamos una publicación que representó el inicio de mi trabajo doctoral, y además a Jorge por ser quien me presentó el problema de la minería y con quien me es siempre grato discutir sobre distintos problemas aplicados.

En el camino del doctorado, uno se encuentra con mucha gente que decidió tomar la misma vía, y son éstos quienes mejor entienden lo duro del proceso, por lo mismo el apoyo mutuo que nos brindamos sirve mucho para afrontar los periodos de estrés y ansiedad. Parto entonces por agradecer al lado chileno, Nico (gracias por subarrendarme el depa el tiempo de pasada en Chile), Marín, Laura, Jorge, Seba (las batallas de rap y las jornadas de trabajo con cumbia ayudaron a esto), Jessica (gracias por tomar la posta del seminario que dejamos con el Seba), Daniel, Yamit, Juan José, Álvaro, Hugo, Felipe bigotinni (Gracias por todas las pautas burocráticas que te sacaste, hasta ahora último incluso), Antoine, Francisco, Juan Pablo y Melanie.

Por el lado francés, parto por el team hispano hablante quienes nos adueñamos de la sala del café durante el periodo de confinamiento y del equipo CROUS de las 12.30, Agustín (de las primeras amistades en Francia, peleando fuertemente con el Seba al mejor compañero de oficina), Ramón (las partidas de age deben volver), Jesús, Nicolás (desde los 14 me sigues a donde voy) y Ana (párrafo aparte). Muchas gracias también al resto que crucé en mi estancia en el LJLL y que por culpa del covid faltó más tiempo para compartir, Lise (la primera del lab que conocí cuando pasé por la escondida oficina del segundo piso), Jules, Allen, Giorgia, Emma, Antoine, Chourouk, Alexiane, Alexandre, Remi, Matthieu, Anouk, Ludovic, Po-Yi, David, Eugenio, Iván y muchos más que pasaron por ahí que de seguro olvido.

Pero no solo los compañeros de doctorado son los que te sostienen en este camino, los exteriores también hacen parte de este apoyo. Gracias al team juegos de fines de semana por zoom, Alvarín, Javi, Seba Illanes, Seba Tapia, Dani y Chapa, sin ustedes mi encierro en Chile habría sido catastrófico. Gracias también a los que han estado desde hace tiempo dando apoyo, Sapu, Salinas, Chi, Hasson, Palma, Mauro

(por alojarme en cada ciudad que estás), Viyo y Telo (los fase 1, que tuvo que haber una pandemia para que retomaran el contacto). A los que me dejó el trote y aún siguen, Cristóbal (gracias Bender también por dejarme ser tu compañero de piso durante 2 meses), Pancho (compañeros de trote y luego de experiencia parisina haciendo doctorado), Cuneo, loco Victor (en su versión de entrenador) y Wolde (durante el doctorado salió PB en la media maratón de Buenos Aires, mi última gran temporada de entrenamiento y la hice contigo).

Mi pasada por París me dejó un grupete que espero siga con sus reuniones anuales, gracias La Muchachada por hacer tan grata mi estadía en Francia, creo que a pesar del covid, ese periodo con randonnées, pichangas, picnics y juntas es el que más he disfrutado en mi paso por el doctorado, muchas gracias Agustín, Ana, Suney (los fundadores), Nicolás, Ramón, Claudia, Jesús, Cossio y Paula. Por mi pasada en Montpellier, agradezco a Ahmad e Ismail con quienes compartí innumerables cafés y me enseñaron mucho sobre la cultura musulmana. Gracias también a Gisell y Amhed que han hecho más gratas mis frecuentes pasadas por Lyon en esta parte final del doctorado.

Punto a parte para agradecer a Ana (o Nany como a ella le gusta que le digan), que conocí en mi pasada por el LJLL en Francia y que a pesar (o gracias?) a la pandemia, entablamos una relación y ha estado junto a mí en gran parte de mi trabajo doctoral, muchas gracias por el apoyo jevita pinareña-habanera-parisina-lyonesa.

Para cerrar, muchas gracias a quienes son la parte más importante de mis logros alcanzados, mi familia. Gracias a mis padres, Javier, susto gigante que nos hizo pasar y casi se pierde este fin de doctorado, gracias por estar ahí siempre atento a cualquier necesidad que tengamos y mostrarse siempre dispuesto a ayudar al mundo; Patricia, mi gran partner y quien más sufre con mis ausencias de Chile, gracias por ser la incondicional y mi pilar emocional. Gracias a mis abuelos, Juan y Rosalia, quienes han sido un ejemplo de esfuerzo y trabajo duro que llevo siempre presente. Gracias a mis hermanos Sebita y Javierito, que aún en su papel de rudos, siempre están pendientes de mí así como yo estoy pendiente de ustedes. Gracias a mi tía Vero, que ha sido importante desde un principio y sigue siempre presente. Esta vez también mencionaré a todo el resto que me he dado cuenta ha estado y ayudado directa o indirectamente a forjarme y ha permitido que alcance este logro, gracias abuelita Menche, tía Sonia, tía Isa, tía Bea, tío Pez, watón Raúl, Daniela, Cristian, Sole, Vanesa, Francisca, Susi y Yuyo.

Finalmente, agradezco a todos y todas que me han ayudado con esto y que posiblemente olvidé.

**APPLICATION OF OPTIMAL CONTROL TECHNIQUES TO NATURAL SYSTEMS MANAGEMENT****Abstract**

Optimal control techniques have numerous applications in engineering and real world problems. This thesis is devoted to using these techniques in two contexts, mining and epidemiology, dividing this document in two respective parts.

In the first part related to mining, we work with the continuous formulation of the Open Pit Problem consisting of finding the optimal shape of an opencast mine representing its profile by a continuous function. Optimality in this context corresponds to maximizing the profit of mineral extraction. We introduce for the first time optimal control models of this problem. We present optimality conditions of solutions along with numerical experiments using local and global methods.

Another relevant problem in this context corresponds to the Sequential version of the Open Pit Problem, which consists of scheduling an extraction program over consecutive time frames (for example, a profile each 6 months), finding nested profiles maximizing a discounted profit. We proposed a novel semi-continuous model to obtain solutions of the sequential problem and we use it to present for the first time, to the best of our knowledge, numerical solutions of a three dimensional case (a possible real world mine) including the original Open Pit Problem (case with a single time-frame).

In the second part we deal with optimal control problems minimizing the maximum value of a state. This problematic was inspired by Covid-19, where hospitals and ICU beds were overcrowded due to a high amount of simultaneous infections. We present four different reformulations of this kind of optimal control problem as a Mayer one, each one having its pros and cons. We present the numerical performance of each formulation in an academic example and in a more realistic SIR model where the problem corresponds to minimizing the peak of infectious compartment with integral constraint in the control. With respect to the latter problem, we prove analytically that the structure of the optimal control is null-singular-null and we used it to assess numerical solutions.

**Keywords:** optimal control, natural system management, mining, epidemiology

---

**Résumé**

Les techniques de contrôle optimale ont plusieurs applications dans les domaines de l'ingénierie et les problèmes de la vie pratique. Cette thèse est consacrée à l'utilisation de ces techniques dans deux contextes, l'exploitation minière et l'épidémiologie, divisant ce document en deux parties respectives.

Dans la première partie relative à l'exploitation minière, nous travaillons avec la formulation continue du problème Final Open Pit consistant à trouver la forme optimale d'une mine à ciel ouvert représentant son profil par une fonction continue. L'optimalité dans ce contexte correspond à maximiser le bénéfice de l'extraction minière. Nous présentons pour la première fois des modèles de contrôle optimal pour ce problème. Nous présentons des conditions optimales de solutions avec des simulations numériques utilisant des méthodes locales et globales.

La version séquentielle du Final Open Pit est également très pertinente dans le contexte de cette thèse. Le problème consiste à planifier un programme d'extraction sur des périodes de temps consécutives (par exemple, un profil tous les 6 mois), en trouvant des profils imbriqués maximisant un bénéfice actualisé. Nous avons formulé un nouveau modèle semi-continu permettant d'obtenir des solutions du problème séquentiel et que nous utilisons pour proposer des solutions numériques dans un cas tridimensionnelle (une mine possible du monde réel) y compris le problème Open Pit original (cas avec une seule période de temps). Cela, à notre connaissance, n'avait jamais été fait auparavant. Dans la deuxième partie, nous traitons les problèmes de contrôle optimal minimisant la valeur maximale d'un état. Ce problème a été inspiré par le contexte sanitaire très difficile suite à la covid-19, où avoir de beaucoup infectés en même temps saturer les hôpitaux et les lits de soins intensifs. Nous présentons quatre reformulations différentes de ce type de problème comme un Mayer de contrôle optimal, ayant chacun ses avantages et ses inconvénients. Nous évaluons la performance numérique de chaque formulation dans un exemple académique et dans un modèle SIR plus réaliste où le problème correspond à minimiser le pic du compartiment infecté avec une contrainte intégrale dans le contrôle. En ce qui concerne ce dernier problème, nous prouvons analytiquement que la structure du contrôle optimal est null-singular-null et nous l'avons utilisé pour évaluer les solutions numériques.

**Mots clés :** contrôle optimal, gestion du système naturel, exploitation minière, épidémiologie

---

**Laboratoire Jacques-Louis Lions**

Sorbonne Université – Campus Pierre et Marie Curie – 4 place Jussieu – 75005 Paris – France





# Contents

<b>Funding</b>	<b>xi</b>
<b>Acknowledgements</b>	<b>xiii</b>
<b>Abstract</b>	<b>xv</b>
<b>Contents</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 General background in optimal control . . . . .	1
1.1.1 Pontryagin Maximum Principle and indirect shooting methods . . . . .	2
1.1.2 Direct transcription approach . . . . .	3
1.1.3 Hamilton-Jacobi-Bellman equation . . . . .	3
1.1.4 Bocop toolbox . . . . .	4
1.2 Dissertation Outline . . . . .	4
<b>I Mining context and the Open Pit problem</b>	<b>7</b>
<b>2 The Open Pit problem</b>	<b>9</b>
2.1 Integer programming formulation . . . . .	10
2.2 Continuous framework . . . . .	11
<b>3 Analysis of optimality conditions for 2D and 3D Final Open Pit</b>	<b>15</b>
3.1 Introduction . . . . .	15
3.2 The 2D open pit problem . . . . .	16
3.2.1 Statement of the problem . . . . .	16
3.2.2 Standing Assumptions . . . . .	18
3.2.3 Basics on state constrained optimal control . . . . .	18
3.2.4 Optimality conditions for the 2D open pit problem . . . . .	20
3.3 The 3D open pit problem . . . . .	23
3.3.1 Statement of the problem . . . . .	24
3.3.2 Standing Assumptions . . . . .	24
3.3.3 Optimality conditions . . . . .	24
3.4 Future work and final remarks . . . . .	27

<b>4 Numerical study of 2D and 3D Final and Sequential Open Pit</b>	<b>29</b>
4.1 Introduction . . . . .	29
4.2 Problem statement . . . . .	30
4.2.1 Continuous formulation . . . . .	30
4.2.2 Semi continuous formulation for SOP . . . . .	31
4.3 Analysis and optimality conditions for FOP . . . . .	35
4.3.1 Applying Pontryagin's Maximum Principle . . . . .	35
4.3.2 Inactive case: bang/singular control . . . . .	37
4.3.3 Active state constraint case . . . . .	38
4.3.4 Control structure summary . . . . .	38
4.4 Numerical simulations . . . . .	38
4.4.1 Numerical methods . . . . .	39
4.4.2 Final open pit (2D): global and local optimization . . . . .	39
4.4.3 Sequential Open Pit (2D and 3D): local optimization . . . . .	41
4.5 Conclusions . . . . .	44
4.6 Implementation details for the semi-continuous approach . . . . .	45
4.7 Additional examples for the final open pit - continuous formulation . . . . .	45
4.7.1 FOP with infinite capacity and constant slope . . . . .	45
4.7.2 FOP with infinite capacity and variable slope . . . . .	46
<b>II Problems inspired by Covid-19 peak reduction</b>	<b>49</b>
<b>5 Compartmental models</b>	<b>51</b>
5.1 SIR model . . . . .	51
5.2 A new compartmental model including vaccines . . . . .	52
<b>6 A Mayer formulation for peak minimization problems</b>	<b>57</b>
6.1 Introduction . . . . .	57
6.2 Problem and hypotheses . . . . .	58
6.3 Formulations with constraint . . . . .	59
6.4 Formulation without state constraints . . . . .	62
6.5 Numerical illustrations . . . . .	65
6.5.1 A particular class of dynamics . . . . .	65
6.5.2 Application to an epidemiological model . . . . .	68
6.6 Discussion and conclusions . . . . .	74
<b>7 A feedback strategy for peak minimization in SIR model</b>	<b>75</b>
7.1 Introduction . . . . .	75
7.2 Definitions and problem statement . . . . .	76
7.3 The NSN feedback . . . . .	77
7.4 Optimal strategy . . . . .	78
7.5 Numerical illustrations and discussion . . . . .	81
<b>8 Conclusion and perspectives</b>	<b>87</b>

---

<b>A Codes global optimization (BocopHJB)</b>	<b>89</b>
A.1 1D FOP - continuous formulation . . . . .	89
A.1.1 Problem definition . . . . .	89
A.1.2 Objective . . . . .	90
A.1.3 Dynamics . . . . .	91
A.1.4 Constraints . . . . .	92
A.1.5 Other functions . . . . .	94
<b>B Codes local optimization (Bocop)</b>	<b>97</b>
B.1 1D SOP - continuous formulation . . . . .	97
B.1.1 Problem definition . . . . .	97
B.1.2 Problem functions . . . . .	98
B.2 1D SOP - semicontinuous formulation . . . . .	101
B.2.1 Problem definition . . . . .	101
B.2.2 Problem functions . . . . .	102
B.3 Local optimization (bocop) for 2D SOP - semicontinuous formulation . . . . .	106
B.3.1 Problem definition . . . . .	106
B.3.2 Problem functions . . . . .	109
<b>Bibliography</b>	<b>117</b>



# Chapter 1

## Introduction

For decades optimal control theory has been an important tool to apply in real world problems and engineering. A main motivation to develop this theory came from aerospace engineering in the context of the aerospace race in the Cold War, to solve problems such as the minimization of the fuel or time spent by an aircraft going from an initial position to another one. Many other examples exist in domains of aeronautics, chemical processes, vehicles and transportation, energy, biological systems, etc. Some of them are showed in [73, 91, 96, 97].

This work consists of applying analytical as well as numerical optimal control techniques to two specific problems. The first one corresponds to the Open Pit problem, well-known in context of mining, but using a new continuous formulation in contrast to the classical binary formulation. The second one arises during the pandemic motivated by overcrowded hospitals due to many infected individuals at the same time, therefore, the idea was to use optimal control tools in order to minimize the peak of infections.

This introduction is devoted to presenting the main optimal control techniques used and at the end, a brief explanation of each chapter. We refer to Chapters 2 and 5 as introductions to Parts I and II respectively.

### 1.1 General background in optimal control

Consider a time interval  $[0, T]$ , and  $(x(t), u(t)) \in \mathbb{R}^n \times \mathbb{R}^m \forall t \in [0, T]$  denoting  $x$  for the state and  $u$  for the control. An optimal control problem in a Bolza form is written as follows:

$$(P_0) \left\{ \begin{array}{l} \text{Minimize} \quad \varphi(x(T)) + \int_0^T f^0(t, x(t), u(t)) dt \\ \text{over all} \quad x \in \mathcal{A} \subset \mathcal{C}^n[0, T] \text{ and } u \in \mathcal{U} \\ \text{satisfying} \quad \dot{x}(t) = f(t, x(t), u(t)) \text{ for a.e.} \\ \quad \quad \quad x(0) = x_0, x(T) \in E \end{array} \right.$$

where the set of admissible controls is usually  $\mathcal{U} = \{u : [0, T] \rightarrow U, u \in L^\infty\}$ . In this formulation, boundary conditions are represented by a fixed initial state  $x(0) = x_0$  and a final state  $x(T)$  belonging to a set  $E$  which typically correspond to  $\{y \in \mathbb{R}^n | c_1(y) = 0, c_2(y) \leq 0\}$  where  $c_1$  and  $c_2$  are continuously differentiable functions with values in  $\mathbb{R}^{k_1}$  and  $\mathbb{R}^{k_2}$  respectively.

Additionally, we can consider optimal control problems with constraints in the state written as

$$(t, x(t)) \in A.$$

We then say that it is an optimal control problem with state constraints. Typically, state constraints are represented by  $g(x(t)) \leq 0, t \in [0, T]$  with  $g$  regular enough. In this same way, we define an optimal control problem with mixed constraints when we add  $(P_0)$  to the expression:

$$(t, x(t), u(t)) \in A$$

We do not focus on existence results for our problems, however those interested can refer to [34, 35, 73, 96]. In the following sections we will briefly recall the principle of 3 main classes of methods in optimal control, furthermore, we present Bocop solver, used in numerical experiments of this thesis.

### 1.1.1 Pontryagin Maximum Principle and indirect shooting methods

After its discovery (see [50] for a review of the history), many versions of Maximum Principle can be found in the literature. Here we present one of them, but other versions are shown throughout this document when necessary. Firstly, we introduce the costate  $p$ , of the same dimension as the state  $x$  and define the Hamiltonian associated to  $(P_0)$  as

$$H(t, x, p, p^0, u) = p^0 f^0(t, x, p) + p^T f(t, x, u).$$

Let  $f^0, f, \varphi$  be  $\mathcal{C}^1$  class and  $u \in L^\infty$ , the Maximum Principle is stated as follow

**Theorem 1.1.1.** *Under previous assumptions, if the pair  $(x(t), u(t))$  is optimal for  $(P_0)$  then there exists an absolutely continuous function  $p : [0, T] \rightarrow \mathbb{R}^n$  and a real value  $p^0 \leq 0$  such that:*

1.  $(p(\cdot), p^0) \neq 0$ .
2.  $\dot{x}(t) = \partial_p H(t, x(t), p(t), p^0, u(t)), \quad -\dot{p}(t) = \partial_x H(t, x(t), p(t), p^0, u(t))$
3.  $H(t, x(t), p(t), p^0, u(t)) = \max_{v \in U} H(t, x(t), p(t), p^0, v)$
4.  $p(T) \in p^0 \nabla \varphi(x(T)) + N_E(x(T))$
5.  $\frac{d}{dt} H(t, x(t), p(t), p^0, u(t)) = \partial_t H(t, x(t), p(t), p^0, u(t))$

A proof of this theorem can be seen in [34, 73]. Note that this base version does not consider state constraints although these will appear in the Final Open Pit problem, therefore, we will present more general versions where appropriate.

Shooting method is based on the Pontryagin Maximum Principle and it is part of the indirect methods (for more detail see [10, 28, 34]). To explain this method, assume the normal case, i.e.,  $p^0 \neq 0$ . and free final conditions on  $x$ , that means,  $E = \mathbb{R}^n$ . The idea of the shooting method is to solve the boundary value problem derived from the Maximum Principle:

$$(BVP) \begin{cases} \dot{x}(t) = f(t, x(t), \bar{u}(t)) \\ \dot{p}(t) = -\partial_x H(t, x(t), p(t), p^0, \bar{u}(t)) \\ x(0) = x_0 \\ p(T) = \nabla \varphi(x(T)) \end{cases}$$

using the control  $\bar{u}$  obtained in point 3 of theorem 1.1.1 when it can be written depending explicitly on  $x(t), p(t)$ , i.e.  $\bar{u}(t) = \Psi(x(t), p(t))$  for a certain function  $\Psi$ . We introduce the shooting function associated

to this system as the functional that returns  $\bar{p}(T) - \nabla\varphi(\bar{x}(T))$  for a value  $z$ , where  $(\bar{x}, \bar{p})$  are solutions of the following Initial value problem

$$(IVP_z) \begin{cases} \dot{x}(t) = f(t, x(t), \Psi(x(t), p(t))) \\ \dot{p}(t) = -\partial_x H(t, x(t), p(t), p^0, \Psi(x(t), p(t))) \\ x(0) = x_0 \\ p(0) = z \end{cases} .$$

If we call  $S : z \rightarrow \bar{p}(T) - \nabla\varphi(\bar{x}(T))$  the shooting function, then, the shooting method consists in solving  $S(z) = 0$ , which is equivalent to solving the system (BVP) and therefore, obtaining a pair  $(x, p)$  fulfilling the conditions of theorem 1.1.1.

This idea can be generalized to a different set  $E$  and a more general optimal control problem, for instance, when  $x_0$  is not fix but belongs to a set  $F \subseteq \mathbb{R}^n$ . In a more general case state constraints can be considered but the generalization is far more difficult.

### 1.1.2 Direct transcription approach

The so-called direct approach transforms the infinite dimensional optimal control problem ( $P_0$ ) into a finite dimensional optimization one (typically non linear). The idea is to discretize the time interval  $[0, T]$  in  $\{t_0 = 0, \dots, t_N = t_f\}$  and apply it to the state and control variables, obtaining the finite dimensional variable  $X = \{x_0, \dots, x_N, u_0, \dots, u_{N-1}\}$ .

Several options exist to discretize the Ordinary Differential Equation (ODE) of  $x$ , for instance, using Euler explicit form and considering an equidistant discretization on time, where we obtain

$$x_{i+1} = x_i + hf(x_i, u_i).$$

Finally, we get the following nonlinear programming problem:

$$(NLP) \begin{cases} \min & \varphi(x_N) + \sum_{i=0}^{N-1} hf^0(t_i, x_i, u_i) \\ s.a & x_{i+1} = x_i + hf(x_i, u_i) & \forall i \in \{0, \dots, N-1\} \\ & u_i \in U & \forall i \in \{0, \dots, N-1\} \\ & x_0 = \bar{x}_0, x_N \in E \end{cases}$$

State and mixed constraints can be incorporated directly in each discretization point. These methods are widely used in industrial applications because they are more straightforward to apply than indirect methods. We refer the reader to [19] and [85] for more details on direct transcription methods and NLP algorithms.

### 1.1.3 Hamilton-Jacobi-Bellman equation

To introduce this results it is necessary to define the value function associated to ( $P_0$ ). Then, for  $t \in [0, T]$  and  $\xi \in \mathbb{R}^n$ , the value function  $V(t, \xi)$  is defined by:

$$V(t, \xi) = \min_{u \in \mathcal{U}} \left\{ \varphi(x(T)) + \int_t^T f^0(s, x(s), u(s)) ds \mid \dot{x}(s) = f(s, x(s), u(s)) \forall s \in [t, T]; x(t) = \xi \right\}.$$

It corresponds to the optimal value of ( $P_0$ ) taking the initial time and position as  $t$  and  $\xi$  respectively. The main result is then



**Theorem 1.1.2.** *The value function  $V(t, \xi)$  is a solution (in some specific sense) of the equation*

$$\begin{aligned} \partial_t V + \min_{v \in U} H(t, \xi, \partial_\xi V, -1, v) &= 0, \quad \forall (t, \xi) \in (0, T) \times \mathbb{R}^n \\ v(T, \xi) &= \varphi(\xi) \end{aligned}$$

An important tool in finding the value function is the Dynamic Programming Principle which corresponds to the following expression

$$V(t, \xi) = \min_{u \in \mathcal{U}} \left\{ \int_t^\tau f^0(s, u(s), y(s)) ds + V(y(\tau), \tau) \right\}. \quad (1.1)$$

Many methods mix the HJB equation and dynamic principle giving a global solution unlike direct transcription which is local, but the computational effort is normally higher.

For a complete introduction to the subject see [13, 92], considering state-constraints [2, 32] and the link with Maximum Principle [37]. This approach is not mainly used in this thesis except for numerical simulations using the HJB version of Bocop, the solver that we are going to present now.

### 1.1.4 Bocop toolbox

In most of the numerical simulation of this thesis we use Bocop, an open source toolbox for optimal control problems, developed by Inria and which can be downloaded directly from <https://www.bocop.org/>. On the official web-page there is the option to choose between two different packages.

The first and original package implements a local optimization method. The optimal control problem is approximated by a finite dimensional optimization problem (NLP) using a time discretization (the direct transcription approach, see section 1.1.2). The NLP problem is solved by the well known software Ipopt [98], using sparse exact derivatives computed by CppAD [18]<sup>1</sup>. In Figure 1.1 its interface is shown and it is possible to observe several options and functionalities of Bocop.

The second package BocopHJB implements a global optimization method. Similarly to the Dynamic Programming approach, the optimal control problem is solved in two steps. First we solve the Hamilton-Jacobi-Bellman equation satisfied by the value function of the problem. Then we simulate the optimal trajectory from any chosen initial condition. The computational effort is essentially taken by the first step, whose result, the value function, can be stored for subsequent trajectory simulations.<sup>2</sup> Figure 1.2 shows interface of BocopHJB.

Bocop runs under Linux, Mac and Windows, besides, in both interfaces it is possible to export solutions in a specified data format allowing its treatment in Matlab or Python. We also take advantage of their interpolation tools, mainly in the mining context, where we have discretized data to be incorporated in the continuous optimal control model. A well detailed user guide for both packages can be found in <https://www.bocop.org/download/>.

## 1.2 Dissertation Outline

In this section we present how the manuscript is organized and the main subjects of each chapter.

Part I of this thesis begins with Chapter 2 where we introduce the Open Pit problem in mining. The focus of this chapter is to show the classical binary programming formulation introduced by Johnson in 1968 and the newer formulation introduced by Alvarez et al. in 2011. The last one, which uses continuous functions to retrieve the profile of a mine, will be the seminal model for Chapters 3 and 4.

In Chapter 3 we reformulate the continuous model of Alvarez et al, looking for optimal profiles among absolutely continuous functions rather than continuous ones. This allowed us to write the Liptchitz

<sup>1</sup>Explication obtained from Bocop user guide in <https://www.bocop.org/download/>

<sup>2</sup>see footnote 1

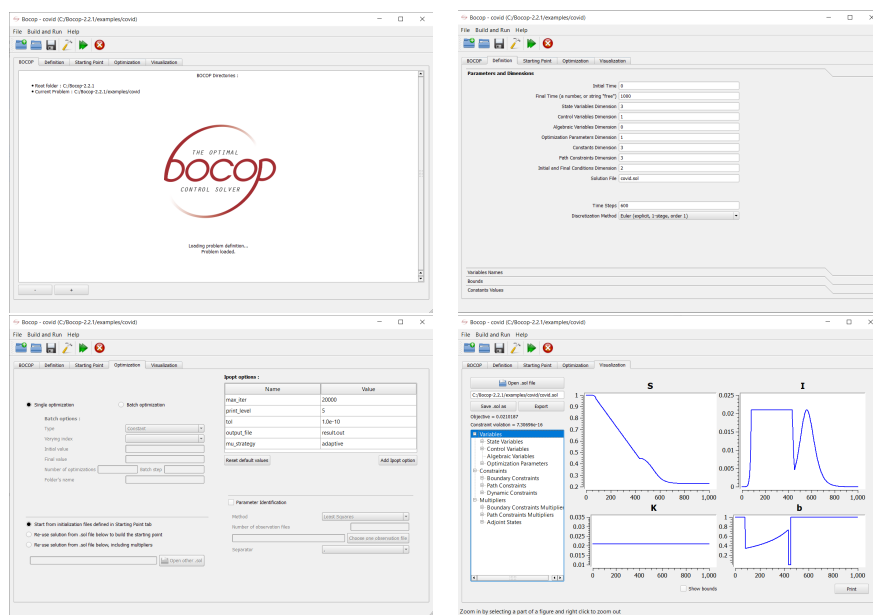


Figure 1.1: Bocop interface and some options

modulus  $L_p(x)$ , introduced in equation (2.1), as a derivative of  $p$ . Using this slight reduction in the functional space, we can use optimal control tools to obtain optimality conditions for profiles in 2D and 3D. This is the main result of this chapter and corresponds to theorems 3.2.1 and 3.3.1.

On the other hand, Chapter 4 focuses on a numerical study of the Open Pit problem and its Sequential version. In this chapter we propose a new semi-continuous formulation, presented in section 4.2.2, of the Sequential Open Pit problem. This formulation is based on a discretization of the space domain and then the profile is represented by a finite set of variables at the discretization nodes. The explicit models  $(SOP)_S^{2D}C$  and  $(SOP)_S^{3D}C$  are shown in section 4.2. We present in this chapter several numerical experiments using local and global methods (using Bocop and BocopHJB) for the two dimensional case and for the first time in literature, a numerical solution for a 3 dimensional case in a continuous framework using this new semi-continuous model (see section 4.4.3).

Moving to Part II, in Chapter 5 we introduce two compartmental models for epidemiological purposes. The first one corresponds to the classical SIR model, which we used in Chapters 6 and 7. The second one is a small contribution developed in the context of a OPS (Organización Panamericana de la salud) project where we set a compartmental model that considers vaccination and booster status (see section 5.2).

In Chapter 6 we work with optimal control problems consisted of minimizing the maximum of a state. We provide four alternative formulations to this problem in the Mayer form, two of which use state constraints, with one using mixed constraints and the other using differential inclusion. For the last one we formulated a family of more regular optimal control problems that approximate from below the optimal value. We compare the numerical performance of each one over both, an academic example and the minimization of peak of infectious in a SIR model with a  $L^1$  constraint on the control. We summarize advantages and drawbacks of the different formulations for numerical computations in Table 6.8 section 6.6.

Chapter 7 shows proof that the optimal control that minimizes the peak of  $I$ , with a  $L^1$  constraints in

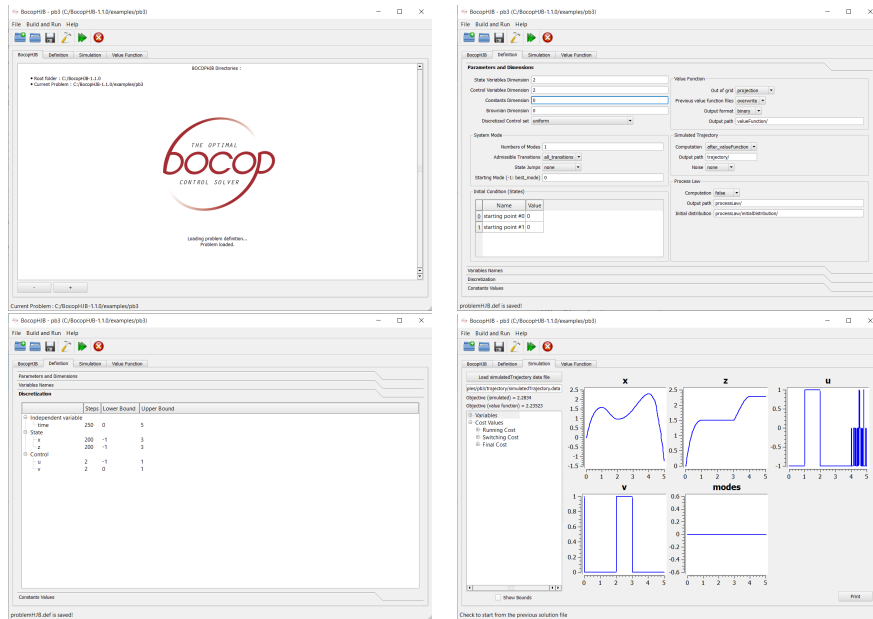


Figure 1.2: BocopHJB interface and some options

the control, has the structure null-singular-null showed in equation (6.5.2). We also compare this solution with the one obtained by Morris et al that impose just one interval of time where the intervention can happen.

Finally, in Chapter 8 we show conclusion and perspectives for each part of this manuscript.

## **Part I**

# **Mining context and the Open Pit problem**



## Chapter 2

# The Open Pit problem

Mining is among the main Chilean economic activities, according to the technical report *Anuario de la minería de Chile* prepared by Sernageomin [7], the governmental agency of geology and mining in Chile. It represented 12.5% of the CDP in 2020, mostly thanks to copper exports, of which Chile is the world's leading producer. Because of this, there exists in Chile a very active community of mining research which motivates the first part of this work.

The Final Open Pit problem (FOP) consists of finding the optimal shape of an open pit mine (in Figure 2.1 an example), in order to maximize the extraction profit while complying with a slope constraint which ensures the mine does not collapse. From this base problem, many other more complex ones can be set, for example, if we add a capacity in the number of blocks, the problem is called Capacitated Final Open Pit (CFOP). Another well studied problem considers the time periods for which we would like to obtain the shape of the mine, thus, obtaining an excavation plan. This problem is called Dynamical Final Open Pit (DFOP) which we also call Sequential Open Pit (SOP).



Figure 2.1: Chuquibambilla mine, the biggest open pit mine in the world located in Atacama region of Chile

We will present in the next section the classical formulation of the FOP, an integer programming prob-

lem based on a block model introduced in 1968 by Johnson in [61], and then the continuous framework introduced in 2011 by Alvarez et al. [3], which is the seminal paper of our work.

## 2.1 Integer programming formulation

Let  $\mathcal{B}$  be a set of blocks where for every  $i \in \mathcal{B}$  we know its benefit of extraction called  $b_i$ . To extract a block  $i$  it is necessary to take other blocks in order to respect the slope constraint. This information is stored as an arc in a direct graph  $(\mathcal{B}, A)$ , i.e.,  $(i, j) \in A$  if and only if to extract block  $i$  you have to take block  $j$ , in which case  $j$  is a predecessor of  $i$ . This is shown in Figure 2.2.

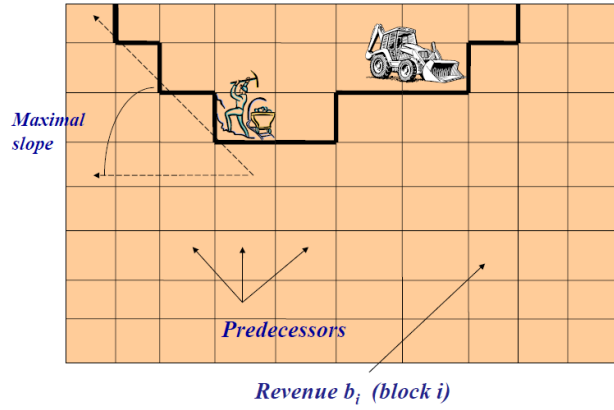


Figure 2.2: Block model example.

To set the optimization problem we define the following variables for each block  $i \in \mathcal{B}$  :

$$x_i = \begin{cases} 1 & \text{if block } i \text{ is chosen for extraction} \\ 0 & \text{if not} \end{cases}$$

therefore, the binary optimization problem is:

$$(FOP_0) \quad \sum_{i \in \mathcal{B}} b_i x_i$$

$$x_j - x_i \geq 0 \quad \forall (i, j) \in A$$

$$x_i \in \{0, 1\} \quad \forall i \in \mathcal{B}$$

adding a maximal capacity  $C$  of extraction in terms of total mass, the model can be modified as

$$(CFOP_0) \quad \max \sum_{i \in \mathcal{B}} b_i x_i$$

$$x_j - x_i \geq 0 \quad \forall (i, j) \in A$$

$$\sum_{i \in \mathcal{N}} p_i x_i \leq C$$

$$x_i \in \{0, 1\} \quad \forall i \in \mathcal{B}$$

where  $p_i$  is the density of block  $i$ . To pose the dynamical problem we consider  $T$  time periods. For each time  $t \in \{1, \dots, T\}$  and each block  $i \in \mathcal{B}$  we define the variable:

$$x_i^t = \begin{cases} 1 & \text{if block } i \text{ is extracted at time } t \\ 0 & \text{if not} \end{cases}.$$

So, for a discount factor  $\alpha \in (0, 1)$  the Dynamical Final Open Pit, named also by us as Sequential Open Pit (SOP), is written as follow

$$(SOP_0) \quad \max \sum_{t=1}^T \sum_{i \in \mathcal{B}} \frac{b_i}{(1+\alpha)^{t-1}} x_i^t$$

$$\begin{aligned} \sum_{t=1}^T x_i^t &\leq 1 & \forall i \in \mathcal{B} \\ \sum_{j=1}^t x_j^j - x_i^t &\geq 0 & \forall (i, j) \in A, t = 1, \dots, T \\ \sum_{i \in \mathcal{B}} p_i x_i^t &\leq C_t & t = 1, \dots, T \\ x_i^t &\in \{0, 1\} & \forall i \in \mathcal{B}, t = 1, \dots, T \end{aligned}.$$

These kinds of formulations have been studied for several years and many others constraints and conditions can be added (see for example [44, 60, 90]). A complete review of these models can be found in [84]. In the following section, we will present a newer and less exploited model, which is based of the first part of this thesis.

## 2.2 Continuous framework

The formulation presented in this section comes from [3] and we start by introducing the notation used. Let  $\Omega$  be the region of interest in  $\mathbb{R}^1$  or  $\mathbb{R}^2$ , supposed to be open and bounded. The border of a pit shall be determined by a continuous function  $p: \bar{\Omega} \rightarrow \mathbb{R}$  that is called profile and where  $p(x)$  represents the depth of the pit at point  $x \in \Omega$ .

The slope of the pit at point  $x \in \bar{\Omega}$  will be computed as the Lipschitz modulus  $L_p(x)$  defined as follows

$$L_p(x) := \limsup_{\bar{x} \rightarrow x \leftarrow \hat{x}} \frac{|p(\bar{x}) - p(\hat{x})|}{|\bar{x} - \hat{x}|} \quad (2.1)$$

and it will be bounded by a function  $w$  writing  $L_p(x) \leq w(x, p(x)), \forall x \in \bar{\Omega}$ . The natural shape of the ground will be represented by a continuous function  $p_0$ , therefore, each feasible profile has to be deeper than this initial one and it has to connect with  $p_0$  in the borders. That condition correspond to the following constraints:

$$\begin{aligned} p(x) - p_0(x) &\geq 0, \forall x \in \bar{\Omega} \\ p(x) - p_0(x) &= 0, \forall x \in \partial\Omega. \end{aligned}$$

In Figure 2.3 it is possible to observe the previous description. To finally set the optimization problem, the effort and gain density functions are defined as  $e(x, z) \geq e_o > 0$ ,  $g(x, z) \in \mathbb{R}$ ,  $\forall (x, z) \in \Omega \times Z$ , supposed



uniformly bounded, and the functionals

$$G[p, q] := \int_{\Omega} \int_{p(x)}^{q(x)} g(x, z) dz dx, \quad E[p, q] := \int_{\Omega} \int_{p(x)}^{q(x)} e(x, z) dz dx$$

$$G[q] := G[p_0, q] \quad E[q] := E[p_0, q]$$

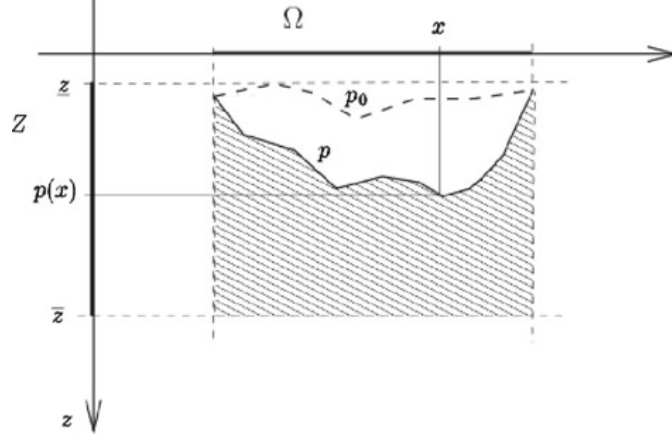


Figure 2.3: Explication image from [3].

The Final Open Pit problem in this continuous framework is then posed as:

$$(FOP_c) \quad \max G[p]$$

$$p(x) - p_0(x) \geq 0, \quad \forall x \in \overline{\Omega}$$

$$p(x) - p_0(x) = 0, \quad \forall x \in \partial\Omega$$

$$L_p(x) \leq w(x, p(x)) \quad \forall x \in \overline{\Omega}$$

$$p \in \mathcal{C}(\overline{\Omega})$$

and after adding the capacity constraint, the Capacitated Final Open Pit correspond to:

$$(CFOP_c) \quad \max G[p]$$

$$p(x) - p_0(x) \geq 0, \quad \forall x \in \overline{\Omega}$$

$$p(x) - p_0(x) = 0, \quad \forall x \in \partial\Omega .$$

$$L_p(x) \leq w(x, p(x)) \quad \forall x \in \overline{\Omega}$$

$$E[p] \leq C$$

$$p \in \mathcal{C}(\overline{\Omega})$$

To model the dynamical problem in [3] they consider a continuous interval of time  $[0, T]$  and for each time  $t \in [0, T]$ ,  $p(t, \cdot)$  is a feasible profile of  $(FOP_c)$ . Let also  $c \in L^\infty(0, T)$ ,  $c(t) \geq 0$  be the capacity at time  $t$  and

$$C(s, t) = \int_s^t c(\tau) d\tau$$

the total capacity in the interval  $[s, t] \subseteq [0, T]$ . Introducing a monotonically decreasing function  $\varphi \in \mathcal{C}^1[0, T]$  representing the discount factor, such that  $\varphi(0) = 1$  and  $0 < \varphi(T) < 1$  (typically  $\varphi(t) = e^{-\delta t}$ ,

for a given  $\delta > 0$ ), and noting  $P(t)(x) := p(t, x)$ , the present value of the gain is then:

$$\int_0^T \varphi(t) \int_{\Omega} g(x, p(t, x)) dx dP(t) = \int_{\Omega} \int_0^T \varphi(t) g(x, p(t, x)) \frac{\partial p}{\partial t}(t, x) dt dx.$$

Finally, the dynamical problem in the continuous framework can be written as:

$$(SOP_c) \quad \max \int_{\Omega} \int_0^T \varphi(t) g(x, p(t, x)) \frac{\partial p}{\partial t}(t, x) dt dx$$

$$\begin{aligned} p(t, x) &= p_0(x) && \forall x \in \partial\Omega, t \in [0, T] \\ L_{p(t, \cdot)}(x) &\leq w(x, p(t, x)) && \forall x \in \Omega, t \in [0, T] \\ p_0 &= p(0, \cdot) \leq p(s, \cdot) \leq p(t, \cdot) && \forall s, t \in [0, T], s \leq t \\ \int_{\Omega} \int_{p(s, x)}^{p(t, x)} e(x, z) dz dx &\leq C(s, t) && \forall s, t \in [0, T], s \leq t \\ p(t, \cdot) &\in \mathcal{C}^1(\overline{\Omega}) \end{aligned}$$

**Remark 2.2.1.** *The objective function of (SOP<sub>c</sub>) suggest that  $p(t, x)$  must be differentiable with respect to  $t$ , so, to avoid that requirement, it can be written in the following form (expression obtained integrating by parts):*

$$\varphi(T) \int_{\Omega} \int_{p(0, x)}^{p(T, x)} g(x, z) dz dx + \int_{\Omega} \int_0^T -\varphi'(t) \int_{p(0, x)}^{p(t, x)} g(x, z) dz dt dx$$

In this work the authors proved existence results for previous problems (see propositions 6 and 11 in [3]) but neither was any characterization of solutions nor numerical experiments shown. These two subjects are the novelty of Part I of this thesis.



## Chapter 3

# Analysis of optimality conditions for 2D and 3D Final Open Pit

This chapter corresponds to the article [4] entitled "Optimality conditions for the continuous model of the final open pit problem"

### 3.1 Introduction

The long term planning of a mine operation consists of defining a sequence for the extraction of material from the mine in order to maximize profit. As a first step in this process, decision-makers usually must decide the final pit limit, which corresponds to the identification of a maximum value on the total mass to be extracted from the site, which enables an upper bound on the discounted value of the profit over several periods to be defined. This first step is called the Final Open Pit or Ultimate Open Pit problem. A very early contribution to the practical resolution of this problem was proposed by Lerchs and Grossman [74] and, since then, a great variety of models and algorithms have been proposed. See Hustrulid et al [59] and Newman et al [84] for a more thorough introduction to open pit mine planning. The first effort to formally describe a practical mathematical model to solve this problem in an integrated way seems to be the work by Johnson [61].

Three different problems are usually considered for the economic valuation, design and planning of open pit mines. The first is the Final Open Pit (FOP) problem, which aims at finding the region of maximal economic value under geotechnical stability constraints. Another more realistic problem is what we call here the Capacity Final Open Pit (CFOP), which adds an additional constraint on the total capacity for extraction. The third problem is a multi-period version of the latter, which we call the Capacity Dynamic Open Pit (CDOP) problem, with the goal of finding an optimal sequence of volumes to be extracted with bounded capacities during each period.

The usual formulation of these problems consists of describing an ore reserve as a three-dimensional block model. Each block corresponds to a unitary volume of extraction, characterized by several physical and economic attributes, most of which are estimated from experimental sampling. Block models can be represented as directed graphs where nodes represent the blocks and arcs determine block precedence (order of extraction). Block precedence is essentially induced by operational constraints, such as those derived from slope stability. This discrete approach usually gives rise to huge, combinatorially large-scale instances of Integer Programming, such as that presented by Cacetta [29]. A great number of publications dealing with discrete block modeling for open pit mines have been published over the last 60 years. The seminal methodology for obtaining the ultimate pit limit, introduced by Lerchs and Grossman [74], has

been extensively applied in real mines for many years. The capacity dynamic problem is more difficult to solve and many methods using discrete optimization techniques have been proposed by Boland et al [22], Cacetta and Hill [30] and Hochbaum and Chen [58]. This problem is beyond the scope of this chapter, but we can mention some dynamic programming formulations, for instance, Johnson and Sharp [62] and Wright [101]. Metaheuristic and evolutionary algorithms have also been extensively tested by Denby and Schofield [39] and Ferland et al [47].

In this chapter we use an alternative approach to the above mentioned (CFOP) problem based on a continuous framework, proposed by Alvarez et al [3]. The basic idea is to describe pit contours by a continuous real-valued function, which maps each pair of horizontal coordinates to the corresponding vertical depth. Slope stability is ensured by means of a spatially distributed constraint on the local Lipschitz constant of the profile function. The maximal feasible local slope may vary throughout the site, depending on the geotechnical properties of the mineral deposit. The extraction capacity and operational costs are described by a possibly discontinuous effort density, a scalar function defined on the three-dimensional mining site. Concerning the continuous approach, we mention here the contribution by Ekeland and Queyranne [43], who proposed an alternative approach based on determining an optimum pit from an optimum dual solution of a particular transportation problem. Additionally, in [55], the authors derive duality results for the stationary open pit problem in the continuous framework, employing an additional condition called convex-likeness. The same authors, in [93], propose a partial differential equation model and show that, under suitable assumptions, the physically stable excavation path is the solution of a certain Hamilton-Jacobi equation.

The economic value of the blocks is given by a gain density defined on the deposit, which can also be a discontinuous function. Our goal here is to extend the existence results develop by Alvarez et al [3] to the qualitative properties of the optimal solutions. This qualitative characterization is derived from the optimality conditions in the calculus of variations and control theory.

The chapter is organized as follows. In Section 3.2 we describe the stationary problem in terms of continuous profile functions and we establish the basis of our approach, in the context of a "2D-mine", which permits to give a simple motivation of the real 3D problem and to derive relevant results that can be generalized to the real case. Section 3.3 is devoted to the study of the realistic 3D instance, extending the main results of the previous section. By using tools from the calculus of variations, we derive an operational characterization of the optimal profile, particularly to show that the gain function must take the value zero along the border of the optimal profile, unless the capacity or slope constraints are active. In Section 3.4 we briefly summarize the main contributions of this chapter and indicate some avenues for future research.

## 3.2 The 2D open pit problem

To fix ideas, we begin by considering the idealized case of an open pit on the plane, that is, the framework where the profiles are modeled using a continuous function that depends only on a single space variable (denoted  $x$  for simplicity). Generically, we denote a profile of an open pit by  $p : [a, b] \rightarrow \mathbb{R}_+$  where  $a$  and  $b$  are the extreme points of the open pit (there is no loss of generality in taking  $a < b$ ) and where  $p(x)$  represents the depth of the profile at the point  $x \in [a, b]$ ; see Figure 3.1.

### 3.2.1 Statement of the problem

For the sake of notation, we assume that the depth of a profile is always positive. In this framework, an admissible profile is a function  $p : [a, b] \rightarrow \mathbb{R}$  that must satisfy some conditions, the first one being as follows: given an initial profile  $p_0 : [a, b] \rightarrow \mathbb{R}$  an admissible profile has to satisfy

$$p_0(x) \leq p(x), \quad \forall x \in [a, b].$$

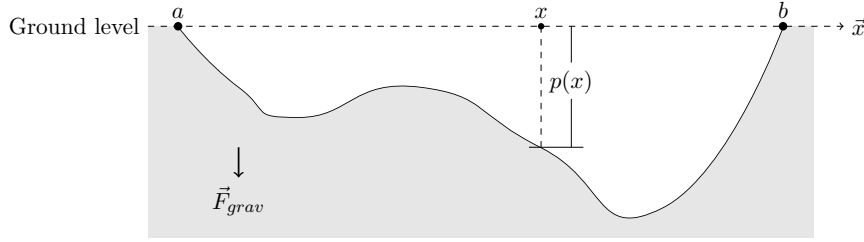


Figure 3.1: Profile of an open pit on the plane

which means that a feasible profile must be deeper than the initial profile  $p_0$ .

Given a profile  $p : [a, b] \rightarrow \mathbb{R}$ , we define its *slope* at the point  $x \in [a, b]$  as the *Lipschitz modulus* of  $p$  at  $x$  (see for example Dontchev and Rockafellar [42, Section 1D] ), that is,

$$L_p(x) := \limsup_{\bar{x} \rightarrow x \leftarrow \hat{x}} \frac{|p(\bar{x}) - p(\hat{x})|}{|\bar{x} - \hat{x}|}.$$

Due to the risk of landslides, the slope of a profile cannot be too steep. Note that the maximal slope allowed may change depending on the position and depth in the pit. This constraint is then represented via the condition

$$L_p(x) \leq \kappa(x, p(x)), \quad \forall x \in [a, b],$$

where  $\kappa(x, z)$  represents the maximal slope at the point  $(x, z)$  allowed for a profile  $p : [a, b] \rightarrow \mathbb{R}$  to be admissible. Note that if the profile is continuously differentiable on  $(a, b)$ , then the slope agrees with the absolute value of the profile's derivative (see [42, Section 1D] ), that is,

$$L_p(x) = |\dot{p}(x)|, \quad \forall x \in (a, b).$$

However, in our setting, working with smooth functions is too restrictive. For this reason we choose to work with a broader class of functions, namely, the collection of continuous functions whose derivatives exist almost everywhere on  $[a, b]$  and which satisfy

$$p(x) = p(a) + \int_a^x \dot{p}(s) ds, \quad \forall x \in [a, b].$$

This class of functions is the so-called set of absolutely continuous functions, which we denote by  $\mathcal{AC}[a, b]$ . It turns out that absolutely continuous functions are well behaved with respect to the slope, in the sense that the slope agrees almost everywhere with the derivative of an absolutely continuous profile.

**Lemma 3.2.1.** *Let  $p \in \mathcal{AC}[a, b]$ , then  $L_p(x) = |\dot{p}(x)|$  almost everywhere on  $[a, b]$ .*

On the other hand, due to physical or economic constraints, the capacity of extraction is indeed limited. Given a position  $x$ , the effort associated with extracting a block at depth  $z \geq p_0(x)$  can be represented by a nonnegative quantity  $e(x, z)$ . Thus, given a maximal budget  $c_{\max} > 0$ , the capacity constraints associated with a profile  $p : [a, b] \rightarrow \mathbb{R}$  can be expressed via the condition

$$\int_a^b \int_{p_0(x)}^{p(x)} e(x, z) dz dx \leq c_{\max}.$$

Concerning optimality, the marginal profit at each  $x \in [a, b]$  of an admissible profile  $p : [a, b] \rightarrow \mathbb{R}$  is

given by

$$\int_{p_0(x)}^{p(x)} g(x, z) dz$$

where  $g(x, z)$  represents the profit earned (or gain) for carrying out extraction at the block  $(x, z)$  for any  $z \in [p_0(x), p(x)]$ . Therefore, the total profit associated with an admissible profile  $p : [a, b] \rightarrow \mathbb{R}$  is given by

$$\int_a^b \int_{p_0(x)}^{p(x)} g(x, z) dz dx.$$

We are now in a position to formally state the 2D *Final Open Pit* problem:

$$\left\{ \begin{array}{l} \text{Maximize } \int_a^b \int_{p_0(x)}^{p(x)} g(x, z) dz dx \\ \text{over all } p \in \mathcal{AC}[a, b] \text{ subject to } p(a) = p_0(a), \quad p(b) = p_0(b) \\ p_0(x) \leq p(x), \quad \text{for all } x \in [a, b], \\ |\dot{p}(x)| \leq \kappa(x, p(x)) \quad \text{for a.e. } x \in [a, b] \\ \int_a^b \int_{p_0(x)}^{p(x)} e(x, z) dz dx \leq c_{\max}. \end{array} \right. \quad (\text{P}_{2\text{D}})$$

### 3.2.2 Standing Assumptions

Throughout the remainder of this section, unless otherwise stated, we will assume  $-\infty < a < b < +\infty$  and  $c_{\max} > 0$  are fixed parameters of the problem. The initial profile  $p_0 : [a, b] \rightarrow \mathbb{R}$  is a given continuously differentiable function.

The profit objective function  $g : [a, b] \times \mathbb{R}$  is assumed to be a nonnegative, bounded and piecewise continuous function. The marginal cost of extraction  $e : [a, b] \times \mathbb{R} \rightarrow \mathbb{R}$  is assumed to be a nonnegative, bounded and continuous function. Also, the maximal slope allowed  $\kappa : [a, b] \times \mathbb{R} \rightarrow \mathbb{R}$  is assumed to be continuous, nonnegative and bounded with  $p \mapsto \kappa(x, p)$  being continuously differentiable for any  $x \in [a, b]$  fixed and such that  $(x, q) \mapsto \nabla_q \kappa(x, q)$  is bounded on  $[a, b] \times \mathbb{R}^n$ .

Under these assumptions, the existence of an optimal profile is ensured, as proved by Alvarez et al [3]. Moreover, the fact that an optimal profile is absolutely continuous (Lipschitz continuous actually) is enforced by the boundedness and continuity of the maximal slope  $\kappa$ . This existence result concerns as well the 3D case studied in §3.3.

**Remark 3.2.1.** *In the light of Alvarez et al [3, Lemma 1], the feasible set of  $(\text{P}_{2\text{D}})$  without the capacity constraint, is convex provided  $z \mapsto \kappa(x, z)$  is concave for any  $x \in [a, b]$  fixed. Moreover, by [3, Proposition 5], if  $z \mapsto e(x, z)$  is monotonically increasing and  $z \mapsto g(x, z)$  is monotonically decreasing (for  $x \in [a, b]$  fixed), then the problem  $(\text{P}_{2\text{D}})$  turns out to be a convex one. The previous paragraph can also be applied to the 3D case. However, under the assumptions we have done so far, these hypotheses cannot be assured. As a matter of fact, the problem  $(\text{P}_{2\text{D}})$  may have several local minima, which are not necessarily global. It worths to mention then that in general setting of this manuscript we deal with non-convex problems. The previous comment also applies to the 3D case.*

### 3.2.3 Basics on state constrained optimal control

Let us point out that the formulation of  $(\text{P}_{2\text{D}})$  is slightly more restrictive than what has been treated in Alvarez et al [3]. Essentially, we restrict our analysis to a small class of functions, those that are absolutely continuous. The main advantage of doing so is that now the *Final Open Pit* problem can be treated as an optimal control problem with state constraints, and optimality conditions can be derived by fairly standard methods.

For the sake of completeness, we state the main tool from optimal control theory we are going to use in the analysis provided in this section. Let us consider a general Mayer optimal control problem on  $\mathbb{R}^n$ :

$$\left\{ \begin{array}{ll} \text{Minimize} & \varphi(q(b)) \\ \text{over all} & q \in \mathcal{AC}^n[a, b] \text{ and measurable functions } u \\ \text{satisfying} & \dot{q}(x) = f(x, q(x), u(x)), \text{ for a.e. } x \in [a, b], \\ & u(x) \in U, \text{ for a.e. } x \in [a, b], \\ & h(x, q(x)) \leq 0, \text{ for any } x \in [a, b], \\ & (q(a), q(b)) \in E. \end{array} \right. \quad (\text{P}_M)$$

Here,  $q \in \mathcal{AC}^n[a, b]$  means that  $q: [a, b] \rightarrow \mathbb{R}^n$  and if  $q = (q_1, \dots, q_n)$ , then each component  $q_i \in \mathcal{AC}[a, b]$ . Furthermore, for the purposes of our analysis, we only need to consider the case in which:

- $\varphi: \mathbb{R}^n \rightarrow \mathbb{R}$  is a continuously differentiable function,
- $f: [a, b] \times \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$  is such that,  $x \mapsto f(x, \hat{x}, \hat{u})$  is measurable,  $q \mapsto f(\hat{x}, q, \hat{u})$  is Lipschitz continuous (uniformly with respect to  $(\hat{x}, \hat{u})$ ) and  $u \mapsto f(\hat{x}, \hat{q}, u)$  is continuous for any  $(\hat{x}, \hat{q}, \hat{u}) \in [a, b] \times \mathbb{R}^n \times U$  fixed,
- $h: [a, b] \times \mathbb{R}^n \rightarrow \mathbb{R}$  is continuous, with  $q \mapsto h(\hat{x}, q)$  being differentiable for any  $\hat{x} \in [a, b]$  fixed and such that  $(x, q) \mapsto \nabla_q h(x, q)$  is continuous on  $[a, b] \times \mathbb{R}^n$ ,
- $U \subseteq \mathbb{R}^m$  is a given nonempty compact set,
- $E \subseteq \mathbb{R}^n \times \mathbb{R}^n$  is a nonempty closed convex set.

It is worth recalling that the (convex) normal cone to a set  $S \subseteq \mathbb{R}^k$  is defined by

$$N_S(s) := \{ \eta \in \mathbb{R}^k \mid \langle \eta, \tilde{s} - s \rangle \leq 0, \forall \tilde{s} \in S \}, \quad \forall s \in S.$$

In particular, given  $s_0 \in \mathbb{R}$ , we have

$$N_{(-\infty, s_0]}(s) = \begin{cases} \{0\} & \text{if } s < s_0 \\ [0, +\infty) & \text{if } s = s_0, \end{cases} \quad \text{and} \quad N_{\{s_0\}}(s_0) = \mathbb{R}.$$

**Definition 3.2.1.** An arc  $\bar{q} \in \mathcal{AC}^n[a, b]$  admissible for  $(\text{P}_M)$  is said to be a weak local minimizer of the problem (related to an optimal control  $\bar{u}$ ) if there is  $\varepsilon > 0$  such that

$$q \in \mathcal{AC}^n[a, b] \text{ is admissible for } (\text{P}_M) \text{ and } \|q - \bar{q}\|_{W^{1,1}} \leq \varepsilon \implies \varphi(\bar{q}(b)) \leq \varphi(q(b)).$$

Here  $\|\cdot\|_{W^{1,1}}$  stands for the usual norm of the Sobolev space  $W^{1,1}([a, b]; \mathbb{R}^n)$ .

It turns out that, in this setting, weak local minimizers of  $(\text{P}_M)$  satisfy Maximum Principle for State Constrained problems (Vinter [97, Theorem 9.3.1]).

**Lemma 3.2.2.** Under the conditions stated above, if  $\bar{q} \in \mathcal{AC}^n[a, b]$  is a weak local minimizer of  $(\text{P}_M)$  related to the optimal control  $\bar{u}$ , then there exist  $\lambda \in \mathcal{AC}^n[a, b]$ ,  $\eta \in \{0, 1\}$ , a (positive) Radon measure  $\mu$  on  $[a, b]$ , and a Borel measurable function  $\gamma: [a, b] \rightarrow \mathbb{R}^n$  satisfying

$$\gamma(x) = \nabla_q h(x, \bar{q}(x)), \quad \text{for } \mu\text{-a.e. } x \in [a, b],$$

such that



1.  $(\lambda, \mu, \eta) \neq (0, 0, 0)$ ;
2.  $-\dot{\lambda}(x) \in \partial_q^c H(x, \bar{q}(x), \xi(x), \bar{u}(x))$  for a.e.  $x \in [a, b]$ ;
3.  $(\lambda(a), -\xi(b)) \in \{0\} \times \{\eta \nabla \varphi(\bar{q}(b))\} + N_E(\bar{q}(a), \bar{q}(b))$ ;
4.  $H(x, \bar{q}(x), \xi(x), \bar{u}(x)) = \max_{u \in U} H(x, \bar{q}(x), \xi(x), u)$  for a.e.  $x \in [a, b]$ ;
5.  $\text{supp}(\mu) \subseteq \{x \in [a, b] \mid h(x, \bar{q}(x)) = 0\}$ .

Here  $H(x, q, \xi, u) = \langle \xi, f(x, q, u) \rangle$  and

$$\xi(x) = \lambda(x) + \int_{[a, x[} \gamma(s) \mu(ds) \quad \forall x \in [a, b[ \quad \text{and} \quad \xi(b) = \lambda(b) + \int_{[a, b]} \gamma(s) \mu(ds)$$

### 3.2.4 Optimality conditions for the 2D open pit problem

In this part of the chapter, we analyze the behavior of an optimal profile by using the tools from optimal control theory described earlier. The following result can in principle be stated for local optima as well. However, to keep the presentation of the chapter simple, we prefer to present it only for a global optimum.

**Theorem 3.2.1.** *Let  $\bar{p} \in \mathcal{AC}[a, b]$  be an optimal profile of the problem  $(P_{2D})$ . Then there are  $\zeta \in \mathcal{AC}[a, b]$ ,  $\eta \in \{0, 1\}$ ,  $\bar{\lambda} \leq 0$  and a (positive) Radon measure  $\mu$  on  $[a, b]$ , with at least one of them not equal to zero, such that*

$$-\dot{\zeta}(x) \in \eta G(x, \bar{p}(x)) + \bar{\lambda} e(x, \bar{p}(x)) + |\mu([a, x[) - \zeta(x)| \partial_p \kappa(x, \bar{p}(x)), \quad \text{a.e. on } [a, b],$$

with  $\text{supp}(\mu) \subseteq \{x \in [a, b] \mid p_0(x) = \bar{p}(x)\}$ , and where

$$G(x, p) = \text{co} \{g(x, p^-), g(x, p^+)\}, \quad \forall x \in [a, b], \forall p \in \mathbb{R}.$$

Furthermore, we also have

$$\bar{\lambda} \left( \int_a^b \int_{p_0(x)}^{\bar{p}(x)} e(x, z) dz dx - c_{\max} \right) = 0$$

and  $(\zeta(x) - \mu([a, x[))(|\bar{p}(x)| - \kappa(x, \bar{p}(x))) = 0$  for a.e.  $x \in [a, b]$ .

*Proof.* The proof of the result is based on a transformation of the *Final Open pit* problem  $(P_{2D})$  into a Mayer problem such as  $(P_M)$ , for which  $\bar{p}$  provides a weak local minimizer related to the optimal control

$$\bar{u}(x) := \begin{cases} \frac{\dot{\bar{p}}(x)}{\kappa(x, \bar{p}(x))} & \text{if } \kappa(x, \bar{p}(x)) \neq 0 \\ 0 & \text{otherwise.} \end{cases} \quad (3.1)$$

We divide the proof into several parts for the sake of exposition.

1. First we show that  $(P_{2D})$  is an instance of the Mayer problem  $(P_M)$ . The key points here are to interpret the slope condition as a controlled ordinary differential equation and to be able to handle the capacity constraints

$$\int_a^b \int_{p_0(x)}^{\bar{p}(x)} e(x, z) dz dx \leq c_{\max} \quad (3.2)$$

as an end-point constraint of an additional state. Let  $p \in \mathcal{AC}[a, b]$  be a given profile. On the one hand, note that for any  $x \in [a, b]$  such that  $\kappa(x, p(x)) \neq 0$ , the condition  $|\dot{p}(x)| \leq \kappa(x, p(x))$  is equivalent to  $-1 \leq u(x) := \frac{\dot{p}(x)}{\kappa(x, p(x))} \leq 1$ . This implies then that the condition  $|\dot{p}(x)| \leq \kappa(x, p(x))$  is actually equivalent to

$$\dot{p}(x) = u(x)\kappa(x, p(x)), \quad \text{with } -1 \leq u(x) \leq 1, \quad \text{for a.e. } x \in [a, b].$$

On the other hand, note that (3.2) is actually an isoperimetric inequality constraint. To deal with it, we introduce a new auxiliary state. Let  $q_1 : [a, b] \rightarrow \mathbb{R}$  be given by

$$q_1(t) = \int_a^t \int_{p_0(x)}^{p(x)} e(x, z) dz dx, \quad \forall t \in [a, b].$$

Thus, it is clear that (3.2) can be written as  $q_1(b) \leq c_{\max}$ . Furthermore,  $q_1(a) = 0$  and the velocity of  $q_1$  is given by the expression

$$\dot{q}_1(x) = \int_{p_0(x)}^{p(x)} e(x, z) dz, \quad \text{for a.e. } x \in [a, b].$$

Also, by defining  $q_2 : [a, b] \rightarrow \mathbb{R}$  via the formula

$$q_2(t) = \int_a^t \int_{p_0(x)}^{p(x)} g(x, z) dz dx, \quad \forall t \in [a, b],$$

it is clear that the total profit is given by  $q_2(b)$ , and that this new state satisfies

$$\dot{q}_2(x) = \int_{p_0(x)}^{p(x)} g(x, z) dz, \quad \text{for a.e. } x \in [a, b], \quad \text{with } q_2(a) = 0.$$

Therefore, setting  $q(x) = (q_1(x), q_2(x), p(x))$  for any  $x \in [a, b]$ , we see that  $(P_{2D})$  is an instance of the Mayer problem  $(P_M)$  with  $\varphi(q) = -q_2$ ,  $h(x, q) = p_0(x) - q_3$ ,  $U = [-1, 1]$ ,

$$f(x, q, u) = \left( \int_{p_0(x)}^{q_3} e(x, z) dz, \int_{p_0(x)}^{q_3} g(x, z) dz, u\kappa(x, q_3) \right)$$

and

$$E = \{(\alpha, \beta) \in \mathbb{R}^3 \times \mathbb{R}^3 \mid \alpha_1 = \alpha_2 = 0, \alpha_3 = p_0(a), \beta_1 \leq c_{\max} \text{ and } \beta_3 = p_0(b)\}.$$

2. Now, since  $\bar{p}$  is assumed to be an optimal solution of  $(P_{2D})$ , it follows that  $\bar{p}$  provides a weak local minimizer of  $(P_M)$ , related to the optimal control defined in (3.1).

Moreover, the condition under which Lemma 3.2.2 has been stated are satisfied by the data provided in the preceding part; the Lipschitz continuity of  $q \mapsto f(\hat{x}, q, \hat{u})$  (uniformly with respect to  $(\hat{x}, \hat{u}) \in [a, b] \times U$ ) comes from the fact that  $e$  and  $g$  are measurable bounded functions and  $\kappa$  is Lipschitz continuous in the second variable, uniformly with respect to the first one. Therefore, we can apply Lemma 3.2.2, and so, there exist  $\lambda \in \mathcal{AC}^3[a, b]$ ,  $\eta \in \{0, 1\}$ , a (positive) Radon measure  $\mu$  on  $[a, b]$ , and a Borel measurable function  $\gamma : [a, b] \rightarrow \mathbb{R}^3$  fulfilling the conditions in Lemma 3.2.2. Note first that the Hamiltonian does not depend on  $q_1$  nor on  $q_2$ , and also that

$$\nabla_q h(x, q) = (0, 0, -1).$$

Because of point 2 in Lemma 3.2.2 and the definition of  $x \mapsto \xi(x)$ , we can deduce that there are  $\bar{\lambda}_1, \bar{\lambda}_2 \in \mathbb{R}$  such that

$$\xi_1(x) = \lambda_1(x) = \bar{\lambda}_1 \quad \text{and} \quad \xi_2(x) = \lambda_2(x) = \bar{\lambda}_2, \quad \forall x \in [a, b],$$

and

$$\xi_3(x) = \lambda_3(x) - \mu([a, x]), \quad \forall x \in [a, b[, \quad \text{and} \quad \xi_3(b) = \lambda_3(b) - \mu([a, b]).$$

Note also that  $\nabla \varphi(q) = (0, -1, 0)$  and

$$N_E(\bar{q}(a), \bar{q}(b)) = \mathbb{R}^3 \times \{\beta \in \mathbb{R}^3 \mid \beta_1 \geq 0, \beta_1(\bar{q}_1(b) - c_{\max}) = 0 \text{ and } \beta_2 = 0\}.$$

By point 3 in Lemma 3.2.2, we have that  $\bar{\lambda}_2 = \eta \in \{0, 1\}$  and  $\bar{\lambda}_1 \leq 0$  with

$$\bar{\lambda}_1 \left( \int_a^b \int_{p_0(x)}^{\bar{p}(x)} e(x, z) dz dx - c_{\max} \right) = 0.$$

By point 4 in Lemma 3.2.2 we have, since  $\kappa$  is nonnegative, that

$$\xi_3(x) \bar{u}(x) \kappa(x, \bar{p}(x)) = |\xi_3(x)| \kappa(x, \bar{p}(x)), \quad \text{for a.e. } x \in [a, b].$$

Note that whenever  $\kappa(x, \bar{p}(x)) \neq 0$  (a.e. on  $[a, b]$ ) we have that

$$\xi_3(x) \dot{\bar{p}}(x) = |\xi_3(x)| \kappa(x, \bar{p}(x)).$$

This implies that whenever  $\dot{\bar{p}}(x) < \kappa(x, \bar{p}(x))$ , then necessarily  $\xi_3(x) = 0$ , and so

$$\xi_3(x) (\dot{\bar{p}}(x) - \kappa(x, \bar{p}(x))), \quad \text{for a.e. } x \in [a, b].$$

Finally, note that since  $e$  is continuous and  $\kappa$  is continuously differentiable in the second variable, for any  $x \in [a, b]$   $\xi \in \mathbb{R}^3$  and  $u \in [-1, 1]$  fixed, such that  $p \mapsto q(x, p)$  is continuous at  $p = q_3$ , we have

$$\nabla_q H(x, q, \xi, u) = (0, 0, \xi_1 e(x, q_3) + \xi_2 g(x, q_3) + \xi_3 u \partial_p \kappa(x, q_3))$$

In particular, by point 2 in Lemma 3.2.2, for a.e.  $x \in [a, b]$  such that  $p \mapsto q(x, p)$  is continuous at  $p = \bar{p}(x)$  we have

$$-\dot{\lambda}_3(x) = \bar{\lambda}_1 e(x, \bar{p}(x)) + \eta g(x, \bar{p}(x)) + \xi_3(x) \bar{u}(x) \partial_p \kappa(x, \bar{p}(x))$$

because in this case  $q \mapsto H(x, q, \xi(x), \bar{u}(x))$  is continuously differentiable at

$$q = \left( \int_a^x \int_{p_0(\bar{x})}^{p(\bar{x})} e(\bar{x}, z) dz d\bar{x}, \int_a^x \int_{p_0(\bar{x})}^{p(\bar{x})} g(\bar{x}, z) dz d\bar{x}, \bar{p}(x) \right).$$

Since the functions  $g$  is piecewise continuous, for any  $x \in [a, b]$ , if  $p \mapsto q(x, p)$  is not continuous at  $p = \bar{p}(x)$  we have that

$$\partial_q^C H(x, q, \xi, u) = \{(0, 0)\} \times (\xi_1 e(x, q_3) + \xi_2 G(x, q_3) + \xi_3 u \partial_p \kappa(x, q_3))$$

where

$$G(x, p) = \text{co}\{g(x, p^-), g(x, p^+)\}, \quad \forall x \in [a, b], p \in \mathbb{R}.$$

Also, on the one hand, by Maximum Principle (point 4 in Lemma 3.2.2), for a.e.  $x \in [a, b]$  such that  $\kappa(x, \bar{p}(x)) > 0$  we must have that  $\xi_3(x)\bar{u}(x) = |\xi_3(x)|$ . On the other hand, if  $\kappa(x, \bar{p}(x)) = 0$ , we must have that  $\partial_p \kappa(x, \bar{p}(x)) = 0$  because  $p = \bar{p}(x)$  is a local minimum of  $p \mapsto \kappa(x, p)$ . Combining these two issues we get

$$\xi_3(x)\bar{u}(x)\partial_p \kappa(x, \bar{p}(x)) = |\xi_3(x)|\partial_p \kappa(x, \bar{p}(x)), \quad \text{for a.e. } x \in [a, b].$$

Therefore, setting  $\bar{\lambda} = \bar{\lambda}_1$  and  $\zeta = \lambda_3$  the conclusion follows.  $\square$

We now state a direct consequence of the preceding theorem in the case when the slope condition is not active, and the state constraint is only active at the end-points.

**Corollary 3.2.1.** *Let  $\bar{p} \in \mathcal{AC}[a, b]$  be an optimal profile of the problem (P<sub>2D</sub>). Suppose that*

$$p_0(x) < \bar{p}(x), \quad \forall x \in ]a, b[ \quad \text{and} \quad |\dot{\bar{p}}(x)| < \kappa(x, \bar{p}(x)), \quad \text{for a.e. } x \in [a, b]. \quad (3.3)$$

Then there are  $\eta \in \{0, 1\}$  and  $\bar{\lambda} \leq 0$ , such that

$$0 \in \eta G(x, \bar{p}(x)) + \bar{\lambda} e(x, \bar{p}(x)), \quad \text{a.e. on } [a, b].$$

with

$$\bar{\lambda} \left( \int_a^b \int_{p_0(x)}^{\bar{p}(x)} e(x, z) dz dx - c_{\max} \right) = 0.$$

In particular, if  $p \mapsto g(x, p)$  is continuous for any  $x \in [a, b]$  fixed, then the condition reduces to

$$\eta g(x, \bar{p}(x)) + \bar{\lambda} e(x, \bar{p}(x)) = 0, \quad \forall x \in [a, b].$$

Moreover,

1. if the marginal cost associated with extracting a block at any depth is zero (there is no capacity constraint), that is,  $e(x, z) = 0$  for any  $x \in [a, b]$  and  $z \geq p_0(x)$ , then the marginal gain of extracting a block at any depth must be zero on the subsection  $[a, b]$ , that is,

$$g(x, \bar{p}(x)) = 0, \quad \forall x \in [a, b].$$

2. if the marginal cost associated with extracting a block at any depth is positive (there is an effective capacity constraint), that is,  $e(x, z) > 0$  for any  $x \in [a, b]$  and  $z \geq p_0(x)$ , then  $\eta = 1$  and

$$g(x, \bar{p}(x)) + \bar{\lambda} e(x, \bar{p}(x)) = 0, \quad \forall x \in [a, b].$$

*Proof.* It is enough to apply directly Theorem 3.2.1, and check that  $\zeta(x) = \mu([a, x])$  for a.e.  $x \in [a, b]$  and note that  $\mu([a, x]) = 0$  for a.e.  $x \in [a, b]$  because  $\text{supp}(\mu) \subseteq \{a, b\}$ .  $\square$   $\square$

### 3.3 The 3D open pit problem

We now turn into the more realistic case of an open pit in the 3D space. The profiles in this framework are modeled using a continuous function that depends on the two horizontal space variable (denoted  $x$  and  $y$  for simplicity). Generically, we denote a profile of an open pit by  $p : \Omega \rightarrow \mathbb{R}$  where  $\Omega \subseteq \mathbb{R}^2$  is the bounded domain in  $\mathbb{R}^2$  that represents the open pit and where  $p(x, y)$  represents the depth of the profile at the point  $(x, y) \in \Omega$ .

### 3.3.1 Statement of the problem

As done for the 2D case, we assume that the depth of a profile is always positive. The final open pit problem in the 3D case has the same structure as in the 2D case. This means that for a given initial profile  $p_0 : \overline{\Omega} \rightarrow \mathbb{R}$ , the total profit and total extraction associated with an admissible profile  $p : \overline{\Omega} \rightarrow \mathbb{R}$  are given respectively by

$$\int_{\Omega} \int_{p_0(x,y)}^{p(x,y)} g(x,y,z) dz dx dy \quad \text{and} \quad \int_{\Omega} \int_{p_0(x,y)}^{p(x,y)} e(x,y,z) dz dx dy.$$

The maximal slope allowed is also considered to be bounded, and thus profiles are Lipschitz continuous mappings. The associated constraint is then represented via the condition

$$L_p(x,y) := \limsup_{(\bar{x},\bar{y}) \rightarrow (x,y) \leftarrow (\hat{x},\hat{y})} \frac{|p(\bar{x},\bar{y}) - p(\hat{x},\hat{y})|}{\sqrt{|\bar{x} - \hat{x}|^2 + |\bar{y} - \hat{y}|^2}} \leq \kappa(x,y,p(x,y)), \quad \forall (x,y) \in \Omega.$$

Therefore, the *Final Open Pit* problem in the 3D case is the following:

$$\left\{ \begin{array}{l} \text{Maximize} \quad \int_{\Omega} \int_{p_0(x,y)}^{p(x,y)} g(x,y,z) dz dx dy \\ \text{over all} \quad p \in \text{Lip}(\overline{\Omega}) \\ \text{subject to} \quad p(x,y) = p_0(x,y), \quad \text{for any } (x,y) \in \partial\Omega \\ \quad \quad \quad p_0(x,y) \leq p(x,y), \quad \text{for any } (x,y) \in \Omega, \\ \quad \quad \quad L_p(x,y) \leq \kappa(x,y,p(x,y)) \quad \text{for any } (x,y) \in \Omega \\ \quad \quad \quad \int_{\Omega} \int_{p_0(x,y)}^{p(x,y)} e(x,y,z) dz dx dy \leq c_{\max}. \end{array} \right. \quad (\text{P}_{3\text{D}})$$

**Remark 3.3.1.** *A more general model that considers profiles having a time dependance has been studied by Álvarez et al in [3]. The analysis of this problem, called Capacitated Dynamic Open Pit, becomes more difficult and we plan to study it in details elsewhere.*

**Remark 3.3.2.** *Similarly as for the 2D case, a control setting can be introduced to deal with the 3D case; see the proof of Theorem 3.2.1. This is certainly a suitable approach to handle Theorem 3.3.1, however, in this setting the control is distributed and also subject to constraints; see the discussion in §3.4. Moreover, optimality conditions for problems of this kind are known to be harder to handle and for this reason we take another path to prove Theorem 3.3.1 base on classical calculus of variations.*

### 3.3.2 Standing Assumptions

Throughout the remainder of this section, unless otherwise stated, we will assume  $\Omega \subseteq \mathbb{R}^2$  is an open bounded domain and  $c_{\max} > 0$  is fixed parameter of the problem. The initial profile  $p_0 : \overline{\Omega} \rightarrow \mathbb{R}$  is a given continuously differentiable function.

The densities of gain and effort are now  $g : \Omega \times \mathbb{R} \rightarrow \mathbb{R}$  and  $e : \Omega \times \mathbb{R} \rightarrow \mathbb{R}$ , respectively. They are assumed to be bounded, measurable and the second one (the densities of effort) nonnegative. Also, the maximal slope allowed  $\kappa : \Omega \times \mathbb{R} \rightarrow \mathbb{R}$  is assumed to be continuous, nonnegative and bounded with  $p \mapsto \kappa(x,p)$  being continuously differentiable for any  $x \in \Omega$  fixed and such that  $(x,q) \mapsto \partial_p \kappa(x,p)$  is bounded on  $\Omega \times \mathbb{R}^n$ .

### 3.3.3 Optimality conditions

We now present some necessary optimality conditions that extend the one given for the 2D case. The conditions obtained in this case do not require the continuity of the gain function  $g$  nor the continuity

of the effort  $e$ . However, because of the nonholonomic character of the slope constraints, the result we present is only valid for the case when optimal profiles do not saturate this condition (see assumption (3.4) below). Nonholonomic constraints are hard to handle in calculus of variations of multiple integral and require technical assumptions which may be too strong for the scope of this chapter. The main difficulty is that the construction of suitable *variations* is not always ensured; see for instance [52, Chapter 2]. It remains then as an open problem and future work to provide necessary optimality conditions for the general case where nonholonomic restriction may be active.

**Theorem 3.3.1.** *Let  $\bar{p} \in \text{Lip}(\bar{\Omega})$  be an optimal profile of  $(P_{3D})$ . Assume that  $\bar{p} \neq p_0$  and let  $\Omega_0 \subseteq \Omega$  be the open domain of  $\mathbb{R}^2$  given by*

$$\Omega_0 = \{(x, y) \in \mathbb{R}^2 \mid p_0(x, y) < \bar{p}(x, y)\}.$$

*If the slope constraints is not active on  $\Omega_0$ , that is,*

$$\sup_{(x, y) \in \Omega_0} \{L_{\bar{p}}(x, y) - \kappa(x, y, \bar{p}(x, y))\} < 0, \quad (3.4)$$

*then there is  $\bar{\lambda} \leq 0$  such that*

$$g(x, y, \bar{p}(x, y)) + \bar{\lambda}e(x, y, \bar{p}(x, y)) = 0, \text{ ct } p. \text{ in } \Omega_0.$$

*Furthermore,  $\bar{\lambda}$  satisfies the following properties:*

1. *If  $\int_{\Omega} \int_{p_0(x, y)}^{\bar{p}(x, y)} e(x, y, z) dz dx dy < c_{\max}$  then  $\bar{\lambda} = 0$ .*
2.  *$\bar{\lambda}$  can be taken to be any value  $\lambda = - \int_{\Omega} g(x, y, \bar{p}(x, y)) \psi(x, y) dx dy$ , provided that  $\psi \in \mathcal{C}_0^\infty(\Omega)$  is such that  $\int_{\Omega} e(x, y, \bar{p}(x, y)) \psi(x, y) dx dy = 1$ .*

*Proof.* The proof follows rather standard arguments in calculus of variations, adapted to be able to handle the integral inequality constraint of isoperimetric type. Assume first that  $e(x, y, \bar{p}(x, y))$  is not identically zero in  $\Omega_0$ . Take some  $\psi \in \mathcal{C}_0^\infty(\Omega_0)$  such that

$$\int_{\Omega_0} e(x, y, \bar{p}(x, y)) \psi(x, y) dx dy = 1.$$

Since  $e(x, y, \bar{p}(x, y)) \geq 0$  in  $(x, y) \in \Omega_0$  and it is not identically zero, the existence of such function  $\psi$  is guaranteed. Now take  $\varphi \in \mathcal{C}_0^\infty(\Omega_0)$  arbitrary and define for  $s, t \in \mathbb{R}$  the profile  $p_{s, t} \in \text{Lip}(\bar{\Omega})$  given by  $p_{s, t}(x, y) = \bar{p}(x, y) + s\varphi(x, y) + t\psi(x, y)$ . Consider the functions  $(s, t) \mapsto f(s, t)$  and  $(s, t) \mapsto h(s, t)$  defined on  $\mathbb{R}^2$  via the formulas

$$f(s, t) := \int_{\Omega_0} \int_{p_0(x, y)}^{p_{s, t}(x, y)} g(x, y, z) dz dx dy \quad \text{and} \quad h(s, t) := \int_{\Omega_0} \int_{p_0(x, y)}^{p_{s, t}(x, y)} e(x, y, z) dz dx dy.$$

By the definition of  $\Omega_0$ , the continuity of  $p \mapsto \kappa(x, y, p)$  and (3.4), it follows then that there is  $\delta > 0$  such that for any  $(x, y) \in \Omega_0$  and any  $s, t \in (-\delta, \delta)$  we have

$$p_{s, t}(x, y) > p_0(x, y) \quad \text{and} \quad L_{p_{s, t}}(x, y) < \kappa(x, y, p_{s, t}(x, y)).$$

Since  $\bar{p}$  is an optimal profile for the final open pit problem  $(P_{3D})$ , it follows that  $(0, 0)$  is a local maximum

of the problem

$$\text{Maximize } f(s, t) \text{ over all } s, t \in \mathbb{R} \text{ subject to } h(s, t) \leq c_{\max}.$$

This nonlinear optimization problem satisfies the so-called Mangasarian-Fromovitz condition because

$$\partial_t h(0, 0) = \int_{\Omega_0} e(x, y, \bar{p}(x, y)) \psi(x, y) dx dy = 1.$$

Where the first equality is justified by the Dominated Convergence Theorem and the fact that  $e \in L^\infty(\Omega \times \mathbb{R})$ . Therefore, by the Karush-Kuhn-Tucker theorem, there is  $\bar{\lambda} \leq 0$ , which in principle depends on  $\psi$  and  $\varphi$ , such that

$$\nabla f(0, 0) + \bar{\lambda} \nabla h(0, 0) = 0 \quad \text{and} \quad \bar{\lambda} (h(0, 0) - c_{\max}) = 0.$$

Moreover, similarly as justified above, it is not difficult to see that

$$\partial_s h(0, 0) = \int_{\Omega_0} e(x, y, \bar{p}(x, y)) \varphi(x, y) dx dy,$$

$$\partial_s f(0, 0) = \int_{\Omega_0} g(x, y, \bar{p}(x, y)) \varphi(x, y) dx dy,$$

$$\partial_t f(0, 0) = \int_{\Omega_0} g(x, y, \bar{p}(x, y)) \psi(x, y) dx dy.$$

On the one hand, by the condition over the partial derivatives with respect to the  $t$  variable we get  $\bar{\lambda}$  does not depend on  $\varphi$  because

$$\int_{\Omega_0} g(x, y, \bar{p}(x, y)) \psi(x, y) dx dy = \partial_t f(0, 0) = -\bar{\lambda} \partial_t h(0, 0) = -\bar{\lambda}.$$

On the other hand, by the condition over the partial derivatives with respect to the  $s$  variable we get

$$\int_{\Omega_0} (g(x, y, \bar{p}(x, y)) + \bar{\lambda} e(x, y, \bar{p}(x, y))) \varphi(x, y) dx dy = 0.$$

But, since  $\varphi \in \mathcal{C}_0^\infty(\Omega)$  is arbitrary and  $(x, y) \mapsto g(x, y, \bar{p}(x, y)) + \bar{\lambda} e(x, y, \bar{p}(x, y))$  belongs in particular to  $L^2(\Omega_0)$ , by the fundamental lemma of the calculus of variations (cf. [66, Lemma 3.2.3]) the conclusion follows.

Finally, for the case that  $e(x, y, \bar{p}(x, y))$  is identically zero in  $\Omega_0$  it is enough to define  $s \mapsto f(s)$  on  $\mathbb{R}$  via the formula

$$f(s) := \int_{\Omega_0} \int_{p_0(x, y)}^{\bar{p}(x, y) + s\varphi(x, y)} g(x, y, z) dz dx dy,$$

with  $\varphi \in \mathcal{C}_0^\infty(\Omega_0)$  arbitrary and check that  $s = 0$  is a local minimum of  $f$ . Then the conclusion follows by using the Fermat rule ( $f'(0) = 0$ ) and fundamental lemma of the calculus of variations.  $\square$   $\square$

**Remark 3.3.3.** Note in particular that Theorem 3.3.1 says that if the marginal cost associated with extracting a block at any depth is zero (there is no capacity constraint), that is,  $e(x, y, z) = 0$  for any  $(x, y) \in \Omega$  and  $z \geq p_0(x, y)$ , then the marginal gain of extracting a block at any depth must be zero on the subsection  $\Omega_0$ , that is,

$$g(x, y, \bar{p}(x, y)) = 0, \quad \text{ct p. in } \Omega_0.$$

### 3.4 Future work and final remarks

In this chapter we have provided necessary optimality conditions for a profile of an open pit mine to be an optimum for the final open pit problem in the 2D as well as on the 3D. Both settings involve isoperimetric restriction, which are hard to handle in general. Nonholonomic restrictions, such as the slope condition has been treated only for the 2D case. The 3D case remains as an open question that deserves some attention, and which we plan to address in a future work.

Obtaining numerical solutions is still an open problem. For this purpose, a classical direct method or an indirect method using the results of this chapter can be implemented. Preliminary simulations have been done with the help of the INRIA solver for optimal control problems BOCOP [27]. This is an issue that need to be investigated in more details.

Finally, let us mention that, similarly as done for the 2D case, the maximal slope condition in the 3D can actually be subsumed by a control type condition

$$\nabla p(x,y) = \kappa(x,y,p(x,y)) (\cos(\theta), \sin(\theta)), \quad \text{for a.e. } (x,y) \in \Omega, \theta \in [0, 2\pi).$$

Thus, a possible way to address the final open pit problem is to study the optimal control problem associated with this constraints. This issue needs to be investigated in details.





## Chapter 4

# Numerical study of 2D and 3D Final and Sequential Open Pit

This chapter corresponds to the submitted pre-print [80] titled "Optimal control approaches for Open Pit planning"

### 4.1 Introduction

In long-term planning of mine operation, a common task consists in determining the profile of the total mass of material to be extracted from the site to optimally design an opencast mine. This so-called Final Open Pit problem was introduced in the early works Ref. [74, 61], with a more recent overview in Ref. [84].

The typical approach used to solve this problem is based on a discrete block model of the site, each block having an associated extraction cost and profit value, based on topographical and geological data. Using a graph of block precedence (i.e. order of extraction) allows to take into account slope constraints for the mine stability, and gives rise to large Integer Programming problems, see for instance Ref. [29]. The dynamic programming approach was also investigated in this framework, see e.g. Ref. [101]. Another approach presented in Ref. [93] uses a PDE formulation for time labeling functions.

The present chapter follows the continuous approach introduced in Ref. [3] with the reformulation of the Open Pit using a calculus of variation framework, and then in Chapter 3 as an optimal control problem. The main contributions of the present work include the analysis of the Final Open Pit (FOP from now) with capacity, slope and initial profile constraints, using Pontryagin's Maximum Principle to extend the results previously obtained in chapter 3. Then we introduce a new semi-continuous formulation that can handle the Sequential Open Pit problem (SOP from now), i.e. optimization of the mine profile over a sequence of several time-frames, for a 2D space domain (3D mine profile). Finally, numerical simulations are provided for both the continuous and semi-continuous approaches, including global optimization for the 2D FOP case, and to our knowledge the first results for the 3D profile optimization as an optimal control problem. The outline of the chapter is as follows. After the introduction presenting context, Section II covers the SOP problem statement with the continuous approach, and introduces the semi-continuous formulation. Section III presents the FOP analysis using Pontryagin's Maximum Principle and in particular discusses the control structure in terms of bang, constrained and singular arcs. Section IV presents the numerical simulations for three test cases: 2D FOP, 2D SOP and 3D SOP, and is followed by the conclusions.

## 4.2 Problem statement

For a given spatial domain  $\Omega$ , we consider a continuous function  $p : \Omega \rightarrow \mathbf{R}$  called profile that delimits the shape of the mine pit. The aim is to determine the profile that maximizes the gain from the excavated soil, while respecting some limits for the excavated capacity and maximal slope of the mine. We recall now the continuous approach for open pit planning, and introduce a new semi-continuous approach that can handle the 3D profile case. Both of these approaches lead to optimal control formulations of the problem.

### 4.2.1 Continuous formulation

The key idea in the so-called continuous formulation, originally introduced in [3], is to use the distance (position along the x-axis) as independent variable, which allows to define the mine profile as a function of this new 'time'. Introducing a suitable dynamics for this function, with the associated control function, allows to formulate the open pit planning as an optimal control problem (OCP).

#### Final open pit planning problems (FOP)

For the 1-dimensional case, the domain  $\Omega = [a, b]$  will correspond to the independent variable or 'time' of the optimal control problem. Consider the state variables  $P, c : [a, b] \rightarrow \mathbb{R}^+$  for the depth profile of the pit and the excavated capacity of the mine. Let us also denote  $P_0 \in \mathcal{C}^1 \geq 0$  the initial profile corresponding to the natural shape of the ground. We set the state constraint  $P(t) \geq P_0(t), \forall t \in [a, b]$ , and the boundary conditions  $P(a) = P_0(a), P(b) = P_0(b)$ . An additional final condition is that the total excavated capacity is limited, i.e.  $c(b) \leq c_{max}$ .

We also introduce  $\kappa : [a, b] \times \mathbb{R} \rightarrow \mathbb{R}^*$  such that  $\kappa(t, z)$  is the maximal pit slope at position  $t$  and depth  $z$ . Instead of the original dynamics  $\dot{P} = u$ , we choose to use a normalized control  $u : [a, b] \rightarrow [-1, 1]$  which is a bit simpler than having the mixed state-control constraint  $|u(t)| \leq \kappa(t, P(t))$  for the maximal slope. As part of the soil characteristics, we also note  $g, e : [a, b] \times \mathbb{R} \rightarrow \mathbb{R}$  the densities of gain and effort for excavating at a given position and depth. The optimal control formulation of (FOP) is then as follows:

$$(FOP) \left\{ \begin{array}{l} \max \int_a^b \int_{P_0(t)}^{P(t)} g(t, z) dz dt \\ \dot{P}(t) = u(t) \kappa(t, P(t)) \quad \forall t \in [a, b] \\ \dot{c}(t) = \int_{P_0(t)}^{P(t)} e(t, z) dz \quad \forall t \in [a, b] \\ u(t) \in [-1, 1] \quad \forall t \in [a, b] \\ P_0(t) - P(t) \leq 0 \quad \forall t \in [a, b] \\ P(a) = P_0(a), P(b) = P_0(b) \\ c(a) = 0, c(b) \leq c_{max} \end{array} \right.$$

**Remark 4.2.1.** *In the following we take the basic effort function  $e \equiv \mathbf{1}$ . The gain function  $g$  is typically defined by interpolation over tabular data, and has to be integrated numerically along the depth  $z$ .*

#### Sequential open pit planning (SOP)

We introduce now an extended version of (FOP), in which we want to schedule an extraction program over  $N$  consecutive time-frames. This case is quite relevant in mine planning since mining companies

divide the digging process into periods for operational purposes. We extend the notations of (*FOP*) to the multi-frame framework, and note  $P_i$  the mine profile at time-frame  $i$ , with the associated control  $u_i$ , while  $c_i$  is the excavated capacity during time-frame  $i$ . Each mine profile has to be deeper than the previous one, i.e. the constraint  $P \geq P_0$  from (*FOP*) is generalized as  $P_i \geq P_{i-1}, i = 1 \dots N$ . The capacity limit  $c_{max}^i$  is now enforced at each individual time-frame. Finally, the objective function now takes into account a depreciation rate  $\alpha > 0$  over time, with the gains for the more distant time-frames being valued less than for the more immediate time-frames. This new optimal control problem reads as follows

$$(SOP) \left\{ \begin{array}{l} \max \sum_{i=1}^N \int_a^b \int_{P_{i-1}(t)}^{P_i(t)} \frac{g(t,z)}{(1+\alpha)^{t-1}} dz dt \\ \dot{P}_i(t) = u_i \kappa(t, P_i(t)) \quad \forall t \in [a, b], \quad i = 1, \dots, N \\ \dot{c}_i = \int_{P_{i-1}(t)}^{P_i(t)} e(t, z) dz \quad \forall t \in [a, b], \quad i = 1, \dots, N \\ u_i(t) \in [-1, 1], \quad \forall t \in [a, b], \quad i = 1, \dots, N \\ P_{i-1}(t) - P_i(t) \leq 0 \quad \forall t \in [a, b], \quad i = 1, \dots, N \\ P_i(a) = P_0(a), P_i(b) = P_0(b) \quad i = 1, \dots, N \\ c_i(a) = 0, c_i(b) \leq c_{max}^i \quad i = 1, \dots, N \end{array} \right.$$

**Remark 4.2.2.** Note that (*SOP*) with  $N = 1$  corresponds to (*FOP*). Numerically, the multi-process (*SOP*) can be reformulated by duplicating the state and control variables (as well as the constraints) for each time-frame. Adding the proper linking constraints between the final and initial conditions of the successive time-frames, we obtain a single process version of (*SOP*) that can be solved by standard methods. The overall problem dimension, however, is higher, therefore computationally expensive methods such as global optimization may be able to handle (*FOP*) but not (*SOP*), see section 4.4.

**Remark 4.2.3.** For the discrete (block) formulation, it is known (see for instance Ref. [30] and [79]) that each profile (or 'pit') which is solution of (*SOP*) is not deeper than the optimal pit of (*FOP*) with the same parameters and infinite capacity. A similar result has been obtained for the continuous framework in Ref. [3].

### 4.2.2 Semi continuous formulation for SOP

The main limitation of the continuous approach is that using the independent variable to represent the position in space makes it difficult to handle the 3D profile case, both in terms of dynamics / controls and profile slopes. This is why we introduce a new approach called the semi-continuous formulation, based on an explicit discretization of the space domain  $\Omega$ . The mine profile is therefore represented by a finite set of variables at the discretization nodes, as illustrated in Figure 4.1. Control variables are defined as the excavation effort at each discretization node. Slope constraints are modeled as state constraint linking each node with their neighbors. The independent variable is here standard time, expressed in time-frames such that the final time  $T$  is the total number of time-frames. Since *SOP* is a multi-phase problem, one standard way to formulate it is to normalize the time interval to  $[0, 1]$  and duplicate the variables for each time-frame. This approach yields another optimal control formulation of the Sequential Open Pit problem, for which extension from 1D to 2D space domain is rather straightforward, at the cost of an increase in overall problem dimension.

**Notations.** In the context of the semi-continuous approach, for functions of both space and time such as profile, controls and slopes, we will typically use subscripts for the space discretization node in  $\Omega$ , and

exponents for the time-frame of the multi-phase Sequential Open Pit. For instance,  $P_i^k$  will represent the profile depth at node  $i$  and time-frame  $k$ , and  $P^k := (P_i^k), i = 0, \dots, N$  refers to the mine profile at time-frame  $k$ . Similarly,  $U_i^k$  will denote the digging at node  $i$  and time-frame  $k$ , with  $U^k := (U_i^k), i = 0, \dots, N$  corresponding to the overall excavation effort over the domain  $\Omega$  at time-frame  $k$ . We will also denote by  $\int_{P^k}^{P^{k+1}}$  the integral of a function between the two mine profiles at time-frames  $k$  and  $k+1$ ; for 2D profiles this is a 2D integral along  $x$  and the depth  $z$ , and a 3D integral along  $x, y, z$  for 3D profiles.

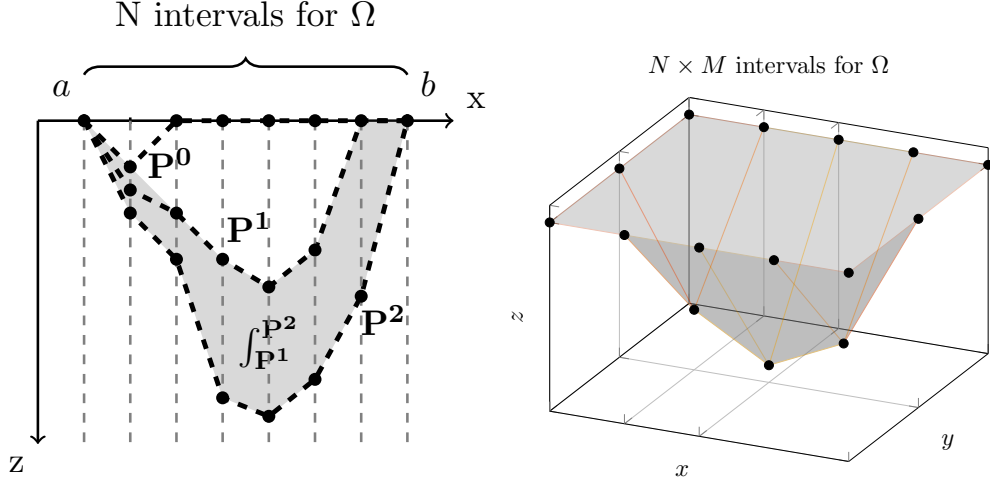


Figure 4.1: 1D / 2D Space discretization of the mine profile

### One dimensional profile space domain

**Discrete profile.** We discretize the space domain  $\Omega = [a, b]$  into  $N$  equal intervals of length  $\Delta x = \frac{b-a}{N}$ , with  $N+1$  discretization nodes  $(x_i)$ , and note  $I = \{0, \dots, N\}$  the set of indices for the nodes. We define the state variables for the profile nodes  $(P_i)_{i \in I}$  as functions of time. We also introduce the control variables at each node  $(U_i)_{i \in I} \geq 0$ , corresponding to the excavation effort, so that the profile variables follow the simple dynamics

$$\dot{P}_i(t) = U_i(t), \quad \forall i \in I, \quad \forall t \in [0, T]. \quad (4.1)$$

**Gain.** The achieved gain during time-frame  $k$  is the integral of  $g$  between the current profile  $P^k$  and the previous  $P^{k-1}$ . Taking into account the depreciation rate  $\alpha$  introduced in 4.2.1, the overall gain to be maximized is

$$\sum_{k=1}^T \int_{P^{k-1}}^{P^k} \frac{g(x, z)}{(1 + \alpha)^{k-1}} dx dz. \quad (4.2)$$

The computation of this objective is detailed in 4.6.

**Slope.** We denote  $S_i^k$  the slope at node  $i$  and time-frame  $k$ , which is a function of time. The maximal slope condition writes as

$$-1 \leq \frac{S_i^k(t)}{\kappa(x_i, P_i^k(t))} \leq 1, \quad \forall i \in I, \quad \forall k = 0 \dots T, \quad \forall t \in [0, 1] \quad (4.3)$$

In the 2D case we will use the simple slope formula

$$S_i^k = (P_i^k - P_{i-1}^k)/\Delta x \quad (4.4)$$

and the slope limits are state constraints.

**Capacity.** The excavation effort at each time-frame  $k$  corresponds to the integral of the effort  $E$  between the two consecutive profiles  $P^{k-1}$  and  $P^k$ , and the capacity limit writes as

$$\int_{P^{k-1}}^{P^k} e(x, z) dx dz \leq C_k, \quad \forall k = 1, \dots, T. \quad (4.5)$$

The computation of this integral is detailed in 4.6.

**Initial profile.** This is now a standard initial condition of the form

$$P_i^0(0) = p_0(x_i), \quad \forall i \in I. \quad (4.6)$$

We obtain the following multi-phase problem  $(SOP)_{SC}^{2D}$  with Fig. 4.2 illustrating the profile discretization in the 2D case, with  $N = 7$  and  $T = 2$ . Implementation details regarding the approximation of the various integrals are presented in 4.6

$$(SOP)_{SC}^{2D} \left\{ \begin{array}{l} \max \sum_{k=1}^T \int_{P^{k-1}}^{P^k} \frac{g(x, z)}{(1 + \alpha)^{k-1}} dx dz \\ P_i^k(t) = U_i^k(t), \quad i \in I, \quad k = 1, \dots, T, \quad t \in [0, 1] \\ -1 \leq \frac{S_i^k(t)}{\kappa(x_i, P_i^k(t))} \leq 1, \quad i \in I, \quad k = 1, \dots, T, \quad t \in [0, 1] \\ \int_{P^{k-1}}^{P^k} e(x, z) dx dz \leq C_k, \quad k = 1, \dots, T \\ P_i^0(0) = p_0(x_i), \quad i \in I \\ U_i^k(t) \geq 0, \quad i \in I, \quad k = 1, \dots, T, \quad t \in [0, 1] \end{array} \right.$$

**Remark 4.2.4.** Setting  $T = 1$  corresponds to the Final Open Pit problem with a single time-frame.

**Remark 4.2.5.** The boundary condition  $P|_{\partial\Omega} = 0$  is in practice built in directly in the problem formulation by eliminating the profile and control variables at the nodes corresponding to the boundary of the space domain.

**Remark 4.2.6.** Moreover, the constraint that each profile must be deeper than the previous one, which was a state constraint in the continuous formulation, is now simply enforced by the conditions  $U_i \geq 0$ .

**Remark 4.2.7.** The step size  $\Delta x$  for the discretization of  $\Omega$  in the semi-continuous approach can be seen as the analogue of the time step  $\Delta t$  for the continuous approach, which uses distance as independent variable.

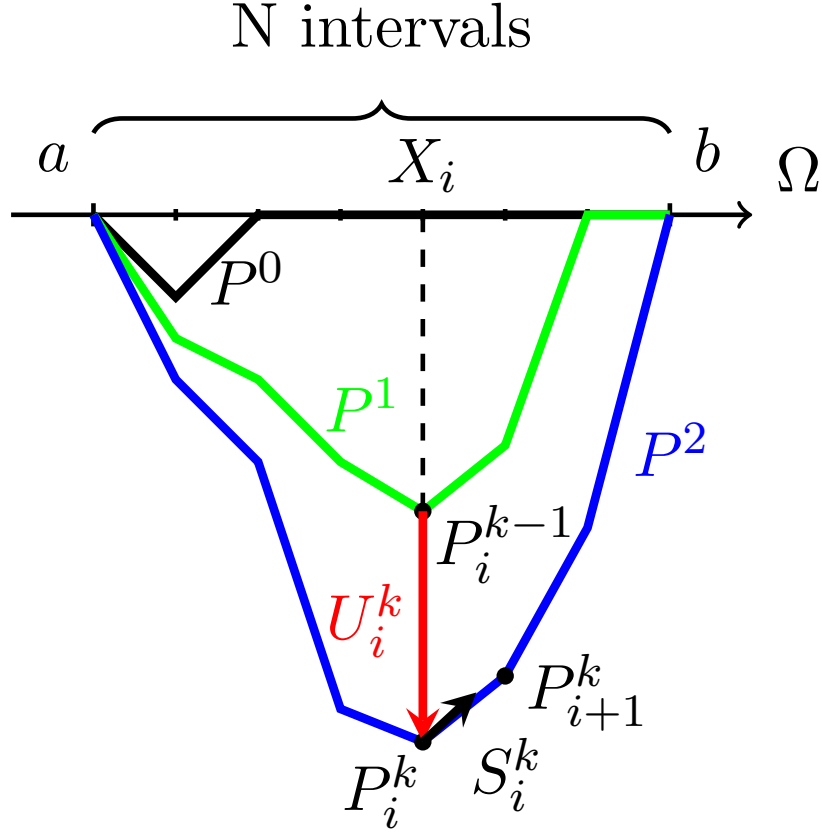


Figure 4.2: Illustration of the 2D profile model discretized w.r.t. space and time as a set of  $P_i^k$  state variables with  $i = 0, \dots, N$  profile nodes and  $k = 0, \dots, T$  time-frames. Controls  $U_i^k$  are the depths excavated from the previous time-frame at each node. Slopes  $S_i^k$  between neighbor nodes must be smaller than the local maximal slopes i.e.  $\kappa(X_i, P_i^k)$ .

### Two dimensional profile space domain

For the two dimensional case, the extraction domain considered is  $\Omega = [a, b] \times [c, d]$ . Generalizing the 2D case, we discretize  $[a, b]$  and  $[c, d]$  into  $N$  and  $M$  intervals of length  $\Delta x = \frac{b-a}{N}$  and  $\Delta y = \frac{d-c}{M}$  respectively, and obtain a grid with  $(N+1) \times (M+1)$  nodes. Noting  $J = \{0, \dots, M\}$ , we introduce the state variables (functions of time)  $(P_{i,j})_{i,j \in I \times J}$  representing the mine depth at each node  $(x_i, y_j) := (a + i\Delta x, c + j\Delta y)$ . The mine profile at time-frame  $k$  is now a surface represented by the set of points  $P^k := (P_{i,j}^k(0))$ . We introduce the  $(N+1) \times (M+1)$  non-negative controls  $U_{i,j}^k \geq 0$ ,  $i, j \in I \times J$ , with the same dynamics  $\dot{P}_{i,j}^k = U_{i,j}^k$ .

Initial profile conditions are written as:

$$P_{i,j}^0(0) = p_0(x_i, y_j), \quad i, j \in I \times J. \quad (4.7)$$

The objective and capacity limit are similar to the 2D case, except that the integrals of  $g$  and  $e$  between two consecutive profiles are now in 3D instead of 2D. The relevant implementation details are provided

in 4.6.

The main adjustment concerns the slope constraint: for each point  $P_{i,j}$  of the profile we now choose to consider two slopes  $S_{i,j}$  and  $T_{i,j}$ , in the x-axis and y-axis directions respectively. Using the same basic forward finite differences as in 2D, we obtain the two sets of slope constraints at time-frame  $k$ :

$$-1 \leq \frac{P_{i+1,j}^k(t) - P_{i,j}^k(t)}{\kappa(x_i, y_j, P_{i,j}^k(t)) \Delta x} \leq 1, \quad \forall i = 0, \dots, N-1, \quad j = 0, \dots, M-1, \quad \forall t \in [0, 1]. \quad (4.8)$$

$$-1 \leq \frac{P_{i,j+1}^k(t) - P_{i,j}^k(t)}{\kappa(x_i, y_j, P_{i,j}^k(t)) \Delta y} \leq 1, \quad \forall i = 0, \dots, N-1, \quad j = 0, \dots, M-1, \quad \forall t \in [0, 1]. \quad (4.9)$$

Note that more sophisticated choices could be used for the slopes, such as centered differences formulas or increasing the number of slopes considered at each point. The formulation of  $(SOP)_{SC}^{3D}$  is summarized below, with Fig. 4.3 illustrating the profile discretization in the 3D case with  $N = 5$  and  $M = 3$ .

$$(SOP)_{SC}^{3D} \left\{ \begin{array}{l} \max \sum_{k=1}^T \int_{p^{k-1}}^{p^k} \frac{g(x, y, z)}{(1 + \alpha)^{k-1}} dx dy dz \\ \dot{P}_{i,j}^k(t) = U_{i,j}^k(t), \quad \forall (i, j) \in I \times J, \quad k = 1, \dots, T, \quad t \in [0, 1] \\ -1 \leq \frac{S_{i,j}^k(t)}{\kappa(x_i, y_j, P_{i,j}^k(t))} \leq 1, \quad \forall (i, j) \in I \times J, \quad k = 1, \dots, T, \quad t \in [0, 1] \\ -1 \leq \frac{T_{i,j}^k(t)}{\kappa(x_i, y_j, P_{i,j}^k(t))} \leq 1, \quad \forall (i, j) \in I \times J, \quad k = 1, \dots, T, \quad t \in [0, 1] \\ \int_{p^{k-1}}^{p^k} e(x, y, z) dx dy dz \leq C_k, \quad k = 1, \dots, T \\ P_{i,j}^0(0) = p_0(x_i, y_j), \quad \forall (i, j) \in I \times J \\ \dot{U}_{i,j}^k(t) \geq 0, \quad \forall (i, j) \in I \times J, \quad k = 1, \dots, T, \quad t \in [0, 1] \end{array} \right.$$

### 4.3 Analysis and optimality conditions for FOP

We study the final open pit problem in continuous formulation ( $FOP$ ) by applying Pontryagin's Maximum Principle (Ref. [87]), and look at the possible control structure of optimal profiles.

Optimality conditions for  $(SOP)$  are not detailed here, and are more involved in particular due to the state constraint  $P \leq P_0$  being generalized over the sequence of time-frames, i.e.  $P_i \leq P_{i-1}, i = 1, \dots, N$ .

#### 4.3.1 Applying Pontryagin's Maximum Principle

Following the formulation in Ref. [25], we now state the PMP for ( $FOP$ ). We denote  $y$  the state variables,  $p$  the associated costate variables,  $l$  the running cost,  $f$  the dynamics and  $h$  the state constraint. In all the following we assume the so-called **normal case**, i.e. the multiplier associated to the cost is nonzero and



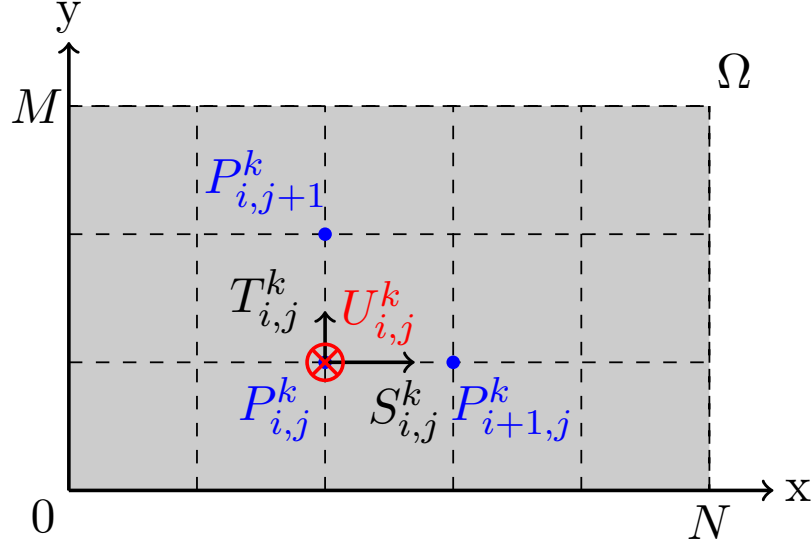


Figure 4.3: Illustration of the 3D profile model with  $N = 5$  and  $M = 3$ . View is from 'above', with the state variable  $P_{i,j}^k$  giving the profile depth at node  $(x_i, y_j)$  at time-frame  $k$ . Slopes  $S_{i,j}^k, T_{i,j}^k$  with neighbors along the x-axis and y-axis must be smaller than the maximal allowed slopes given by the function  $\kappa$ . The control  $U_{i,j}^k$  (along the z-axis) corresponds to the excavated depth from the same profile node at the previous time-frame  $P_{i,j}^{k-1}$ .

can be normalized to 1. Let us define the pre-Hamiltonian, omitting the argument  $t$  of functions  $u, y, p$  for clarity:

$$H(t, u, y, p) = l(t, u, y) + p \cdot f(t, y, u) \quad (4.10)$$

$$= - \int_{P_0(t)}^{P(t)} g(t, z) dz + p_P \kappa(t, P) + p_c (P(t) - P_0(t)) \quad (4.11)$$

Noting the function of bounded variation  $\mu \in BV(0, T)$  the multiplier associated with the state constraint, the adjoint equation writes as

$$-dP(t) = \nabla_y H(t, u, y, p) dt + \nabla_y h(t, y) d\mu(t) \quad (4.12)$$

Then for any local optimum  $(\bar{y}, \bar{u})$ , there exists a non-trivial set of multipliers  $(\bar{p}, \bar{\mu})$  such that the following relations are satisfied:

i) Adjoint equation

$$d\bar{p}_P(t) = (g(t, \bar{P}) - \bar{p}_P(t) \bar{u}(t) \kappa_P(t, \bar{P}) - \bar{p}_c(t)) dt + d\bar{\mu}(t) \quad (4.13)$$

$$d\bar{p}_c(t) = 0 \quad (4.14)$$

ii) Transversality conditions

$$\bar{p}_P(a), \bar{p}_P(b), \bar{p}_c(a) \text{ are free; } \bar{p}_c(b) \geq 0 \text{ with } \bar{p}_c(b) = 0 \text{ if } \bar{c}(b) < c_{max} \quad (4.15)$$

iii) Hamiltonian minimization

$$\bar{u}(t) \in \underset{w}{\operatorname{argmin}} H(t, w, \bar{y}(t), \bar{p}(t)) \text{ a.e. on } (a, b) \quad (4.16)$$

iv) State constraint complementary relations

$$d\bar{\mu}(t) \geq 0, \int_a^b (P_0(t) - \bar{P}(x)) d\bar{\mu}(t) = 0 \text{ and } \bar{\mu}(b) = 0 \quad (4.17)$$

i.e.  $\bar{\mu}$  is a non-decreasing function and is constant when the state constraint is not active.

**Remark 4.3.1.** *The state constraint is of order 1 since the control appears in its first time derivative  $\dot{h} = -u\kappa$ . We refer the reader to, for instance, Ref. [25] for a more in-depth analysis of state constraints in the PMP framework, and especially the so-called "alternate adjoint" formulation.*

### 4.3.2 Inactive case: bang/singular control

We start by studying the case when the state constraint is not active. Since per (4.16) the optimal control minimizes the pre-Hamiltonian which is linear in the control, solutions typically consist in a sequence of **bang** (saturated control) and/or **singular** control arcs. We introduce the **switching function** whose sign will determine the optimal control

$$\psi(t) := H_u(t) = p_P(t)\kappa(t, P(t)) \quad (4.18)$$

As  $\kappa$  has strictly positive values we obtain the control law:

$$\bar{u}(t) = \begin{cases} 1 & \text{if } p_P(t) < 0 \\ -1 & \text{if } p_P(t) > 0 \\ u_s(t) & \text{if } p_P(t) = 0 \text{ over an interval} \end{cases} \quad (4.19)$$

The value of the singular control  $u_s$  is traditionally determined from the fact that  $\psi$  and all its time derivatives vanish over a singular arc.

Over a singular arc,  $\psi$  vanishes and the first time derivative of the switching function can be reduced to

$$\dot{\psi}(t) = (g(t, P(t)) - p_c)\kappa(t, P(t)) \quad (4.20)$$

and similarly, by plugging  $\dot{\psi}(t) = 0$  in the second derivative and recalling  $\dot{p}_c = 0$ , we obtain

$$\ddot{\psi}(t) = (g_t(t, P(t)) + u\kappa g_P(t, P(t)))\kappa \quad (4.21)$$

Solving  $\ddot{\psi}(t) = 0$  for the singular control leads to

$$u_s(t) = -\frac{g_t(t, P(t))}{\kappa g_P(t, P(t))} \quad (4.22)$$

We can now derive the two following lemmas concerning singular arcs.

**Lemma 4.3.1.** *A singular arc is not admissible when  $|\frac{g_t(t, P)}{\kappa g_P(t, P)}| > 1$ , and in particular when  $g_P(t, P) = 0$ .*

*Proof.* Immediate consequence of (4.22) and the control constraint  $u \in [-1, 1]$   $\square$

**Lemma 4.3.2.** *Let  $\bar{P}$  be an optimal profile solution of (FOP), then, over a singular arc the curve  $(t, \bar{P}(t))$  follows the geodesic of  $g$ . Moreover, when maximal capacity is not reached, singular arcs follow more specifically the geodesics of null gain  $g = 0$ .*

*Proof.* From (4.20), over a singular arc the equation  $\dot{\psi} = 0$  indicates that the derivative  $\dot{g} := g_t(t, P(t)) + \dot{P}g_P(t, P(t))$  vanishes, therefore the mine profile will follow the geodesics of  $g$ . If the maximal capacity constraint is not active, then the associated costate  $p_c$  is zero (see (4.15)), and  $\dot{\psi} = 0$  then gives  $g = 0$ .  $\square$

These lemmas expand the analysis of singular arcs obtained in Chapter 3 with calculus of variations techniques.

### 4.3.3 Active state constraint case

Over a constrained arc, the control  $u_c$  is such that the constraint remains active, i.e.  $h = P_0 - P = 0$ , leading to the expression

$$u_c(t) = \frac{\dot{P}_0(t)}{\kappa(t, P(t))} \quad (4.23)$$

Note that a constrained arc can only occur if the  $u_c$  is admissible, i.e.  $|\dot{P}_0(t)| \leq \kappa(t, P(t))$ . This simply means that the initial profile must satisfy the maximal slope constraint.

### 4.3.4 Control structure summary

To summarize, the optimal profile, in terms of control structure, is a sequence of bang, singular and/or constrained arcs. **Constrained** arcs are where the profile is the same as the initial one, meaning there was no further excavation on these parts of the domain. **Bang** arcs correspond to the parts of the profile where the slope reaches its maximal allowed value, i.e. the digging is as steep as possible. **Singular** arcs, on the other hand, follow the geodesics of the gain function, meaning the gain is constant along these parts of the profile. Moreover, if the capacity limit is not reached, then this geodesic is more specifically the one of null gain, i.e. the digging stops where excavation is not profitable anymore.

Optimality conditions for **the semi-continuous formulation** are more involved and remain to be investigated thoroughly. The main complications arise from the spatial discretization of the profile, leading to maximal slope limits now being state constraints that involve adjacent nodes variables (including controls).

## 4.4 Numerical simulations

We present now the numerical simulations that illustrate the continuous and semi-continuous formulations of the Open Pit problem. After a brief description of the algorithms used for the global and local optimizations, we detail three test cases. First is the 2D FOP with limited capacity, that we solve with the continuous (both global and local optimization) and semi-continuous formulation (local optimization). The second example is the 2D SOP with limited capacity, for which we compare the results of both continuous and semi-continuous formulations (both with local optimization). Finally, we present a test case for the 3D SOP problem with the semi-continuous formulation. All simulations were carried out on a standard laptop, with numerical settings for all methods recalled in Table 4.1 p.39.

### 4.4.1 Numerical methods

#### Global optimization: FOP with continuous formulation

The Final Open Pit problem is low-dimensional, with only two state variables (not counting the running cost) and one control variable. Therefore it makes sense to try a global optimization method such as dynamic programming, or the so-called Hamilton-Jacobi-Bellman (HJB) approach. We use here the software BOCOPHJB [27], and refer to for instance Ref. [46] for a detailed description of the HJB method. In this approach the value function of a fully discretized (time, state and control variables) version of the problem is computed, with the global optimum then being reconstructed from this information.

#### Local optimization: FOP and SOP with continuous and semi-continuous formulations

Since the numerical cost of the global method is too high for the Sequential Open Pit problem, we also use a local optimization method, namely the direct transcription approach. This method approximates the original (OCP) problem by a discretized reformulation as a nonlinear optimization problem (NLP), using a discretization of the time interval. We refer interested readers to for instance Ref. [19] for a review of direct methods. We use here the software BOCOP [26], based on the solver IPOPT [98] with sparse derivative computed by the automatic differentiation tool CPPAD [18]. This local optimization method is also used for the semi-continuous formulation with explicit discretization of the space domain.

#### Numerical settings

In Table 4.1 are the settings for the different numerical methods used in the simulations.

2D FOP (global)	$t : 123$ steps; $P : 50$ steps, $e : 210$ steps; $u : 100$ steps
2D FOP/SOP (local)	$t : 123$ steps; $tol = 10^{-10}$ ; $maxiter = 10000$
2D SOP SC (local)	$t : T$ steps; $N = 123$ nodes; $tol = 10^{-10}$ ; $maxiter = 10000$
3D SOP SC (local)	$t : T$ steps; $30 \times 10$ nodes; $tol = 10^{-6}$ ; $maxiter = 10000$

Table 4.1: Numerical settings for the continuous and semi continuous formulations

### 4.4.2 Final open pit (2D): global and local optimization

We start with the 2D FOP as first example, since it is the only one for which all formulations, including global optimization, are available. We set a maximal capacity  $c_{max} = 20.000$ . The gain function  $g$  is interpolated from values found in the Marvin block model of MINELIB, a publicly available library of test problem instances for open pit mining problems (see [45]).

**Remark 4.4.1.** *Solutions for the unlimited capacity case, with a different control structure (i.e. singular arcs), are shown in 4.7, with both constant and variable maximal slope.*

#### 2D FOP with global optimization for continuous approach

The solution obtained by the global optimization is displayed in Figure 4.4. At first glance, the control structure seems to be of the form **Constrained-Bang-Bang-Constrained**. On both sides the constraint  $P = P_0$  is active, meaning there is no additional digging from the initial profile. In the middle, digging occurs with maximal slope, leading to the two bang arcs.

**Remark 4.4.2.** *The non zero control around  $x = 200$  simply follows the existing initial profile  $P_0$ , and is part of the first constrained arc. See 4.4.2 for more details.*

**Remark 4.4.3.** *It is worth noting that an estimate of the PMP costate can be derived from the gradient of the value function computed by the global method, see for instance Refs. [37, 38]. In the present case however, the gradient turns out to be quite noisy and of little practical use. This could be improved by increasing the discretizations, although the increase in computational times would not be competitive with respect to using a direct method.*

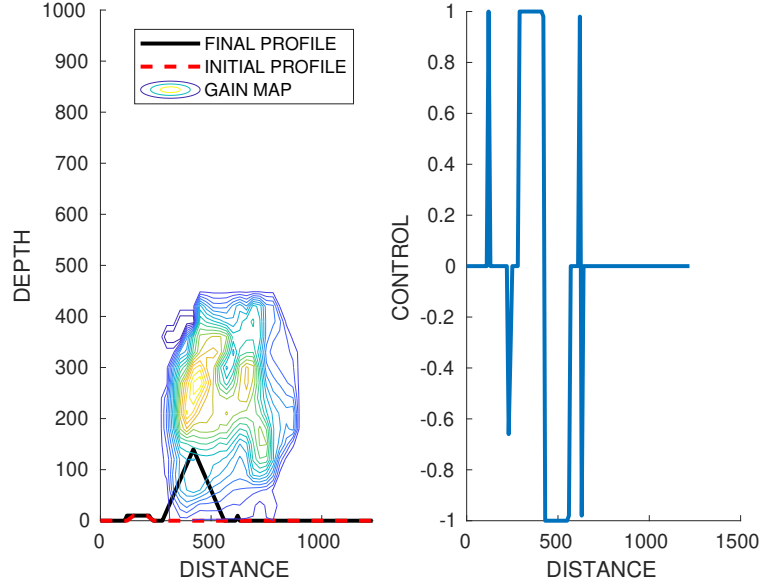


Figure 4.4: 2D profile with limited capacity - global optimization (HJB method)

## 2D FOP with local optimization for continuous approach

The solution from the local optimization is displayed in Figure 4.5 with the optimal profile and control as well as the PMP costate check. This solution is actually extremely close to the one in section 4.4.2, which indicates that the direct method actually found the global optimum as well, with the benefit of a more accurate solution. In particular, we can here clearly see that the first two arcs with nonzero control around  $x = 200$  are not bang arcs since  $|u| < 1$ : they are actually part of the first constrained arc and correspond to the region where  $P_0$  varies, thus the control  $u_c(t) = \frac{\dot{P}_0(t)}{\kappa(t, P(t))}$  from (4.23) is not just zero.

Moreover, we can now check that the **Constrained-Bang-Bang-Constrained** control structure is consistent with the switching function and the path constraint. We observe a perfect match between the adjoint estimate from the discretized problem and the recomputed PMP costate. Figure 4.5 shows the value of the state constraint  $h = P_0 - P$  and its associated multiplier  $d\mu$ . We retrieve  $d\mu$  from the multiplier of the state constraint in the discretized problem (the correspondence can be inferred from comparing the expression of the PMP Hamiltonian and the Lagrangian of the NLP problem). In accordance with (4.17), the multiplier  $d\mu$  is positive, and null when the constraint is not active. We also observe that the costate  $p_P$  is continuous at the junctions between bang and singular arcs, while the control is discontinuous.

**Remark 4.4.4.** *In this particular case, the solution has no singular arcs, which is due to the capacity limit that prevents reaching the null gain region. The examples with unlimited capacity in 4.7 and 4.7.2*

illustrate solutions with singular arcs where the optimal profile follows the geodesic  $g = 0$ .

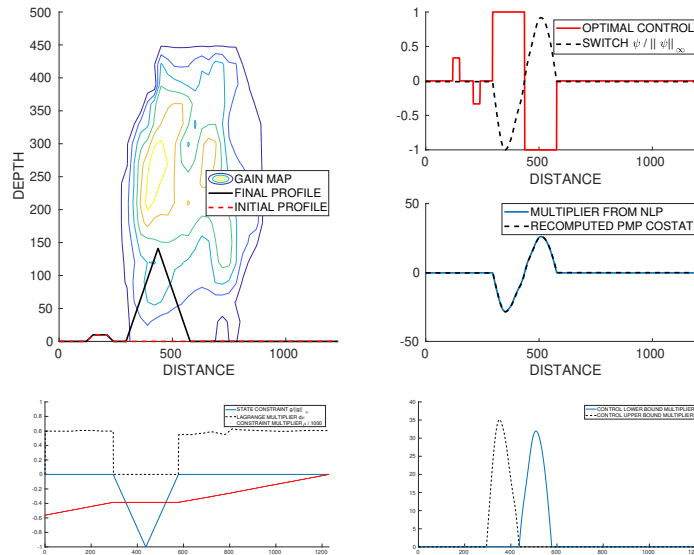


Figure 4.5: 2D profile with limited capacity - local optimization (direct method) and optimality conditions

### 2D FOP with semi-continuous approach

We finally present the solution obtained for the same problem using the semi-continuous formulation with a single phase (i.e  $T = 1$ ). As can be seen in Figure 4.6 and Table 4.2, the solution is similar to the global and local optimizations using the continuous approach, with close values for the objective. CPU times are of the same order of magnitude for the two local optimizations with continuous and semi-continuous formulation, while global optimization is significantly slower (two orders).

Method	Objective	CPU
2D FOP Global optim.	10846	369s
2D FOP Local optim.	11093	3s
2D FOP SC Local optim.	11100	2s

Table 4.2: Solutions for the 2D FOP with limited capacity.

### 4.4.3 Sequential Open Pit (2D and 3D): local optimization

In this section we present solutions for the Sequential Open Pit. First we solve a 2D example using both the continuous and semi-continuous formulations. Then we show a solution for a more realistic 3D problem using the semi-continuous formulation. To our knowledge, this is the first attempt to tackle the 3D case in an optimal control framework.

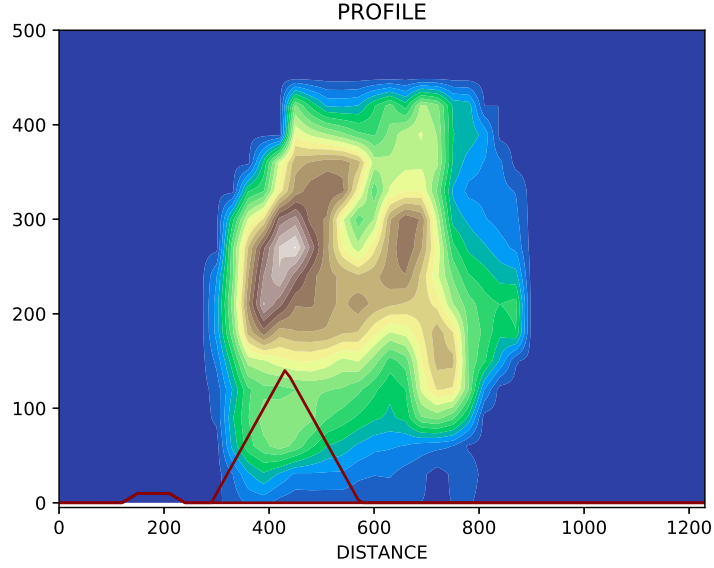


Figure 4.6: 2D profile, limited capacity - local optimization using semi-continuous formulation

### 2D SOP with continuous and semi-continuous approach

**Continuous approach.** We solve the 2D SOP problem for 12 time-frames, with a constant function  $\kappa = 1$ , a rate  $\alpha = 0.1$  and a maximal capacity  $c_{max} = 1e4$  for each time-frame. The solution indicates that most of the excavation effort is concentrated in the high gain regions of the domain, which is not surprising.

**Semi continuous approach.** We now solve the same SOP problem with the semi-continuous formulation. Both approaches give similar solutions, as can be seen in Figure 4.7. The objective values showed in Table 4.3 are quite close with a difference of 1.3%, while CPU times are in the same order of magnitude.

Method	Objective	CPU
Continuous	89939	31s
Semi-continuous	91153	43s

Table 4.3: Solutions for the 2D SOP problem.

### 3D SOP with semi continuous approach

For the 3D case we consider a domain  $\Omega = [0, 1200] \times [0, 400]$  and an analytical gain density function stated by

$$g(x, y, z) = 1000 - \sqrt{(x - 600)^2 + (y - 200)^2 + (z - 350)^2} \quad (4.24)$$

that reaches its maximal value in  $(600, 200, 350)$  and which decreases radially from this point. The initial profile used in this instance is showed in Figure 4.8.

We set a discount factor  $\alpha = 0.1$  and solve the 3D SOP for different capacity limits and number of time-frames, using a  $30 \times 10$  discretization of  $\Omega$ . Figures 4.9 and 4.10 illustrate the optimal sequence of

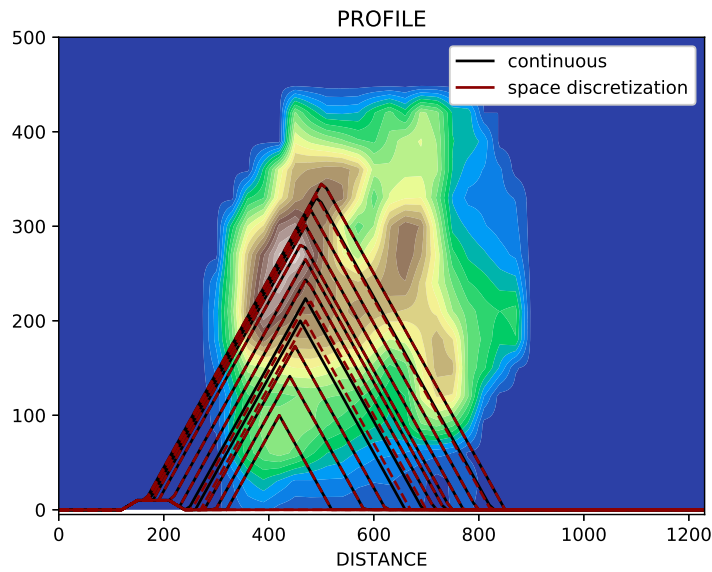


Figure 4.7: 2D SOP: continuous and semi-continuous formulations.

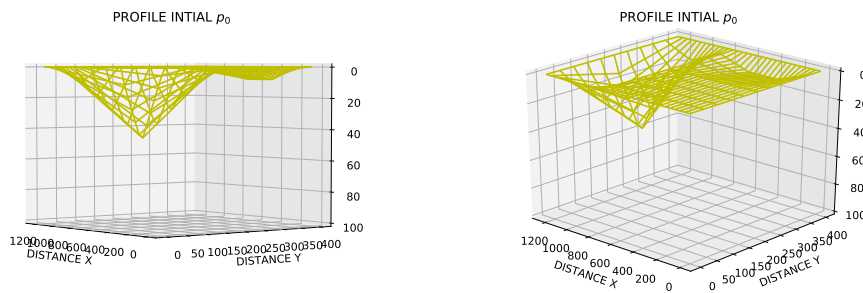


Figure 4.8: Initial profile for the 3D SOP case

3D profile corresponding to  $c_{max} = 10^6$  and  $5 \cdot 10^6$  respectively, with  $T = 2, 3, 6$  time-frames. Table 4.4 shows the objective values and CPU times. Results are consistent overall, with solutions trying to reach the region of highest gain as fast as allowed by the slope and capacity constraints. Increasing the capacity limit and / or the duration of the time interval both yield better objective values, as expected. CPU times are still reasonable, with the longest run at 139s.



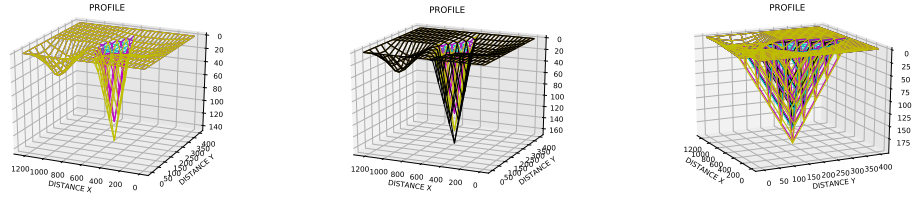


Figure 4.9: 3D profile optimization with limited capacity  $C = 1e6$ , for  $T = 2, 3$  and 6 time-frames.

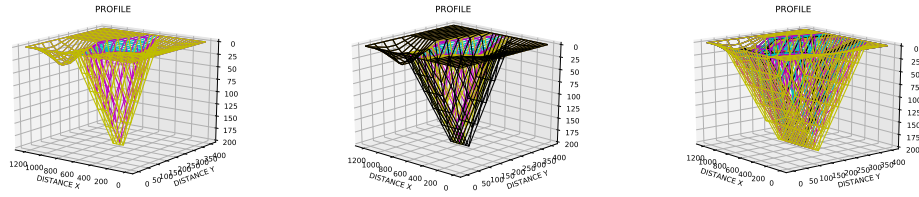


Figure 4.10: 3D profile optimization with limited capacity  $C = 5e6$ , for  $T = 2, 3$  and 6 time-frames.

Times-frames	Capacity limit: $10^6$		Capacity limit : $5 \cdot 10^6$	
	Objective	CPU	Objective	CPU
$T = 2$	69954.88	11s	73089.311	53s
$T = 3$	100103.04	38s	104684.29	124s
$T = 6$	175132.64	31s	183583.57	139s

Table 4.4: 3D SOP: solutions from the semi-continuous formulation, for different time intervals and capacity limit per time-frame.

## 4.5 Conclusions

In the present chapter we focused on the Open Pit problem in an optimal control framework. We extended some previous results on the optimality conditions for the Final Open Pit, and introduced a new semi-continuous formulation that handles the 3D profile sequential optimization. Numerical simulations are provided for the continuous and semi-continuous approaches on several test cases. The 2D FOP case showed a good consistency between global and local optimization for the continuous approach, as well as local optimization for semi-continuous, and matched the optimality conditions from Pontryagin's Principle. Then the 2D SOP case again indicated a good match for the continuous and semi-continuous formulations. Finally we solved a 3D SOP test case, to our knowledge for the first time in an optimal control framework. Perspectives in the continuation of the present work include solving a more complete 3D SOP example using 3D interpolated data for the gain and maximal slope, as well as studying the optimality conditions for the semi-continuous approach. The latter could prepare for the use of indirect shooting methods such as HAMPATH [31], especially since the local optimization method used here can provide the knowledge of the optimal control structure and a costate approximation.

## 4.6 Implementation details for the semi-continuous approach

**Time discretization.** The Sequential Open Pit for the semi-continuous approach described in 4.2.2 is a multi phase problem. Instead of duplicating the variables for each time-frame, we use here in practice a more compact implementation, by using a time step  $\Delta t$  of 1 time-frame, i.e. the time discretization  $t_k = 0 \dots T$  is the sequence of time-frames. This choice makes sense from the operational point of view, since the sequential open pit planning precisely consists in determining the optimal mine profile at each time-frame. It also simplifies a lot the computation of the integrals of the gain and effort functions between two successive mine profiles. We choose an implicit Euler scheme for the time discretization, which gives the trivial discrete dynamics

$$P_i^{k+1} = P_i^k + U_i^{k+1} \quad (4.25)$$

that easily gives the next / previous mine profile when needed in the computations.

**Gain.** An additional state variable  $g$  is added to represent the gain realized along the time-frames, whose dynamics can be written as

$$\dot{g}(t_k) = \frac{1}{(1 + \alpha)^{k-1}} \int_{P^{k-1}}^{P^k} g(x, z) dx dz, \quad \forall k = 1, \dots, T \quad (4.26)$$

The objective is then to maximize  $g(T)$ . For the 2D case, we approximate the 2-dimensional integral of  $g$  by trapezoidal rule over  $x$  then along  $z$ . In the 3D profile case, the 3D integral of  $g$  for the computation of the gain is approximated using a 2D trapezoidal rule along  $(x, y)$  then a standard trapezoidal rule along  $z$ .

**Capacity.** At each time-frame, the integral of the excavation effort over the domain  $\Omega$  can be approximated by

$$\int_{P^{k-1}}^{P^k} e(x, z) dx dz \approx \sum_{i=0}^N \Delta x \left( \int_{P_i^{k-1}}^{P_i^k} e(x_i, z) dz \right) \quad (4.27)$$

Since  $E = 1$  and from the discrete dynamics  $P_i^k = P_i^{k-1} + U_i^k$ , we can use the following formula

$$\int_{P^{k-1}}^{P^k} e(x, z) dx dz \approx \Delta x \sum_{i=0}^{N-1} U_i^k. \quad (4.28)$$

Similarly, for the 3D profile case, the excavation effort at time-frame  $k$  is approximated as

$$\int_{P^{k-1}}^{P^k} e(x, y, z) dx dy dz \approx \Delta x \Delta y \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} U_{i,j}^k. \quad (4.29)$$

## 4.7 Additional examples for the final open pit - continuous formulation

### 4.7.1 FOP with infinite capacity and constant slope

We show here the basic example with unconstrained capacity, namely  $c_{max} = \infty$ . Fig. 4.11 shows the solution obtained by the global method, and Fig.4.12 shows the solution from the local method, and we observe that both solutions match. With infinite capacity, the solution, as expected, digs as much as possible with respect to the maximal slope, until it reaches negative gain. This corresponds to the observed

**Bang-Singular-Bang** control structure (neglecting the two very small constrained arcs  $P = P_0 = 0$  at the extremities). As stated in Lemma 2, the singular arc in the middle follows the geodesic  $g = 0$ . The corresponding control also matches the theoretical expression of the singular control (4.19), despite some oscillations at the junctions with the bang arcs.

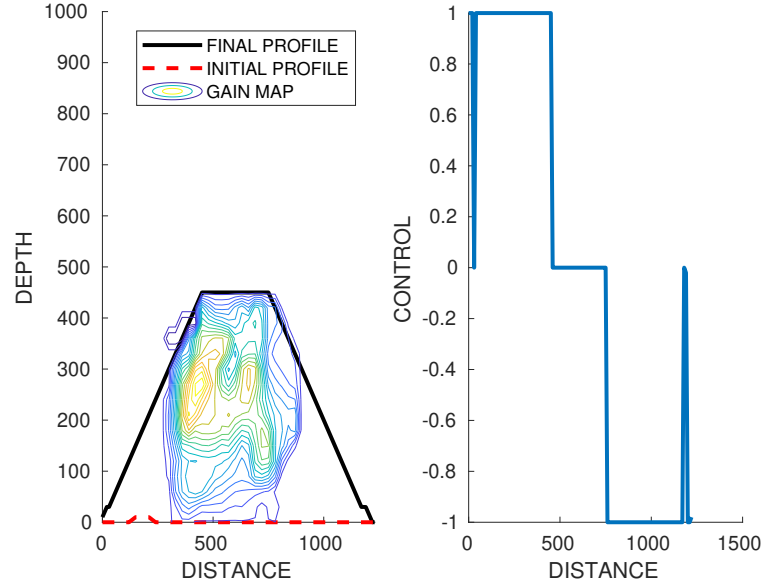


Figure 4.11: 2D profile with infinite capacity - global optimization (HJB method)

## 4.7.2 FOP with infinite capacity and variable slope

Here we illustrate a case with a non-constant maximal slope  $\kappa$ . For this, we consider following arbitrary  $\kappa$  function:

$$\kappa(x, z) = \begin{cases} 0.5 & x \in [0, 330) \\ 1 & x \in [330, 960) \\ 5 & x \in [960, 1230] \end{cases} \quad (4.30)$$

We chose a piece-wise constant function so that the assumption in section 4.3 is satisfied almost everywhere. The solutions from the global and local optimizations are shown in Fig. 4.13 and Fig. 4.14 respectively. This time we obtain a control structure that includes all possible types of arcs: **Bang-Singular-Bang-Constrained**. The main difference compared to the constant slope case is that the optimal profile digs less ground on the right side region where the gain is negative, as a steeper slope is allowed there. As for the previous examples we observe that the singular control and costate from the solution closely match their formal expressions from the PMP.

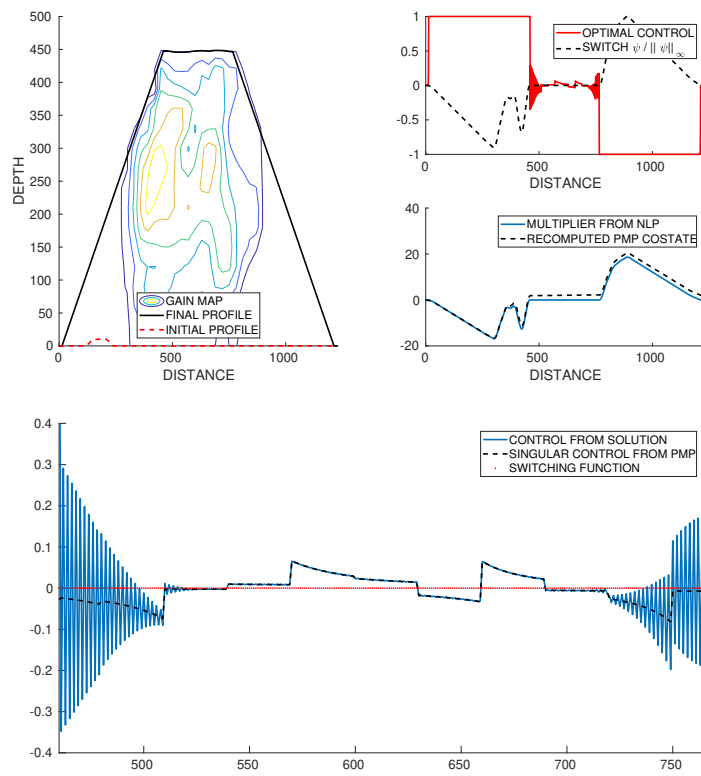


Figure 4.12: 2D profile with infinite capacity - local optimization (direct method) and consistency with PMP optimality condition

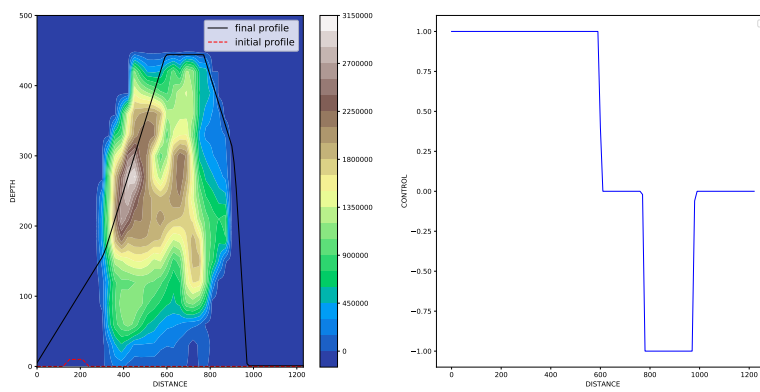


Figure 4.13: 2D profile with infinite capacity and variable maximal slope - global optimization (HJB method)

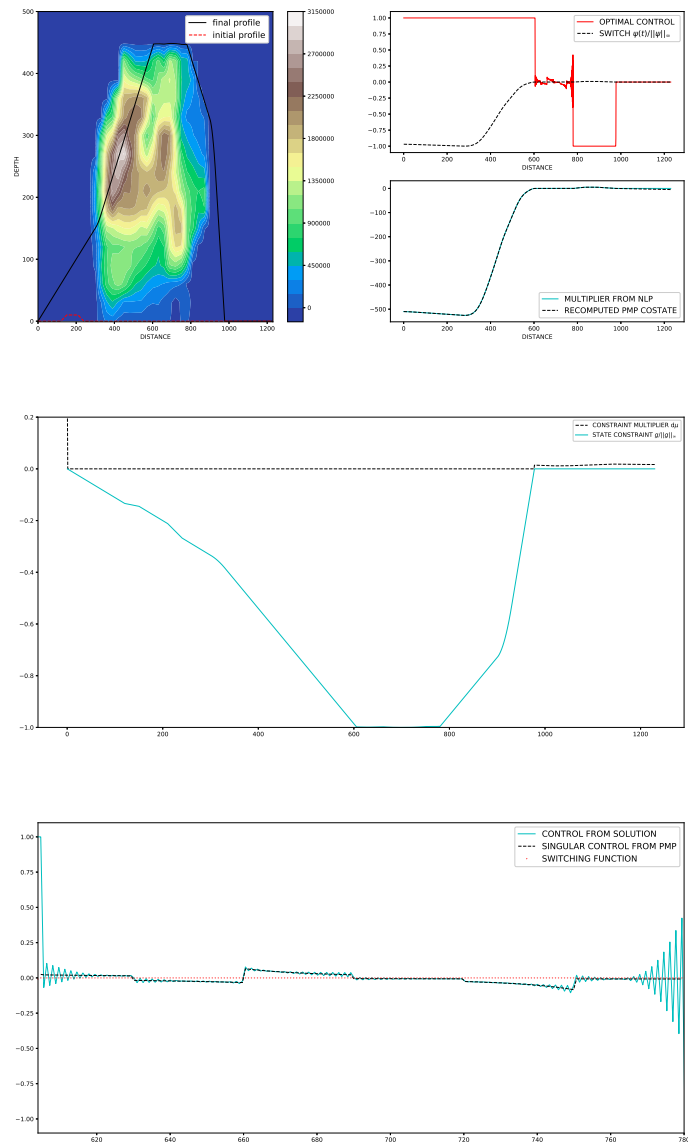


Figure 4.14: 2D profile with infinite capacity and variable maximal slope - local optimization (direct method) with PMP optimality conditions

## **Part II**

# **Problems inspired by Covid-19 peak reduction**



## Chapter 5

# Compartmental models

Since the Covid-19 outbreak, numerous research works have coupled optimal control techniques with epidemiological models for the study and management of the pandemic, such as reducing the number of infected people (see for example [20, 56, 67, 71, 72, 86]). In the context of this PhD, I had the opportunity to contribute to the study of the so-called overcrowding problem, which is a significant issue for hospitals. More precisely, our main objective was to minimize the peak of infections in order to keep the health system below saturation levels.

We worked with the most widespread compartmental model in epidemiology called the SIR model, which we recall in the following section. More specifically we started from the work of Morris et al. [83] on peak minimization for the SIR model, which despite not using optimal control techniques per se, is closely related to control theory in its formulation.

After the classical SIR model we introduce a new compartmental model that takes vaccination into account. This model was developed in the framework of a Pan American Health Organization<sup>1</sup> project by a joint team of mathematicians and doctors devoted to the evaluation of the vaccination campaign of which I was a part of. We hope that this new model can be used in future works.

### 5.1 SIR model

The so-called SIR model introduced in 1927 by Kermack and McKendrick [69] is the most widespread model for the study of directly transmitted infectious diseases that spread through contact from an individual to another, such as, for instance, flu, tuberculosis, MERS-CoV and Covid-19.

The model derives its name from partitioning a given population into three compartments, namely Susceptible, Infected and Recovered individuals. We note respectively  $S(t)$ ,  $I(t)$  and  $R(t)$  the size at time  $t \in [0, T]$  of each group. Many assumption can be made modifying the final model, we consider here the case of a constant population, i.e. without any natural births or deaths. Normalizing the total population, we have the relation  $S(t) + I(t) + R(t) = 1, \forall t \geq 0$ .

The dynamic of the three groups follows the ODE system:

$$\begin{aligned}\dot{S} &= -\beta SI \\ \dot{I} &= \beta SI - \gamma I \\ \dot{R} &= \gamma I,\end{aligned}\tag{5.1}$$

with constants  $\beta > 0$  and  $\gamma > 0$  corresponding to transmission and recovery rates respectively. The so-called **basic reproductive number**  $\mathcal{R}_0$ , defined as the average number of secondary infection cases

---

<sup>1</sup>In Spanish Organización Panamericana de la Salud (OPS)



caused by a single primary case in a susceptible population, is therefore given by  $\mathcal{R}_0 = \beta/\gamma$  in the SIR model. An obvious property of the model is that a disease will spread among the population when  $\mathcal{R}_0 > 1$ , making this case the most relevant for study. We refer readers interested in more properties of SIR models and basic reproductive number to for instance [5, 63, 69, 100].

In order to fight the pandemic, local governments have implemented a variety of measures to reduce the transmission of the disease. These efforts can typically be categorized as non-pharmaceutical, such as lock-downs and quarantines, and pharmaceutical such as vaccines and treatments. The first kind can be introduced in the SIR model as a control variable  $b(t) \in [0, 1]$  applied as a factor to the transmission rate beta:

$$\begin{aligned}\dot{S} &= -b(t)\beta SI \\ \dot{I} &= b(t)\beta SI - \gamma I \\ \dot{R} &= \gamma I\end{aligned}\tag{5.2}$$

In this case,  $b = 1$  corresponds to no intervention, while an identically null control  $b = 0$  will completely contain the infection, however, this is a hardly applicable policy at large scale, since it would amount to cutting off contact between individuals completely. This is why a more realistic constraint is usually imposed on the control, for example, in Chapters 6 and 7 we considered a maximal budget limit  $Q$  imposed on the  $L^1$  norm of the policy intensity  $1 - b$ . This is somewhat less restrictive than the original constraint in [83] that limited the control effectively applied (i.e  $b < 1$ ) to a certain time window  $[t_i, t_i + \tau]$  to be determined. With this constraint, the authors main result on the peak minimization, i.e.  $\min_b \max_{t \in [0, T]} I(t)$ , in [83] was:

**Proposition 5.1.1.** *The optimal intervention which is a solution of  $\min_b \max_{t \in [0, T]} I(t)$  is:*

$$b_{opt}(t) = \begin{cases} \frac{\gamma}{\beta S(t)}, & t \in [t_i, t_i + f\tau) \\ 0, & t \in [t_i + f\tau, t_i + \tau] \end{cases}$$

for a certain  $f \in [0, 1]$ .

This was our starting point to study the peak minimization in the SIR model, and in Chapter 7 we present our results and compare them to this original work.

Regarding pharmaceutical measures, we studied how to include vaccine programs in compartmental models. The result of this research will be presented in the next section and it corresponds to an original model posed by our team.

## 5.2 A new compartmental model including vaccines

We introduce here an extension of the SIR model that includes the effect of vaccination campaigns. This model was developed in the framework of a project supported by the OPS and the MINSAL (Chile ministry of health). The aim of this model is to evaluate the effectiveness of different.

Many extensions of the original SIR model, typically with more compartments, have been proposed, see for instance [6, 9, 53, 88]. The main novelty of the model presented here is the partition of the individuals into three so-called fundamental parts, according to their vaccination status: (i) not fully vaccinated, (ii) fully vaccinated and (iii) fully vaccinated with additional booster injection. This division follows the idea that the disease dynamics are altered by vaccination. Each fundamental part is further modeled with 6 compartments (state variables): susceptible ( $S$ ), asymptomatic infected ( $I^a$ ), symptomatic infected ( $I^s$ ), asymptomatic recovered ( $R^a$ ), symptomatic recovered ( $R^s$ ) and deceased ( $D$ ). To distinguish variables from the three fundamental parts, we add the subscripts  $N$  (non fully vaccinated),  $V$  (fully vaccinated) and  $V_r$  (vaccinated with booster).

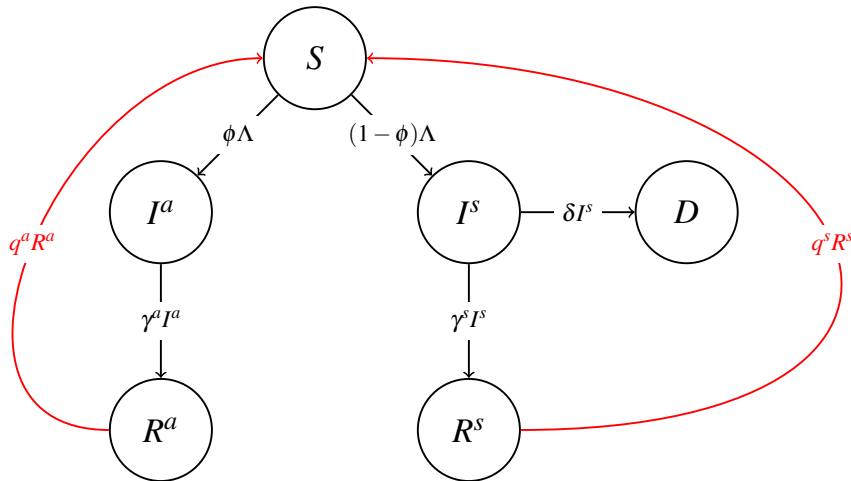


Figure 5.1: Schema of one fundamental part as a 6-compartment model: susceptible ( $S$ ), asymptomatic infected ( $I^a$ ), symptomatic infected ( $I^s$ ), asymptomatic recovered ( $R^a$ ), symptomatic recovered ( $R^s$ ) and dead ( $D$ ). Red arcs represent the loss of immunity after infection.

Figure 5.1 shows the compartments and transitions inside of each fundamental part. The complete model that we will call *base structure* has 18 compartments (6 for each part) as shown in Figure 5.2, with the connections between the three fundamental parts corresponding to vaccination actions.

We suppose the population modeled by the base structure is homogeneous (meaning there is no age structure in the model), isolated, susceptible to only one variant circulating and constant (no natural birth or death). We denote the transmission rates by  $\beta_i^k$  for  $i \in \{N, V, V_R\}$  and  $k \in \{a, s\}$ , using  $a$  for asymptomatic and  $s$  for symptomatic individuals. For example,  $\beta_N^a$  represents the transmission rate after contact with a non-vaccinated, asymptomatic, infected person. The overall transmission rate for the susceptible partition  $S_i$  with  $i \in \{N, V, V_R\}$  will be noted  $\Lambda_i$  defined as:

$$\Lambda_i(t) = (1 - f_i(t)) \left( \sum_{j \in \{N, V, V_R\}} ((1 - u(t))\beta_{i,j}^a I_j^a(t) + (1 - \mu)\beta_{i,j}^s I_j^s(t)) \right) S_i(t) \quad i \in \{N, V, V_R\}.$$

Here  $\mu \in [0, 1]$  represents the portion (assumed constant) of infected, symptomatic population that is completely isolated. The factor  $u(t) \in [0, \mu]$  corresponds to non pharmaceutical measures, similar to  $b(t)$  in the basic SIR model except it only affects now the asymptomatic individuals, since the symptomatic portion are assumed to be in maximum isolation.

We also denote by  $f_i(t)$  the vaccine effectiveness to reduce transmission, being  $f_N(t) \equiv 0$  for the non-fully vaccinated part. Noting  $e_{S_N}(\tau)$ ,  $e_{R_N^a}(\tau)$ ,  $e_{R_N^s}(\tau) \in [0, 1]$  the effectiveness for the compartments ( $S_N$ ), ( $R_N^a$ ) and ( $R_N^s$ ), respectively,  $\tau$  days after becoming fully vaccinated, and  $v_{S_N}(t)$ ,  $v_{R_N^a}(t)$  and  $v_{R_N^s}(t)$  the vaccination rates for each compartment, then the overall effectiveness  $f_i$  is modeled similarly as in [88] by the equation:

$$f_V(t) = \frac{\int_0^{t-t_V} \left( e_{S_N}(\tau) v_{S_N}(t-\tau) + e_{R_N^a}(\tau) v_{R_N^a}(t-\tau) + e_{R_N^s}(\tau) v_{R_N^s}(t-\tau) \right) d\tau}{\int_0^{t-t_V} v_{S_i}(s) + v_{R_i^a}(s) + v_{R_i^s}(s) ds} \quad t \geq t_V,$$

where  $t_V$  is the start of the vaccination plan.

Similarly, for a booster we note  $r_{S_V}(\tau)$ ,  $r_{R_V^a}(\tau)$ ,  $r_{R_V^s}(\tau) \in [0, 1]$  the effectiveness in  $(S_V)$ ,  $(R_V^a)$  and  $(R_V^s)$ , respectively,  $\tau$  days after applying the booster dose and  $v_{S_V}(t)$ ,  $v_{R_V^a}(t)$  and  $v_{R_V^s}(t)$  the booster injection rates for each relevant compartment, then the total effectiveness of a booster dose is given by:

$$f_{V_r}(t) = \frac{\int_0^{t-t_V} \left( r_{S_V}(\tau) v_{S_V}(t-\tau) + r_{R_V^a}(\tau) v_{R_V^a}(t-\tau) + r_{R_V^s}(\tau) v_{R_V^s}(t-\tau) \right) d\tau}{\int_0^{t-t_V} \hat{v}_V(t)(s) ds} \quad t \geq t_V,$$

Finally we describe the remaining parameters of the model:

- **Recovery rates:** We denote  $\gamma_i^k$ , with  $i \in \{N, V, V_r\}$  and  $k \in \{a, s\}$  the recovery rates of infected individuals who are in the fundamental part  $i$  and could be symptomatic ( $s$ ) or asymptomatic ( $a$ ). A natural assumption is that recovery should be quicker for vaccinated people (as reported for example in [78]), leading to the following inequalities

$$\gamma_N^k \leq \gamma_V^k \leq \gamma_{V_r}^k \quad k \in \{a, s\}.$$

- **Death rates:** We denote  $\delta_i$ , with  $i \in \{N, V, V_r\}$  the death rates in the infected population in fundamental part  $i$ . Once again assuming vaccination reduces the mortality (as the evidence show in [8, 77, 68, 89]), we have the inequalities

$$\delta_{V_r} \leq \delta_V \leq \delta_N.$$

- **Loss of immunity rates :** The rate of loss of immunity for a recovered individual will be denoted by  $q_i^k$  with  $i \in \{N, V, V_r\}$  and  $k \in \{a, s\}$ .

- **Probability of being asymptomatic:** The probability of a susceptible individual, belonging to the fundamental part  $i \in \{N, V, V_r\}$ , to be asymptomatic after infection will be denoted by  $\phi_i \in [0, 1]$ . These probabilities should fulfill

$$\phi_N \leq \phi_V \leq \phi_{V_r},$$

because the probability of being asymptomatic should increase as vaccination advances as it is shown in [1, 94].

Table 5.1 shows a summary of the parameters used in this model.

Notation	Description
$\beta_i^k$	Transmission rates, $i \in \{N, V, V_R\}$ and $k \in \{a, s\}$ .
$\mu$	Portion of population which is infected, symptomatic and is completely isolated
$u(t)$	Factor in transmission rate due to non pharmaceutical measures.
$f_i(t)$	Effectiveness of vaccines in transmission, $i \in \{N, V, V_R\}$
$\gamma_i^k$	Recovery rates of infected individuals, $i \in \{N, V, V_r\}$ and $k \in \{a, s\}$
$\delta_i$	Death rates, $i \in \{N, V, V_r\}$ .
$q_i^k$	Rate of loss of immunity for a recovered individual, $i \in \{N, V, V_r\}$ and $k \in \{a, s\}$ .
$\phi_i$	Probability of being asymptomatic, $i \in \{N, V, V_r\}$

Table 5.1: Parameters used in the model.

**Fundamental part (N), non vaccinated:**

$$\begin{aligned}
\dot{S}_N(t) &= -\Lambda_N(t) + q_N^a R_N^a(t) + q_N^s R_N^s(t) - \nu_{S_N}(t) \\
\dot{I}_N^a(t) &= \phi_N \Lambda_N(t) - \gamma_N^a I_N^a(t) \\
\dot{I}_N^s(t) &= (1 - \phi_N) \Lambda_N(t) - (\gamma_N^s + \delta_N) I_N^s(t) \\
\dot{R}_N^a(t) &= \gamma_N^a I_N^a(t) - q_N^a R_N^a(t) - \nu_{R_N^a}(t) \\
\dot{R}_N^s(t) &= \gamma_N^s I_N^s(t) - q_N^s R_N^s(t) - \nu_{R_N^s}(t) \\
\dot{D}_N(t) &= \delta_N I_N^s(t);
\end{aligned}$$

**Fundamental part (V), vaccinated without booster:**

$$\begin{aligned}
\dot{S}_V(t) &= -\Lambda_V(t) + q_V^a R_V^a(t) + q_V^s R_V^s(t) + \hat{\nu}_V(t) - \nu_{S_V}(t) \\
\dot{I}_V^a(t) &= \phi_V \Lambda_V(t) - \gamma_V^a I_V^a(t) \\
\dot{I}_V^s(t) &= (1 - \phi_V) \Lambda_V(t) - (\gamma_V^s + \delta_V) I_V^s(t) \\
\dot{R}_V^a(t) &= \gamma_V^a I_V^a(t) - q_V^a R_V^a(t) - \nu_{R_V^a}(t) \\
\dot{R}_V^s(t) &= \gamma_V^s I_V^s(t) - q_V^s R_V^s(t) - \nu_{R_V^s}(t) \\
\dot{D}_V(t) &= \delta_V I_V^s(t);
\end{aligned}$$

**Fundamental part (V<sub>r</sub>), vaccinated with booster:**

$$\begin{aligned}
\dot{S}_{V_r}(t) &= -\Lambda_{V_r}(t) + q_{V_r}^a R_{V_r}^a(t) + q_{V_r}^s R_{V_r}^s(t) + \hat{\nu}_{V_r}(t) \\
\dot{I}_{V_r}^a(t) &= \phi_{V_r} \Lambda_{V_r}(t) - \gamma_{V_r}^a I_{V_r}^a(t) \\
\dot{I}_{V_r}^s(t) &= (1 - \phi_{V_r}) \Lambda_{V_r}(t) - (\gamma_{V_r}^s + \delta_{V_r}) I_{V_r}^s(t) \\
\dot{R}_{V_r}^a(t) &= \gamma_{V_r}^a I_{V_r}^a(t) - q_{V_r}^a R_{V_r}^a(t) \\
\dot{R}_{V_r}^s(t) &= \gamma_{V_r}^s I_{V_r}^s(t) - q_{V_r}^s R_{V_r}^s(t) \\
\dot{D}_{V_r}(t) &= \delta_{V_r} I_{V_r}^s(t).
\end{aligned}$$

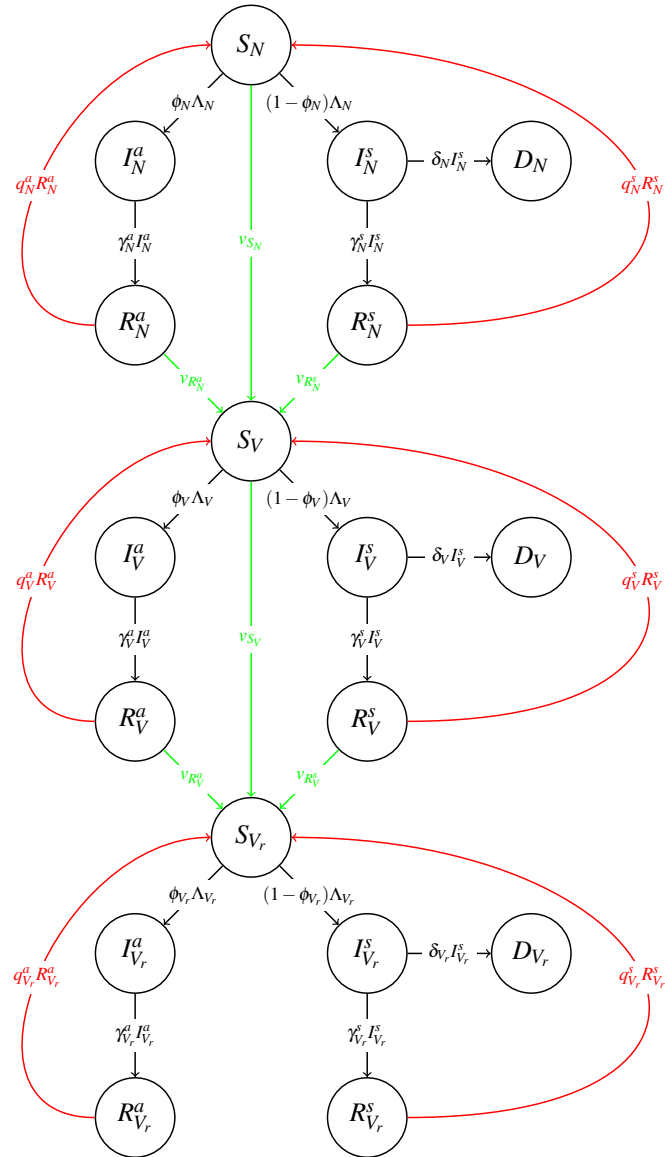


Figure 5.2: Base structure with 3 fundamental parts and 18 compartments in total. Green arcs indicate vaccination actions that link the three parts, red arcs correspond to loss of immunity.

## Chapter 6

# A Mayer formulation for peak minimization problems

This chapter corresponds to the accepted pre-print [82] titled "Equivalent formulations of optimal control problems with maximum cost and applications" to appear in Journal of Optimization Theory and Applications.

### 6.1 Introduction

We consider the optimal control problem which consists in minimizing the maximum of a scalar function over a time interval

$$\inf_{u(\cdot)} \sup_{t \in [t_0, T]} y(t)$$

where  $y(t) = \theta(t, \xi(t))$  and  $\xi(\cdot)$  is the solution of a controlled dynamics  $\dot{\xi} = \phi(\xi, u)$ ,  $\xi(t_0) = \xi_0$ . This problem is not in the usual Mayer, Lagrange or Bolza forms of the optimal control theory, and therefore is not suitable to use the classical necessary optimality conditions of Pontryagin Maximum Principle or existing solving algorithms (based on direct method, shooting or Hamilton-Bellman Jacobi equation). However, this problem falls into the class of optimal control with  $L^\infty$  criterion, for which several characterizations of the value function have been proposed in the literature [15, 16, 54]. Typically, the value function is solution, in a general sense, of a variational inequality of the form

$$\min \left( \partial_t V + \inf_u \partial_\xi V \cdot \phi(x, u), V - \theta \right) = 0$$

without boundary condition. Nevertheless, although necessary optimality conditions and numerical procedures have been formulated [14, 40, 41, 51], there is no practical numerical tool to solve such problems as it exists for Mayer problems, to the best of our knowledge. The aim of the present work is to study different reformulations of this problem into Mayer form in higher dimension with possibly state or mixed constraints, for which existing numerical methods can be used. Indeed, it has already been underlined in the literature that discrete-time optimal control problems with maximum cost do not satisfy the Principle of Optimality but can be transformed into problems of higher dimension with additively separable objective functions [65, 64]. We pursue here this idea but in the continuous time framework, which faces the lack of differentiability of the max function.

This manuscript is organized as follows. In Section 6.2, we establish the setup and the hypotheses of this article, and define the problem. In Section 6.3, we provide equivalent formulations of the stud-

ied problem in the form of two Mayer problems with fixed initial condition, and under state or mixed constraint. In Section 6.4, we propose another formulation in terms of differential inclusion but without constraints, and then we show how the optimal value can be approximated from below by a sequence of more regular Mayer problems. Section 6.5 is devoted to numerical illustrations. We consider two problems for which the optimal solution can be determined explicitly (one borrowed from epidemiology), which allows to estimate and compare the numerical performances of the different formulations. These problems have been chosen linear with respect to the control variable in order to present discontinuous optimal controls, which are known to be numerically more sensitive. More precisely, the optimal solution of the first problem has pure bang-bang controls, while the second one possesses a singular arc. We discuss the issues arising in the numerical implementations of the different formulations, and also compare numerically with  $L^p$  approximations. Finally, we discuss in Section 6.6 about the potential merits of the different formulations as practical methods to compute optimal solution of  $L^\infty$  control problems.

## 6.2 Problem and hypotheses

We shall consider autonomous dynamical systems defined on a invariant domain  $\mathcal{D}$  of  $\mathbb{R}^{n+1}$  of the form

$$\begin{cases} \dot{x} = f(x, y, u) \\ \dot{y} = g(x, y, u) \end{cases} \quad (6.1)$$

(where  $g$  is a scalar function) where the values of the control  $u(\cdot)$  belong to a given set  $U \subset \mathbb{R}^p$ . More specifically, throughout the chapter, we shall assume that the following properties are fulfilled.

### Assumption 1.

- i.  $U$  is a compact set.
- ii. The maps  $f$  and  $g$  are  $C^1$  on  $\mathcal{D} \times U$ .
- iii. The maps  $f$  and  $g$  have linear growth, that is there exists a number  $C > 0$  such that

$$\|f(x, y, u)\| + |g(x, y, u)| \leq C(1 + \|x\| + |y|), \quad (x, y) \in \mathcal{D}, \quad u \in U$$

For instance,  $y(\cdot)$  can be a smooth output of a dynamics

$$\dot{x} = f(x, u), \quad y = h(x)$$

which can be rewritten as

$$\begin{cases} \dot{x} = f(x, u) \\ \dot{y} = g(x, u) := \nabla h(x)^T \cdot f(x, u) \end{cases}$$

Let  $\mathcal{U}$  be the set of measurable functions  $u(\cdot) : [0, T] \mapsto U$  and consider  $(x_0, y_0) \in \mathcal{D}$ ,  $T > 0$ . Under the usual arguments of the theory of ordinary differential equations, Assumption 1 ensures that for any  $u(\cdot) \in \mathcal{U}$  there exists a unique absolutely continuous solution  $(x(\cdot), y(\cdot))$  of (6.1) on  $[0, T]$  for the initial condition  $(x(0), y(0)) = (x_0, y_0)$  (see for instance [48]). Define then the solutions set

$$\mathcal{S} := \{(x(\cdot), y(\cdot)) \in \mathcal{A}\mathcal{C}([0, T], \mathbb{R}^{n+1}), \text{ sol. of (6.1) for } u(\cdot) \in \mathcal{U} \text{ with } (x(0), y(0)) = (x_0, y_0)\}.$$

We consider then the optimal control problem which consists in minimizing the "peak" of the function  $y(\cdot)$ :

$$\mathcal{P} : \inf_{u(\cdot) \in \mathcal{U}} \left( \max_{t \in [0, T]} y(t) \right) = \inf_{(x(\cdot), y(\cdot)) \in \mathcal{S}} \left( \max_{t \in [0, T]} y(t) \right).$$

### 6.3 Formulations with constraint

A first approach considers the family of constrained sets of solutions

$$\mathcal{S}_z := \{(x, y) \in \mathcal{S}, y(t) \leq z, t \in [0, T]\}, \quad (z \in \mathbb{R})$$

and to look for the optimization problem

$$\inf\{z; \mathcal{S}_z \neq \emptyset\}.$$

This problem can be reformulated as a Mayer problem

$$\mathcal{P}_0 : \inf_{u(\cdot) \in \mathcal{U}} z(T)$$

for the extended dynamics in  $\mathcal{D} \times \mathbb{R}$

$$\begin{cases} \dot{x} = f(x, y, u) \\ \dot{y} = g(x, y, u) \\ \dot{z} = 0 \end{cases}$$

under the state constraint

$$\mathcal{C} : z(t) - y(t) \geq 0, t \in [0, T]$$

where  $z(0)$  is free. Direct methods can be used for such a problem. However, as  $z(0)$  is free, solutions are not sought among solutions of a Cauchy problem, which prevents using other methods based on dynamic programming such as the Hamilton-Jacobi-Bellman equation.

We propose another extended dynamics in  $\mathcal{D} \times \mathbb{R}$  with an additional control  $v(\cdot)$  with values in  $[0, 1]$

$$\begin{cases} \dot{x} = f(x, y, u) \\ \dot{y} = g(x, y, u) \\ \dot{z} = \max(g(x, y, u), 0)(1 - v) \end{cases} \quad (6.2)$$

Let  $\mathcal{V}$  be the set of measurable functions  $v : [0, T] \mapsto [0, 1]$ . Note that under Assumption 1, for any  $(x_0, y_0, z_0) \in \mathcal{D} \times \mathbb{R}$  and  $(u, v) \in \mathcal{U} \times \mathcal{V}$ , there exists a unique absolutely solution  $(x(\cdot), y(\cdot), z(\cdot))$  of (6.2) on  $[0, T]$  for the initial condition  $(x(0), y(0), z(0)) = (x_0, y_0, z_0)$ . Here, we fix the initial condition with  $z_0 = y_0$  and consider the Mayer problem

$$\mathcal{P}_1 : \inf_{(u(\cdot), v(\cdot)) \in \mathcal{U} \times \mathcal{V}} z(T) \quad \text{under the constraint } \mathcal{C}$$

and shows its equivalence with problem  $\mathcal{P}$ . We first consider fixed controls  $u(\cdot)$ .

**Proposition 6.3.1.** *For any control  $u(\cdot) \in \mathcal{U}$ , the optimal control problem*

$$\inf_{v \in \mathcal{V}} z(T) \quad \text{under the constraint } \mathcal{C} \quad (6.3)$$

*admits an optimal solution. Moreover, an optimal solution verifies*

$$z(T) = \max_{t \in [0, T]} y(t), \quad (6.4)$$

*and is reached for a control  $v(\cdot)$  that takes values in  $\{0, 1\}$ .*



*Proof.* From equations (6.2), one get that any solution  $z(\cdot)$  is non decreasing, and as  $z$  satisfies the constraint  $z \geq y$ , we deduce that one has

$$z(T) \geq \max_{t \in [0, T]} y(t) \quad (6.5)$$

for any solution of (6.2), and thus

$$\max_{t \in [0, T]} y(t) \leq \inf_{v \in \mathcal{V}} z(T) \text{ under the constraint } z(t) \geq y(t), t \in [0, T].$$

Let  $x(\cdot), y(\cdot)$  be the solution of (6.1) for the control  $u(\cdot)$  and let  $I$  be the set of *invisible points from the left* of  $y$ , that is

$$I := \{t \in (0, T); y(t') > y(t) \text{ for some } t' < t\}.$$

Consider then the control

$$v(t) = \begin{cases} 1, & t \in \text{int} I, \\ 0, & t \notin \text{int} I \end{cases} \quad (6.6)$$

When  $I$  is empty,  $y(\cdot)$  is a non decreasing function and, when  $v(t) = 0$  for all  $t \in [0, T]$ , one has  $z(t) = y(t)$  for any  $t \in [0, T]$ . Therefore, one has

$$z(T) = y(T) = \max_{t \in [0, T]} y(t).$$

When  $I$  is non empty, there exists, from the sun rising Lemma [95], a countable set of disjoint non-empty intervals  $I_n = (a_n, b_n)$  of  $[0, T]$  such that

- the interior of  $I$  is the union of the intervals  $I_n$ ,
- one has  $y(a_n) = y(b_n)$  if  $b_n \neq T$ ,
- if  $b_n = T$ , then  $y(a_n) \geq y(b_n)$ .

Note that when  $t \notin \text{int} I$ , one has  $y(t) \geq y(t')$  for any  $t' \leq t$ . Therefore, the solution  $z$  with control (6.6) verifies

$$z(t) = \begin{cases} y(t), & t \notin \text{int} I \\ y(a_n), & t \in I_n \text{ for some } n \end{cases}$$

(see Figure 6.1 as an illustration). Let  $\bar{t} \in [0, T]$  be such that

$$y(\bar{t}) = \max_{t \in [0, T]} y(t),$$

which implies that any point  $t' > \bar{t}$  in  $[0, T]$  is invisible from the left. Then, one has  $z(T) = z(\bar{t}) \leq y(\bar{t})$ . Thus, from (6.5), we obtain

$$\max_{t \in [0, T]} y(t) = z(T)$$

and deduce

$$\max_{t \in [0, T]} y(t) = \inf_{v(\cdot) \in \mathcal{V}} z(T) \text{ under the constraint } \mathcal{C}.$$

□

**Remark 6.3.1.** *The proof of Proposition 6.3.2 gives an optimal construction of  $z(\cdot)$  which is the lower envelope of non decreasing continuous functions above the function  $y(\cdot)$ , as depicted on Figure 6.1.*

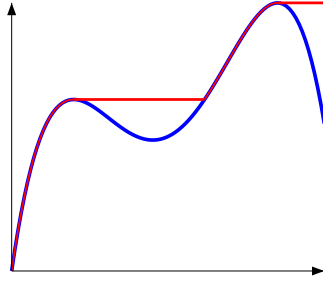


Figure 6.1: Illustration of the function  $z$  (in red) corresponding to a function  $y$  (in blue) with the control given by expression (6.6)

However, there is no uniqueness of the optimal control  $v(\cdot)$ . Any admissible solution  $z(\cdot)$  that is above  $y(\cdot)$  and such that  $z(t) = \hat{y}$  for  $t \geq \hat{t} = \min\{t \in (0, T], y(t) = \hat{y}\}$ , where  $\hat{y} := \max_{s \in [0, T]} y(s)$ , is also optimal.

We then obtain the equivalence between problems  $\mathcal{P}_1$  and  $\mathcal{P}$  in the following sense.

**Proposition 6.3.2.** *If  $(u^*(\cdot), v^*(\cdot))$  is optimal for Problem  $\mathcal{P}_1$ , then  $u^*(\cdot)$  is optimal for Problem  $\mathcal{P}$ . Conversely, if  $u^*(\cdot)$  is optimal for Problem  $\mathcal{P}$ , then  $(u^*(\cdot), v^*(\cdot))$  is optimal for Problem  $\mathcal{P}_1$  where  $v^*(\cdot)$  is optimal for the problem (6.3) for the fixed control  $u^*(\cdot)$ .*

Let us give another equivalent Mayer problem but with a mixed constraint (this will be useful in the next section). We consider again the extended dynamics (6.2), with a control  $v(\cdot)$  which values belong to  $[0, 1]$  and the initial conditions  $(x(0), y(0), z(0)) = (x_0, y_0, y_0)$ . Define then the mixed constraint

$$\mathcal{C}_m : \quad \max(y(t) - z(t), 0)(1 - v(t)) + z(t) - y(t) \geq 0, \quad \text{a.e. } t \in [0, T]$$

and the optimal control problem

$$\mathcal{P}_2 : \quad \inf_{(u(\cdot), v(\cdot)) \in \mathcal{U} \times \mathcal{V}} z(T) \quad \text{under the constraint } \mathcal{C}_m.$$

**Proposition 6.3.3.** *Problems  $\mathcal{P}_1$  and  $\mathcal{P}_2$  are equivalent.*

*Proof.* One can immediately see that for any admissible solution that satisfies constraint  $\mathcal{C}$ , the constraint  $\mathcal{C}_m$  is necessarily fulfilled as  $\max(y - z, 0)$  is identically null.

Conversely, fix an admissible control  $u(\cdot)$  and consider a control  $v(\cdot)$  that satisfies  $\mathcal{C}_m$ . We show that this implies that the solution  $(y(\cdot), z(\cdot))$  verifies necessarily  $z(t) \geq y(t)$  for any  $t \in [0, T]$ . If not, consider the non-empty set

$$E := \{t \in [0, T]; z(t) - y(t) < 0\},$$

which is open as  $z - y$  is continuous. Note that one has  $\dot{z}(t) - \dot{y}(t) \geq 0$  for a.e.  $t \in E$ . Therefore  $z - y$  is non decreasing in  $E$  and we deduce that for any  $t \in E$ , the interval  $[0, t]$  is necessarily included in  $E$ , which then contradicts the initial condition  $z(0) = y(0)$ .  $\square$

## 6.4 Formulation without state constraints

We posit  $\Pi = (x, y, z) \in \mathcal{D} \times \mathbb{R}$  and consider the differential inclusion

$$\dot{\Pi} \in F(\Pi) := \bigcup_{(u,v) \in U \times [0,1]} \begin{bmatrix} f(x, y, u) \\ g(x, y, u) \\ h(x, y, z, u, v) \end{bmatrix} \quad (6.7)$$

with

$$h(x, y, z, u, v) = \max(g(x, y, u), 0)(1 - \mathbb{1}_{\mathbb{R}^+}(z - y))$$

where  $\mathbb{1}_{\mathbb{R}^+}$  is the indicator function

$$\mathbb{1}_{\mathbb{R}^+}(\zeta) = \begin{cases} 1, & \zeta \geq 0 \\ 0, & \zeta < 0 \end{cases}$$

Let  $\Pi_0 = (x_0, y_0, z_0)$  and denote by  $\mathcal{S}_\ell$  the set of absolutely continuous solutions of (6.7) with  $\Pi(0) = \Pi_0 \in \mathcal{D} \times \mathbb{R}$ . We consider the Mayer problem

$$\mathcal{P}_3 : \quad \inf_{\Pi(\cdot) \in \mathcal{S}_\ell} z(T).$$

**Assumption 2.**

$$\forall (x, y) \in \mathcal{D}, \quad G(x, y) := \bigcup_{u \in U} \begin{bmatrix} f(x, y, u) \\ g(x, y, u) \end{bmatrix} \text{ is convex.}$$

**Proposition 6.4.1.** *Under Assumption 2, problem  $\mathcal{P}_3$  admits an optimal solution. Moreover, any optimal solution  $\Pi(\cdot) = (x(\cdot), y(\cdot), z(\cdot))$  verifies*

$$z(T) = \max_{t \in [0, T]} y(t)$$

with  $(x(\cdot), y(\cdot))$  solution of (6.1) for some control  $u(\cdot) \in \mathcal{U}$ , that in turn is optimal for problem  $\mathcal{P}$ .

*Proof.* We fix the initial condition  $\Pi(0) = \Pi_0$  and consider the augmented dynamics

$$\dot{\Pi} \in F^\dagger(\Pi) := \bigcup_{(u,v,\alpha) \in U \times [0,1]^2} \begin{bmatrix} f(x, y, u) \\ g(x, y, u) \\ h^\dagger(x, y, z, u, v, \alpha) \end{bmatrix} \quad (6.8)$$

with

$$h^\dagger(x, y, z, u, v, \alpha) = (1 - \alpha)h(x, y, z, u, v) + \alpha \max_{w \in U} h(x, y, z, w, 0).$$

Under Assumption 2, the values of  $F^\dagger$  are convex compact. One can straightforwardly check that the set-valued map  $F^\dagger$  is upper semi-continuous<sup>1</sup> with linear growth. Therefore, the reachable set  $\mathcal{S}_\ell^\dagger(T)$  (where  $\mathcal{S}_\ell^\dagger$  denotes the set of absolutely continuous solutions of (6.8) with  $\Pi(0) = \Pi_0$ ) is compact (see for instance [11, Proposition 3.5.5]). Then, there exists a solution  $\Pi^*(\cdot) = (x^*(\cdot), y^*(\cdot), z^*(\cdot))$  of (6.8) which minimizes  $z(T)$ .

<sup>1</sup>A set-valued map  $F : \mathcal{X} \rightsquigarrow \mathcal{X}$  is upper semi-continuous at  $\xi \in \mathcal{X}$  if and only if for any neighborhood  $\mathcal{N}$  of  $F(\xi)$ , there exists  $\eta > 0$  such that for any  $\xi' \in B_{\mathcal{X}}(\xi, \eta)$  one has  $F(\xi') \subset \mathcal{N}$  (see for instance [11]).

Note that any admissible solution  $(x(\cdot), y(\cdot), z(\cdot))$  of system (6.2) that satisfies the constraint  $\mathcal{C}_m$  belongs to  $\mathcal{S}_\ell \subset \mathcal{S}_\ell^\dagger$ . We then get the inequality

$$z^*(T) \leq \inf\{z(T); (x(\cdot), y(\cdot), z(\cdot)) \text{ sol. of (6.2) with } \mathcal{C}_m\}. \quad (6.9)$$

Let us show that any solution  $\Pi(\cdot) = (x(\cdot), y(\cdot), z(\cdot))$  in  $\mathcal{S}_\ell$  verifies

$$z(T) \geq \max_{t \in [0, T]} y(t) \quad (6.10)$$

We show that one has  $z(t) \geq y(t)$  for any  $t \in [0, T]$ . We proceed by contradiction, as in the proof of Proposition 6.3.3. If the set  $E = \{t \in (0, T); z(t) - y(t) < 0\}$  is non-empty, one has  $\dot{z}(t) - \dot{y}(t) \geq 0$  for a.e.  $t \in E$  which implies, by continuity, that  $z(s) - y(s) < 0 \forall s \in (0, t), t \in E$  and then  $z(0) - y(0) < 0$  which contradicts the initial condition  $z(0) = y(0)$ . Moreover, as the map  $h$  is non-negative,  $z(\cdot)$  is non decreasing and we conclude that (6.10) is verified.

On another hand, thanks to Assumptions 1 and 2, we can apply Filippov's Lemma to the set-valued map  $G$ , which asserts that  $(x(\cdot), y(\cdot))$  is solution of (6.1) for a certain  $u(\cdot) \in \mathcal{U}$ . With (6.10), we obtain

$$z^*(T) \geq \max_{t \in [0, T]} y^*(t) \geq \inf_{u \in \mathcal{U}} \left\{ \max_{t \in [0, T]} y(t); (x(\cdot), y(\cdot)) \text{ sol. of (6.1)} \right\} \quad (6.11)$$

where  $(x^*(\cdot), y^*(\cdot))$  is solution of (6.1) for a certain  $u^*(\cdot) \in \mathcal{U}$ .

Finally, inequalities (6.9) and (6.11) with Propositions 6.3.2 and 6.3.3 show that  $z^*(T)$  is reached by a solution of (6.2) under the constraint  $\mathcal{C}_m$ , and that  $u^*(\cdot)$  is optimal for problem  $\mathcal{P}$ . We also conclude that the optimal value  $z^*(T)$  is reached by a solution in  $\mathcal{S}_\ell$ , which is thus optimal for problem  $\mathcal{P}_3$ .  $\square$

**Remark 6.4.1.** *Let us stress that the function  $h$  is not continuous, which does not allow to use Filippov's Lemma for the set valued map  $F$ . This means that one cannot guarantee a priori that an absolutely continuous solution  $\Pi(\cdot) = (x(\cdot), y(\cdot), z(\cdot))$  can be synthesized by a measurable control  $(u(\cdot), v(\cdot))$ . Proposition 6.4.1 shows that  $(x(\cdot), y(\cdot))$  is indeed a solution of system (6.1) for a measurable control  $u(\cdot)$ , but one cannot guarantee a priori that  $z(\cdot)$  can be generated by a measurable control  $v(\cdot)$ , which is irrelevant for our purpose.*

We end this section by exhibiting an approximation scheme from below of the optimal cost. These approaches are of major interest for minimization problems because, since upper bounds are commonly obtained via any sub-optimal control of problem  $\mathcal{P}_0$ ,  $\mathcal{P}_1$ ,  $\mathcal{P}_2$  or  $\mathcal{P}_3$  (provided typically by a numerical scheme), they are useful to frame the optimal value of the problem. This will be illustrated in Section 6.5.

Let us consider the family of dynamics parameterized by  $\theta > 0$

$$\begin{cases} \dot{x} = f(x, y, u) \\ \dot{y} = g(x, y, u) \\ \dot{z} = h_\theta(x, y, z, u, v) \end{cases} \quad (6.12)$$

with

$$h_\theta(x, y, z, u, v) = \max(g(x, y, u), 0)(1 - v e^{-\theta \max(y-z, 0)}).$$

Here the expression  $e^{-\theta \max(y-z, 0)}$  plays the role of an approximation of  $\mathbb{1}_{\mathbb{R}^+}(z-y)$  when  $\theta$  tends to  $+\infty$ .

We then define the family of Mayer problems

$$\mathcal{P}_3^\theta : \inf_{\Pi(\cdot) \in \mathcal{S}_\theta} z(T)$$

where  $\mathcal{S}_\theta$  denotes the set of absolutely continuous solutions  $\Pi(\cdot) = (x(\cdot), y(\cdot), z(\cdot))$  of (6.12) for the

initial condition  $\Pi(0) = \Pi_0$ . Let us underline that, for problems with Lipschitz dynamics and without state constraints, necessary conditions based on Pontryagin Maximum Principle can be derived, leading to shooting methods that are known to be very accurate. They can be initialized from numerical solutions of problems  $\mathcal{P}_1$  or  $\mathcal{P}_2$ , that in turn can be obtained, for instance, through direct methods.

**Proposition 6.4.2.** *Under Assumption 2, for any increasing sequence of numbers  $\theta_n$  ( $n \in \mathbb{N}$ ) that tends to  $+\infty$ , the problem  $\mathcal{P}_3^{\theta_n}$  admits an optimal solution, and for any sequence of optimal solutions  $(x_n(\cdot), y_n(\cdot), z_n(\cdot))$  of  $\mathcal{P}_3^{\theta_n}$ , the sequence  $(x_n(\cdot), y_n(\cdot))$  converges, up to sub-sequence, uniformly to an optimal solution  $(x^*(\cdot), y^*(\cdot))$  of Problem  $\mathcal{P}$ , and its derivatives weakly to  $(\dot{x}^*(\cdot), \dot{y}^*(\cdot))$  in  $L^2$ . Moreover,  $z_n(T)$  is an increasing sequence that converges to  $\max_{t \in [0, T]} y^*(t)$ .*

*Proof.* As in the proof of Proposition 6.4.1, we consider for any  $\theta > 0$  the convexified dynamics

$$\begin{cases} \dot{x} = f(x, y, u) \\ \dot{y} = g(x, y, u) \\ \dot{z} = h_\theta^\dagger(x, y, z, u, v, \alpha) := (1 - \alpha)h_\theta(x, y, z, u, v) + \alpha \max_{w \in U} h_\theta(x, y, z, w, 0) \end{cases}$$

where  $\alpha \in [0, 1]$ . Then, there exists an absolutely continuous solution  $(x_\theta^*(\cdot), y_\theta^*(\cdot), z_\theta^*(\cdot))$  and a measurable control  $(u_\theta^*(\cdot), v_\theta^*(\cdot), \alpha_\theta^*(\cdot))$  that minimize  $z(T)$ . For the control  $(u_\theta^*(\cdot), v_\theta^*(\cdot), 0)$ , the solution is given by  $(x_\theta^*(\cdot), y_\theta^*(\cdot), \tilde{z}_\theta^*(\cdot))$  where  $\tilde{z}_\theta^*(\cdot)$  is solution of the Cauchy problem

$$\dot{z} = \tilde{l}_\theta(t, z) := h_\theta^\dagger(x_\theta^*(t), y_\theta^*(t), z; u_\theta^*(t), v_\theta^*(t), 0), \quad z(0) = y(0)$$

while  $z_\theta^*(\cdot)$  is solution of

$$\dot{z} = l_\theta(t, z) := h_\theta^\dagger(x_\theta^*(t), y_\theta^*(t), z, u_\theta^*(t), v_\theta^*(t), \alpha_\theta^*(t)), \quad z(0) = y(0).$$

One can check that the inequality

$$\tilde{l}_\theta(t, z) \leq l_\theta(t, z), \quad t \in [0, T], \quad z \in \mathbb{R}$$

is fulfilled, which gives by comparison of solutions of scalar ordinary differential equations (see for instance [99]) the inequality

$$\tilde{z}_\theta^*(t) \leq z_\theta^*(t), \quad t \in [0, T].$$

We deduce that  $(x_\theta^*(\cdot), y_\theta^*(\cdot), z_\theta^*(\cdot))$  is necessarily a solution of (6.12).

Let

$$\bar{y} := \inf_{u \in \mathcal{U}} \left\{ \max_{t \in [0, T]} y(t); (x(\cdot), y(\cdot)) \text{ sol. of (6.1)} \right\}$$

By Proposition 6.4.1, we know that there exists an optimal solution  $(x(\cdot), y(\cdot), z(\cdot))$  of problem  $\mathcal{P}_3$  such that  $z(T) = \bar{y}$ . Clearly, this solution belongs to  $\mathcal{S}_\theta$  for any  $\theta$ , and we thus get

$$z_\theta^*(T) \leq \bar{y}. \tag{6.13}$$

Let

$$F_\theta(\Pi) := \bigcup_{(u, v) \in U \times [0, 1]} \begin{bmatrix} f(x, y, u) \\ g(x, y, u) \\ h_\theta(x, y, z, u, v) \end{bmatrix}$$

and note that one has

$$\lim_{\theta \rightarrow +\infty} d(F_\theta(\Pi), F(\Pi)) = 0, \quad \Pi \in \mathcal{D} \times \mathbb{R} \tag{6.14}$$

Consider an increasing sequence of numbers  $\theta_n$  ( $n \in \mathbb{N}$ ), and denote  $\Pi_n(\cdot) = (x_n(\cdot), y_n(\cdot), z_n(\cdot))$  an optimal solution of problem  $\mathcal{P}_3^{\theta_n}$ . Note that one has

$$\mathcal{S}_{\theta_{n+1}} \subset \mathcal{S}_{\theta_n} \cdots \subset \mathcal{S}_{\theta_0}. \quad (6.15)$$

Therefore, the sequence  $\dot{\Pi}_n(\cdot)$  is bounded, and  $\Pi_n(\cdot)$  as well. As  $F$  is upper semi-continuous, we obtain that  $\Pi_n(\cdot)$  converges uniformly on  $[0, T]$ , up to a sub-sequence, to a certain  $\Pi^*(\cdot) = (x^*(\cdot), y^*(\cdot), z^*(\cdot))$  which belongs to  $\mathcal{S}_l$  (see for instance [36, Th. 3.1.7]). From property(6.15), we obtain that  $z_n(T)$  is a non decreasing sequence that converges to  $z^*(T)$ , and from (6.13), we get passing at the limit

$$z^*(T) \leq \bar{y}$$

On another hand,  $(x^*(\cdot), y^*(\cdot), z^*(\cdot))$  belongs to  $\mathcal{S}_l$  and we get from Proposition 6.4.1 the inequality

$$z^*(T) \geq \bar{y}$$

Therefore, one has  $z^*(T) = \bar{y}$  and  $(x^*(\cdot), y^*(\cdot), z^*(\cdot))$  is then an optimal solution of problem  $\mathcal{P}_3$ . From Proposition 6.4.1, we obtain that one has necessarily

$$z^*(T) = \max_{t \in [0, T]} y^*(t).$$

Finally, the sequence  $(\dot{x}_n(\cdot), \dot{y}_n(\cdot))$  being bounded, it converges, up to a sub-sequence, weakly to  $(\dot{x}^*(\cdot), \dot{y}^*(\cdot))$  in  $L^2$  thanks to Alaoglu's Theorem.  $\square$

## 6.5 Numerical illustrations

Our aim is to illustrate the different formulations on problems for which the optimal solution is known.

### 6.5.1 A particular class of dynamics

We consider dynamics of the form

$$(\Sigma) : \begin{cases} \dot{x} = f(x) \\ \dot{y} = g(x, u) \end{cases} \quad x \in \mathbb{R}^n, u \in U$$

**Proposition 6.5.1.** *A feedback control  $x \mapsto \phi^*(x)$  such that*

$$g(x, \phi^*(x)) = \min_{u \in U} g(x, u), \quad x \in \mathbb{R}^n$$

is optimal for problem  $\mathcal{P}$ .

*Proof.* For a given  $x_0$  in  $\mathbb{R}^n$ , let  $x(\cdot)$  be the solution of  $\dot{x} = f(x)$ ,  $x(0) = x_0$  independently to the control  $u(\cdot)$ . Then, for any solution  $y(\cdot)$ , one has

$$y(t) = y(0) + \int_0^t g(x(\tau), u(\tau)) d\tau \geq y(0) + \int_0^t \min_{v \in U} g(x(\tau), v) d\tau, \quad t \geq 0$$

Let  $y^*(\cdot)$  be defined as

$$y^*(t) := y(0) + \int_0^t \min_{v \in U} g(x(\tau), v) d\tau, \quad t \geq 0$$

Clearly, one has

$$\max_t y(t) \geq \max_t y^*(t)$$

where  $y^*(\cdot)$  is a solution of  $\Sigma$  for any measurable control  $u^*(\cdot)$  such that

$$g(x(t), u^*(t)) = \min_{v \in V} g(x(t), v), \quad \text{a.e. } t \geq 0.$$

We conclude that  $y^*(\cdot)$  is an optimal trajectory of problem  $\mathcal{P}$  for the control generated by the feedback  $\phi^*$ .  $\square$

As a toy example, we have considered the system

$$\begin{cases} \dot{x} = 1, & x(0) = 0 \\ \dot{y} = (1-x)(2-x)(4-x)(1+u/2), & y(0) = 0 \end{cases} \quad u \in [-1, 1]$$

for which

$$\phi^*(x) = -\text{sign}\left((1-x)(2-x)(4-x)\right)$$

is an optimal control which minimizes  $\max_{t \in [0, T]} y(t)$ . The optimal control is thus pure bang-bang. Remark that this problem can be equivalently written with a scalar non-autonomous dynamics

$$\dot{y} = (1-t)(2-t)(4-t)(1+u/2)$$

for which the open-loop control

$$u^*(t) = -\text{sign}\left((1-t)(2-t)(4-t)\right)$$

is optimal.

For  $T = 5$ , we have first computed the exact optimal solution of problem  $\mathcal{P}$  with the open-loop  $u^*(\cdot)$ , by integrating the dynamics with Scipy in Python software (see Figure 6.2). Effects of perturbations

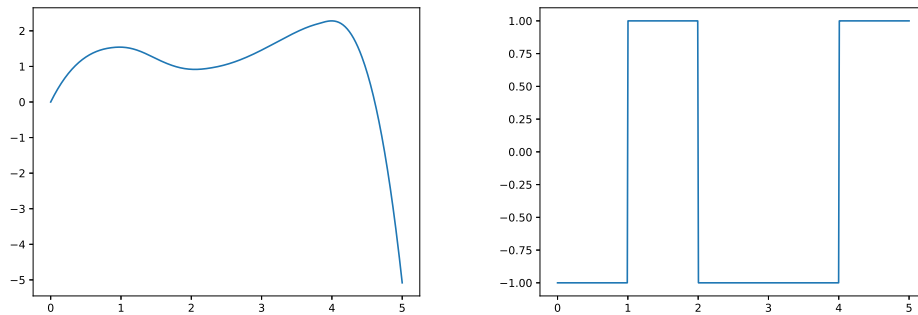


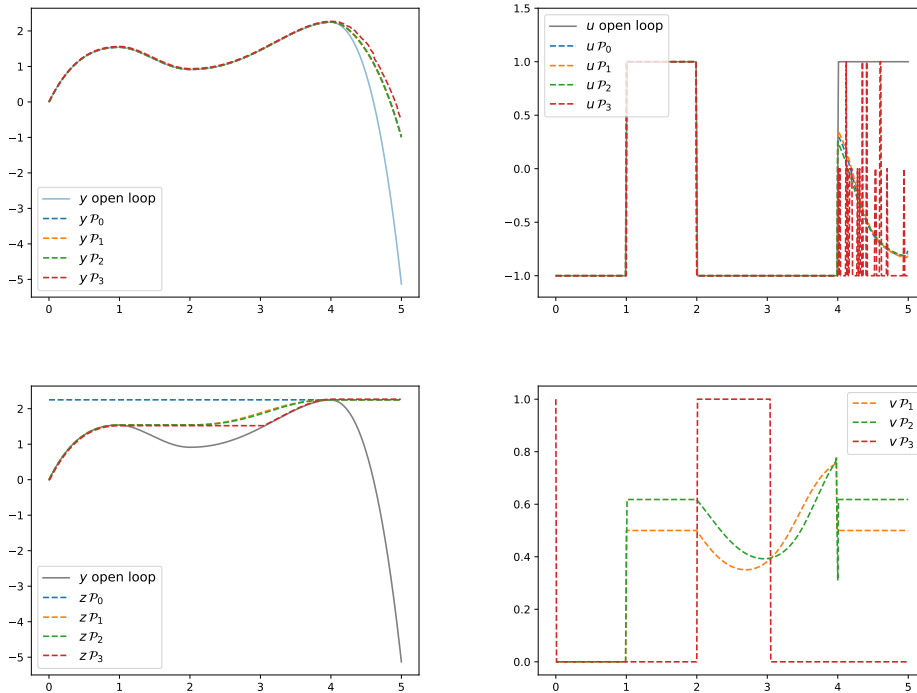
Figure 6.2: Optimal solution:  $y(\cdot)$  on the left,  $u^*(\cdot)$  on the right

on the switching times of the control are presented in Table 6.1, which show a quite high sensitivity of the optimal control for this problem (as it often the case for bang-bang controls). Then, we have solved numerically problems  $\mathcal{P}_0$  to  $\mathcal{P}_2$  with a direct method (Bocop software using Gauss II integration

disturbance	$\max_{t \in [0, T]} y(t)$	error
0	2.24985	0
0.001%	2.24985	$4.10^{-6}\%$
0.01%	2.25010	0.01%
0.1%	2.69457	20%

Table 6.1: Sensitivity to the optimal switching

scheme) for 500 time steps and an optimization relative tolerance equal to  $10^{-10}$ . For problem  $\mathcal{P}_3$ , as the dynamics is not continuous, direct methods do not work well and we have used instead a numerical scheme based on dynamic programming (BocophHJB software) with 500 time steps and a discretization of  $200 \times 200$  points of the state space. For the additional control  $v$ , we have considered only two possible values 0 and 1 as we know that the optimal solution is reached for  $v \in \{0, 1\}$  (see Proposition 6.3.1). The numerical results and computation times are summarized in Table 6.2, while Figure 6.3 presents the corresponding trajectories.

Figure 6.3: Comparisons of the three methods on  $y(\cdot)$ ,  $u(\cdot)$ ,  $z(\cdot)$  and  $v(\cdot)$ .

We note that the direct method give very accurate results, and the computation time for problem  $\mathcal{P}_0$  is the lowest because it has only one control. The computation time for problem  $\mathcal{P}_2$  is slightly higher than for  $\mathcal{P}_1$  because the mixed constraint  $\mathcal{E}_m$  is heavier to evaluate. The numerical method for problem  $\mathcal{P}_3$  is of completely different nature as it computes the optimal solution for all the initial conditions on



problem	$\max_{t \in [0, T]} y(t)$	error	computation time
$\mathcal{P}$	2.24705	0	—
$\mathcal{P}_0$	2.249888	0.126%	0.5 s
$\mathcal{P}_1$	2.24998	0.130%	1.8 s
$\mathcal{P}_2$	2.249941	0.129%	3.8 s
$\mathcal{P}_3$	2.26778	0.8%	248 s

Table 6.2: Comparison of the numerical results

the grid, which explains a much longer computation time. The accuracy of the results is also directly related to the size of the discretization grid and can be improved by increasing this size but at the price of a longer computation time.

On Figure 6.3, one may notice some difference between the obtained trajectories. Let us underline that after the peak of  $y(\cdot)$ , there is no longer uniqueness of the optimal control.

## 6.5.2 Application to an epidemiological model

The SIR model is one of the most basic transmission model in epidemiology for a directly transmitted infectious disease (for a complete introduction, see for instance [100]) and it retakes great importance nowadays due to covid-19 epidemic.

Consider on a time horizon  $[0, T]$  variables  $S(t)$ ,  $I(t)$  and  $R(t)$  representing the fraction of susceptible, infected and recovery individuals at time  $t \in [0, T]$ , so that one has  $S(t) + I(t) + R(t) = 1$  with  $S(t), I(t), R(t) \geq 0$ . Let  $\beta > 0$  be the rate of transmission and  $\gamma > 0$  the recovery rate. Interventions as lock-downs and curfew are modeled as a factor in rate transmission that we denote  $u$  and which represents our control variable taking values in  $[0, u_{max}]$  with  $u_{max} \in (0, 1)$ , where  $u = 0$  means no intervention and  $u = u_{max}$  the most restrictive one which reduces as much as possible contacts among population. The SIR dynamics including the control is then given by the following equations:

$$\dot{S} = -(1-u)\beta SI \quad (6.16)$$

$$\dot{I} = (1-u)\beta SI - \gamma I \quad (6.17)$$

$$\dot{R} = \gamma I \quad (6.18)$$

When the reproduction number  $\mathcal{R}_0 = \beta/\gamma$  is above one and the initial proportion of susceptible is above the herd immunity threshold  $S_h = \mathcal{R}_0^{-1}$ , it is well known that there is an epidemic outbreak. Then, the objective is to minimize the peak of the prevalence

$$\max_{t \in [0, T]} I(t)$$

with respect to control  $u(\cdot)$  subject to a  $L^1$  budget

$$\int_0^T u(t) \leq Q \quad (6.19)$$

on a given time interval  $[0, T]$  where  $T$  is in chosen large enough to ensure the herd immunity of the population is reached at date  $T$ . Note that one can drop the  $R$  dynamics to study this problem. If the constraint (6.19) were not imposed, then the optimal solution would be the trivial control  $u(t) = u_{max}$ ,  $t \in [0, T]$ , which is in general unrealistic from a operational point of view. A similar problem has been

considered in [83] but under the constraint that intervention occurs only once on a time interval of given length, that we relax here. Note that the constraint (6.19) can be reformulated as a target condition, considering the augmented dynamics

$$\dot{S} = -(1-u)\beta SI \quad (6.20)$$

$$\dot{I} = (1-u)\beta SI - \gamma I \quad (6.21)$$

$$\dot{C} = -u(t) \quad (6.22)$$

with initial condition  $C(0) = Q$  and target  $\{C \geq 0\}$ . Extension of the results of Sections 6.3 and 6.4 to problems with target do not present any particular difficulty, and is left to the reader.

For initial conditions  $I_0 = I(0) > 0$  and  $S_0 = S(0) > S_h$ , the optimal solution has been determined in Chapter 7 as the feedback control

$$\psi(I, S) := \begin{cases} 1 - \frac{S_h}{S} & \text{if } I = \bar{I} \text{ and } S > S_h \\ 0 & \text{otherwise} \end{cases}$$

where

$$\bar{I} := \frac{I_0 + S_0 - S_h - S_h \log\left(\frac{S_0}{S_h}\right)}{Q\beta S_h + 1}$$

is the optimal value of the peak. The proof of the optimality of this feedback is out of the scope of the present chapter and can be found in Chapter 7. This control strategy consists in three phases:

1. no intervention until the prevalence  $I$  reaches  $\bar{I}$  (null control),
2. maintain the prevalence  $I$  equal to  $\bar{I}$  until  $S$  reaches  $S_h$  (singular control),
3. no longer intervention when  $S > S_h$  (null control)

as illustrated in Figure 6.4 for the parameters given in Table 6.3. Note that differently to the previous

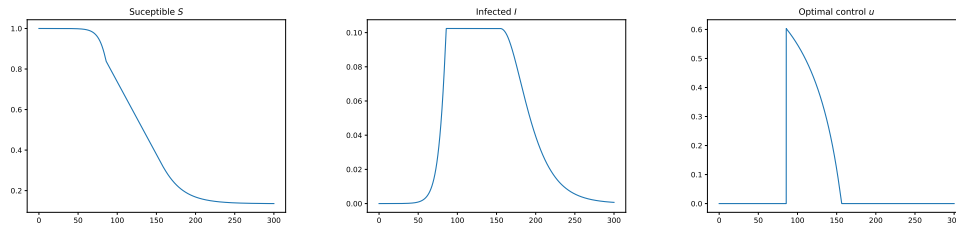


Figure 6.4: The optimal solution for the SIR problem

example, this control strategy is intrinsically robust with respect to a bad choice of  $\bar{I}$ : the maximum value of  $I$  is always guaranteed to be equal to  $\bar{I}$ . However, a mischoice of  $\bar{I}$  has an impact on the budget (see Chapter 7 for more details).

Adding the  $z$ -variable, we end up with a dynamics in dimension four, which is numerically heavier than for the previous example. In particular, methods based on the value function are too time consuming to obtain accurate results for refined grids in a reasonable computation time. So we have considered direct methods only. We do not consider here problem  $\mathcal{P}_3$ , but instead its regular approximations  $\mathcal{P}_3^\theta$  suitable to direct methods. For direct methods that use algebraic differentiation of the dynamics, convergence and

$\beta$	$\gamma$	$T$	$Q$	$S(0)$	$I(0)$	$\bar{I}$
0.21	0.07	300	28	$1 - 10^{-6}$	$10^{-6}$	0.105

Table 6.3: Parameters used in numerical computations and optimal value of the peak

accuracy are much better if one provides differentiable dynamics. This is why we have approximated the  $\max(\cdot, 0)$  operator for problems  $\mathcal{P}_1$  and  $\mathcal{P}_2$  by the Laplace formula

$$\frac{\log(e^{\lambda\xi} + 1)}{\lambda} \xrightarrow{\lambda \rightarrow +\infty} \max(\xi, 0), \quad \xi \in \mathbb{R}$$

with  $\lambda = 100$  for the numerical experiments. For problem  $\mathcal{P}_3^\theta$ , one has to be careful about the interplay between the approximations of  $\max(\cdot, 0)$  and the sequence  $\theta_n \rightarrow +\infty$ , to provide approximations from below of the optimal value. The function  $h_\theta$  is thus approximated by the expression

$$h_\theta(x, y, z, u, v) \simeq \frac{\log(e^{\lambda_1 g(x, y, u)} + 1)}{\lambda_1} \left( 1 - v e^{\frac{\theta}{\lambda_2} \log(e^{\lambda_2(y-z)} + 1)} \right)$$

which depends on three parameters  $\lambda_1$ ,  $\lambda_2$  and  $\theta$ . Posit for convenience

$$\alpha := \frac{\theta}{\lambda_2}$$

and consider the function

$$\omega_{\alpha, \lambda_2}(\xi) := e^{-\alpha \log(e^{-\lambda_2 \xi} + 1)}, \quad \xi \in \mathbb{R}$$

which approximates the indicator function  $\mathbb{1}_{\mathbb{R}^+}$ . One has the following properties.

**Lemma 6.5.1.**

1. For any positive numbers  $\alpha$ ,  $\lambda_2$ , the function  $\omega_{\alpha, \lambda_2}$  is increasing with

$$\lim_{\xi \rightarrow -\infty} \omega_{\alpha, \lambda_2}(\xi) = 0, \quad \lim_{\xi \rightarrow +\infty} \omega_{\alpha, \lambda_2}(\xi) = 1$$

2. For any  $\varepsilon \in (0, 1)$ , one has  $\omega_{\alpha, \lambda_2}(-\varepsilon^2) = \varepsilon$  and  $\omega_{\alpha, \lambda_2}(0) = 1 - \varepsilon$  exactly for

$$\alpha = -\frac{\log(1 - \varepsilon)}{\log(2)}, \quad \lambda_2 = \frac{\log(\varepsilon^{-\frac{1}{\alpha}} - 1)}{\varepsilon^2} \quad (6.23)$$

*Proof.* One has first

$$\omega'_{\alpha, \lambda_2}(\xi) = \lambda_2 \alpha \frac{e^{-\lambda_2 \xi}}{e^{-\lambda_2 \xi} + 1} \omega_{\alpha, \lambda_2}(\xi) > 0$$

and the function  $\omega_{\alpha, \lambda_2}(\cdot)$  is thus increasing. From

$$\lim_{\xi \rightarrow -\infty} -\alpha \log(e^{-\lambda_2 \xi} + 1) = -\infty$$

one get

$$\lim_{\xi \rightarrow -\infty} \omega_{\alpha, \lambda_2}(\xi) = 0$$

and similarly

$$\lim_{\xi \rightarrow +\infty} -\alpha \log(e^{-\lambda_2 \xi} + 1) = 0$$

implies

$$\lim_{\xi \rightarrow +\infty} \omega_{\alpha, \lambda_2}(\xi) = 1$$

Finally, with simple algebraic manipulation of the conditions  $\omega_{\alpha, \lambda_2}(-\varepsilon^2) = \varepsilon$  and  $\omega_{\alpha, \lambda_2}(0) = 1 - \varepsilon$ , one obtains straightforwardly the expressions (6.23).  $\square$

We have taken  $\lambda_1 = 5000$  and considered a sequence of approximations of the indicator function for the values given in Table 6.4 according to expressions (6.23) of Lemma 6.5.1 (see Figure 6.5).

$\varepsilon$	$\alpha$	$\lambda_2$
0.2	0.32	124
0.15	0.234	360
0.1	0.152	1514
0.075	0.112	4094
0.05	0.074	16193

Table 6.4: Values of parameters  $\alpha$ ,  $\lambda_2$  for different  $\varepsilon$

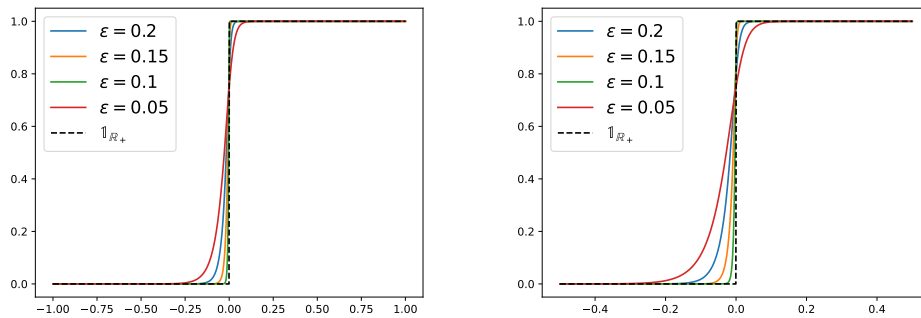
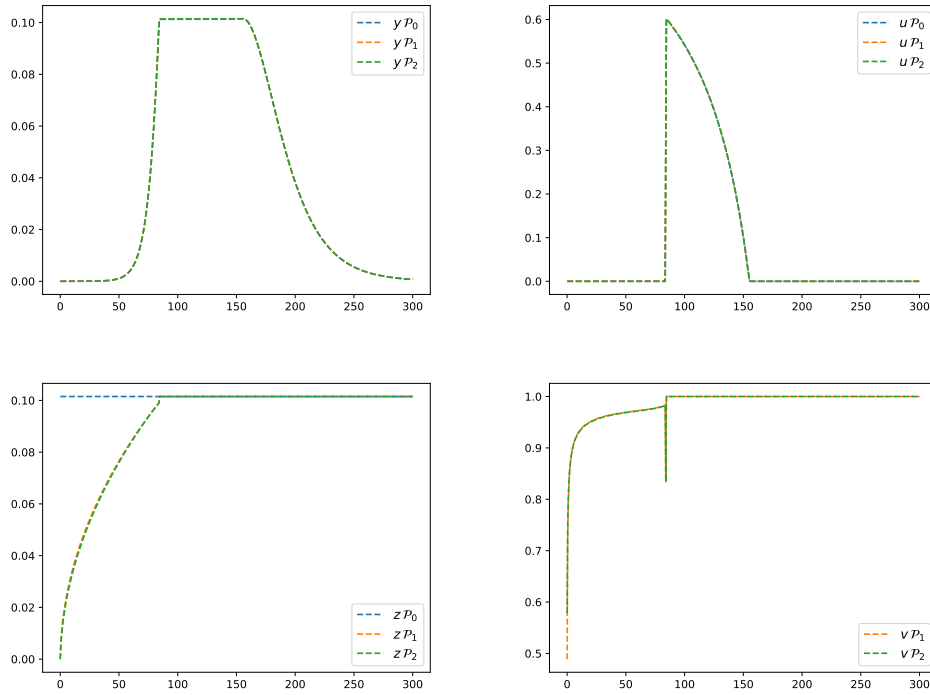


Figure 6.5: Approximation of the indicator function with different values of  $\varepsilon$  (zoom on the abscissa axis on the right)

Computations have been performed with Bocop software on a standard laptop computer (with a Gauss II integration scheme, 600 time steps and relative tolerance  $10^{-10}$ ). As one can see in Figure 6.6 and Table 6.5 problems  $\mathcal{P}_0$ ,  $\mathcal{P}_1$ ,  $\mathcal{P}_2$  give the peak values with a very good accuracy, and present similar performances in terms of computation time. In Figure 6.7 and Table 6.6, the numerical solutions of  $\mathcal{P}_3^\theta$  are illustrated for the values of  $\alpha$  and  $\lambda_2$  given in Table 6.4. As expected, the numerical computation of the family of problems  $\mathcal{P}_3^\theta$  provides an increasing sequence of approximation from below of the optimal value and thus complements the computation of problems  $\mathcal{P}_0$ ,  $\mathcal{P}_1$  or  $\mathcal{P}_2$ . From Figures of Tables 6.5 and 6.6, one can safely guarantee that the optimal value belongs to the interval  $[0.1010, 0.1015]$ . However, the trajectories found for  $\mathcal{P}_3^\theta$  are not as closed as the ones of problems  $\mathcal{P}_0$ ,  $\mathcal{P}_1$  or  $\mathcal{P}_2$ . This can be explained by the fact that problems  $\mathcal{P}_3^\theta$  are not subject to the constraint  $z(t) \geq y(t)$  and thus provides trajectories for which  $z(T)$  is indeed below  $\max_t y(t)$ .

Figure 6.6: Comparisons of numerical results for the methods  $\mathcal{P}_0$ ,  $\mathcal{P}_1$ ,  $\mathcal{P}_2$ 

problem	$\max_{t \in [0, T]} y(t)$	computation time
$\mathcal{P}_0$	0.1015	10s
$\mathcal{P}_1$	0.1015	12s
$\mathcal{P}_2$	0.1015	13s

Table 6.5: Comparison of performances for problems  $\mathcal{P}_0$ ,  $\mathcal{P}_1$ ,  $\mathcal{P}_2$ 

$\varepsilon$	$z(T)$	$\max_{t \in [0, T]} y(t)$	computation time
0.2	0.0684	0.1038	80s
0.15	0.0823	0.1038	65s
0.1	0.0954	0.1037	51s
0.075	0.0993	0.1050	83s
0.05	0.1010	0.1036	97s

Table 6.6: Comparison of performances for problem  $\mathcal{P}_3^\theta$ 

Finally, we have compared our approximation technique with the classical approximation of the  $L^\infty$

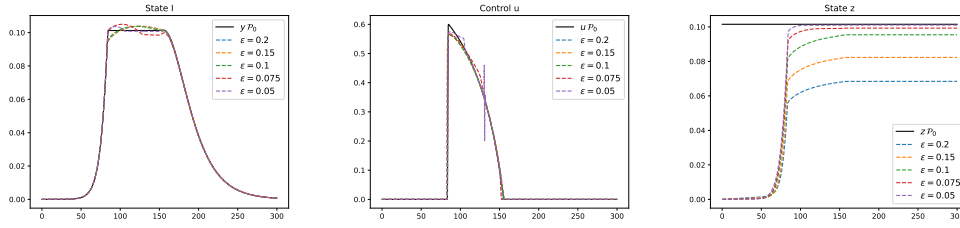


Figure 6.7: Comparison of the numerical results for problem  $\mathcal{P}_3^\theta$

criterion by  $L^p$  norms

$$\mathcal{P}_{L^p} : \inf_{u(\cdot) \in \mathcal{U}} \|y(t)\|_p$$

with the same direct method. To speed up the convergence, we have used the Bocop facility which allows a batch mode which consists in initializing the search from a solution found for a former value of  $p$ , that have been taken  $p \in \{2, 5, 10, 15\}$  (see Figure 6.8). Besides, to ensure convergence it was necessary take 1200 time step instead of 600 as in previous simulations. The total time of the process is 78s after

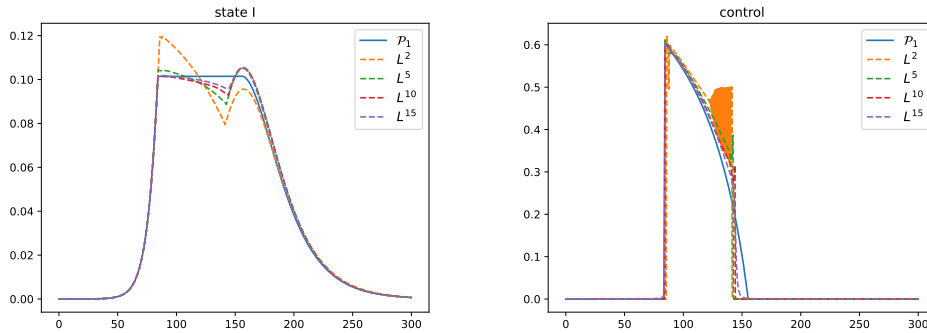


Figure 6.8: Numerical solutions for problems  $\mathcal{P}_{L^p}$

summing computation times given in Table 6.7. However, one can see that the trajectory found for  $p = 15$

$p$	$\max_{t \in [0, T]} y(t)$	$\ y(t)\ _p$	computation time
2	0.119653	1.0222	34 s
5	0.105244	0.2474	14 s
10	0.105375	0.15678	13 s
15	0.105170	0.13549	17 s

Table 6.7: Comparison of the numerical results with the  $L^p$  approximation

is quite far to give a peak value close from the other methods. Moreover, the same method for  $p = 15$  but initialized from the solution found for  $p = 2$  gives poor results for a computation time of 50s (see Figure

6.9). We conclude that the  $L^p$  approximation is not practically reliable for this kind of problems.

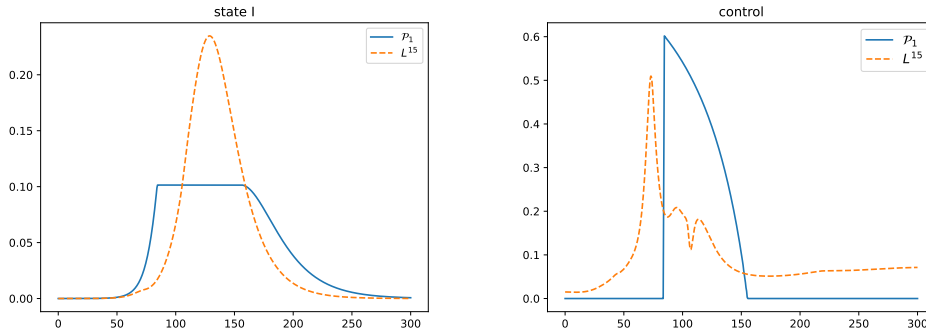


Figure 6.9: Numerical solution for  $\mathcal{P}_{L^{15}}$  without batch iteration (computation time 50s)

## 6.6 Discussion and conclusions

In this work, we have presented different reformulations of optimal control problems with maximum cost in terms of extended Mayer problems, and tested them numerically on two examples whose optimal solution has bang-bang controls and singular arcs. We have proposed two kinds of formulations: one with state or mixed constraints suitable to direct methods, and another one without any constraint but less regular and suitable to dynamical programming type methods. Moreover, for the latter one, we have proposed an approximation scheme generated by a sequence of regular Mayer unconstrained problems, which performs better than approximations based on  $L^p$  norms. However, although this second approach requires larger computation time, it complements the first one providing approximations of the optimal value from above.

This first work puts in perspective the study of necessary optimality conditions for the maximum cost problems with the help of these formulations, which will be the matter of a future work.

Finally, we summarize advantages and drawbacks of the different formulations for numerical computations in Table 6.8, that could help practitioners in the choice of the method.

Formulation	$\mathcal{P}_0$	$\mathcal{P}_1$ or $\mathcal{P}_2$	$\mathcal{P}_3$	$\mathcal{P}_3^\theta$
suitable to direct methods	yes	yes	no	yes
suitable to Hamilton-Jacobi-Bellman methods	no	yes	yes	yes
suitable to shooting methods without constraint	no	no	no	yes
provides approximations from below	no	no	no	yes

Table 6.8: Comparison of the different formulations

## Chapter 7

# A feedback strategy for peak minimization in SIR model

This chapter corresponds to the accepted pre-print [81] titled "An optimal feedback control that minimizes the epidemic peak in the SIR model under a budget constraint" to appear in Automatica journal.

### 7.1 Introduction

Since the pioneering work of Kermack and McKendrick [70], the SIR model has been very popular in epidemiology, as the basic model for infectious diseases with direct transmission (see for instance [100, 75] as introductions on the subject). It retakes great importance nowadays due to the recent coronavirus pandemic. In face of a new pathogen, non-pharmaceutical interventions (such as reducing physical distance in the population) are often the first available means to reduce the propagation of the disease, but this has economic and social prices. In [83, 76], the authors underline the need of control strategies for epidemic mitigation by “flattering the epidemic curve”, rather than eradication of the disease that might be too costly. Several works have applied the optimal control theory considering interventions as a control variable that reduces the effective transmission rate of the SIR model, and studied optimal strategies with criteria based on running and terminal cost over fixed finite interval or infinite horizon [17, 24, 23, 67, 86, 20, 33, 49, 71, 21]. However, the highest peak of the epidemic appears to be the highly relevant criterion to be minimized (especially when there is an hospital pressure to save individuals with severe forms of the infection). In [83], the authors studied the minimization of the peak of the infected population under the constraint that interventions occur on a single time interval of given duration. In the present work, we consider the same criterion, but under a budget constraint on the control (as an integral cost) that we believe to be more relevant as it takes into account the strength of the interventions and does not impose an *a priori* single time interval of given length for the interventions to take place (although we have been able to prove that the optimal solution consists indeed in having interventions on a single time interval but with a control strategy different that the one obtained in [83]). Let us also mention a more recent work [12] that considers a kind of "dual" problem, which consists in minimizing an integral cost of the control under the constraint that the epidemic stays below a prescribed value and an additional constraint on the state at a fixed time. The structure of the optimal strategy given by the authors in [12] is similar to the one we obtained without having to fix a time horizon and a terminal constraint. All the cited works rely on numerical methods to provide the effective control. Here, we give an explicit analytical expression of the optimal control.

Let us stress that optimal control problems with maximum cost are not in the usual Mayer, Lagrange



or Bolza forms of the optimal control theory [34], for which the necessary optimality conditions of Pontryagin's Principle apply, but fall into the class of optimal control with  $L^\infty$  criterion, for which characterizations have been proposed in the literature mainly in terms of the value function (see for instance [15]). Although necessary optimality conditions and numerical procedures have been derived from these characterizations (see for instance [14, 41]), these approaches remain quite difficult and numerically heavy to be applied on concrete problems. On the other hand, for minimal time problems with planar dynamics linear with respect to the control variable, comparison tools based on the application of the Green's Theorem have shown that it is possible to dispense with the use of necessary conditions to prove the optimality of a candidate solution [57]. Although our criterion is of different nature, we show in the present work that it is also possible to implement this approach for our problem.

The chapter is organized as follows. In the next section, we posit the problem of peak minimization to be studied. In Section 7.3, we define a class of feedback strategies that we called "NSN", and give some preliminary properties. Section 7.4 proves the existence of an NSN strategy which is optimal for our problem, and makes it explicit. Finally, Section 7.5 illustrates the optimal solutions on numerical simulations and discusses about the optimal strategy.

## 7.2 Definitions and problem statement

We consider the SIR model

$$\begin{cases} \dot{S} = -\beta SI(1-u) \\ \dot{I} = \beta SI(1-u) - \gamma I \\ \dot{R} = \gamma I \end{cases} \quad (7.1)$$

where  $S$ ,  $I$  and  $R$  denotes respectively the proportion of susceptible, infected and recovered individuals in a population of constant size. The parameters  $\beta$  and  $\gamma$  are the transmission and recovery rates of the disease. The control  $u$ , which belongs to  $U := [0, 1]$ , represents the efforts of interventions by reducing the effective transmission rate. For simplicity, we shall drop in the following the  $R$  dynamics. Throughout the chapter, we shall assume that the *basic reproduction number*  $\mathcal{R}_0$  is larger than one, so that an epidemic outbreak may occur.

**Assumption 3.**

$$\mathcal{R}_0 := \frac{\beta}{\gamma} > 1.$$

For a positive initial condition  $(S(0), I(0)) = (S_0, I_0)$  with  $S_0 + I_0 \leq 1$ , we consider the optimal control problem which consists in minimizing the epidemic peak under a budget constraint

$$\inf_{u(\cdot) \in \mathcal{U}} \max_{t \geq 0} I(t), \quad (7.2)$$

where  $\mathcal{U}$  denotes the set of measurable functions  $u(\cdot)$  that take values in  $U$  and satisfy the  $L^1$  constraint

$$\int_0^{+\infty} u(t) dt \leq Q$$

**Remark 7.2.1.** From equations (7.1), one can easily check that the solution  $I(t)$  tends to zero when  $t$  tends to  $+\infty$  whatever is the control  $u(\cdot)$ , so that the supreme of  $I(\cdot)$  over  $[0, +\infty)$  in (7.2) is reached.

Equivalently, one can consider the extended dynamics.

$$\begin{cases} \dot{S} = -\beta SI(1-u) \\ \dot{I} = \beta SI(1-u) - \gamma I \\ \dot{C} = -u \end{cases} \quad (7.3)$$

with the initial condition  $(S(0), I(0), C(0)) = (S_0, I_0, Q)$  and the state constraint

$$C(t) \geq 0, \quad t \geq 0. \quad (7.4)$$

A solution of (7.3) is *admissible* if the control  $u(\cdot)$  takes its values in  $U$  and the condition (7.4) is fulfilled.

### 7.3 The NSN feedback

Let us denote the *immunity threshold*

$$S_h := \mathcal{R}_0^{-1} = \frac{\gamma}{\beta} < 1.$$

Note that  $S(\cdot)$  is a non increasing function and that one has  $\dot{I} \leq 0$  when  $S \leq S_h$ , whatever is the control. If  $S_0 \leq S_h$ , the maximum of  $I(\cdot)$  is thus equal to  $I_0$  for any control  $u(\cdot)$ , which solves the optimal control problem. We shall now consider that the non-trivial case.

**Assumption 4.**

$$S_0 > S_h.$$

Under this assumption, we thus know that for any admissible solution, the maximum of  $I(\cdot)$  is reached for  $S \geq S_h$ . For the control  $u = 0$ , one can easily check that the following property is fulfilled

$$S(t) + I(t) - S_h \log(S(t)) = S_0 + I_0 - S_h \log(S_0), \quad t > 0, \quad (7.5)$$

and the maximum of  $I(\cdot)$  is then reached for the value

$$I_h := I_0 + S_0 - S_h - S_h \log\left(\frac{S_0}{S_h}\right).$$

We define the "NSN" (for null-singular-null) strategy as follows.

**Definition 7.3.1.** For  $\bar{I} \in [I_0, I_h]$ , consider the feedback control

$$\psi_{\bar{I}}(I, S) := \begin{cases} 1 - \frac{S_h}{S} & \text{if } I = \bar{I} \text{ and } S > S_h \\ 0 & \text{otherwise.} \end{cases} \quad (7.6)$$

We denote the  $L^1$  norm associated to the NSN control

$$\mathcal{L}(\bar{I}) := \int_0^{+\infty} u^{\psi_{\bar{I}}}(t) dt, \quad \bar{I} \in [I_0, I_h],$$

where  $u^{\psi_{\bar{I}}}(\cdot)$  is the control generated by the feedback (7.6).

This control strategy consists in three phases:

1. no intervention until the prevalence  $I$  reaches  $\bar{I}$  (null control),

2. maintain the prevalence  $I$  equal to  $\bar{I}$  until  $S$  reaches  $S_h$  (singular control),
3. no longer intervention when  $S < S_h$  (null control)

**Remark 7.3.1.** *There is no switch of the control between phases 2 and 3, because  $u(t)$  tends to zero when  $S(t)$  tends to  $S_h$ , according to expression (7.6).*

One can check straightforwardly the following properties are fulfilled.

**Lemma 7.3.1.** *For any  $\bar{I} \in [I_0, I_h]$ , the maximal value of the control  $u^{\Psi I}(\cdot)$  is given by*

$$u_{\max}(\bar{I}) := 1 - \frac{S_h}{\bar{S}} < 1,$$

where  $\bar{S}$  is solution of

$$\bar{S} - S_h \log \bar{S} = S_0 + I_0 - S_h \log S_0 - \bar{I}.$$

Moreover, any solution given by the NSN strategy verifies

$$\max_{t \geq 0} I(t) = \bar{I}.$$

## 7.4 Optimal strategy

We first show that the function  $\mathcal{L}$  can be made explicit.

**Proposition 7.4.1.** *One has*

$$\mathcal{L}(\bar{I}) = \frac{I_h - \bar{I}}{\beta S_h \bar{I}}, \quad \bar{I} \in [I_0, I_h]. \quad (7.7)$$

*Proof.* Note first that whatever is  $\bar{I}$ ,  $S(\cdot)$  is decreasing with the control (7.6). One can then equivalently parameterize the solution  $I(\cdot)$ ,  $C(\cdot)$  by

$$\sigma(t) := S_0 - S(t)$$

instead of  $t$ . Posit  $\sigma_h := \sigma(t_h) = S_0 - S_h$ .

As long as  $I < \bar{I}$ , one has  $u = 0$  which gives

$$\begin{cases} \frac{dI}{d\sigma} = f(\sigma) := 1 - \frac{S_h}{S_0 - \sigma} > 0 \\ \frac{dC}{d\sigma} = 0 \end{cases}$$

Remind, from the definition of  $I_h$ , that the solution  $I(\cdot)$  with  $u = 0$  reaches  $I_h$  in finite time. Therefore, one can define the number

$$\bar{\sigma} := \inf\{\sigma \geq 0, I(\sigma) = \bar{I}\} \leq \sigma_h,$$

which verifies

$$\int_0^{\bar{\sigma}} f(\sigma) d\sigma = \bar{I} - I_0. \quad (7.8)$$

For  $\sigma \in [\bar{\sigma}, \sigma_h]$ , one has  $u = 1 - S_h/S$ , that is

$$\begin{cases} \frac{dI}{d\sigma} = 0 \\ \frac{dC}{d\sigma} = -\frac{1}{\beta S_h \bar{I}} \left(1 - \frac{S_h}{S_0 - \sigma}\right) = -\frac{f(\sigma)}{\beta S_h \bar{I}} < 0 \end{cases}$$

One then obtains

$$\mathcal{L}(\bar{I}) = C(0) - C(\sigma_h) = \frac{1}{\beta S_h \bar{I}} \int_{\bar{\sigma}(\bar{I})}^{\sigma_h} f(\sigma) d\sigma$$

and with (7.8) one can write

$$\mathcal{L}(\bar{I}) = \frac{1}{\beta S_h \bar{I}} \left( \int_0^{\sigma_h} f(\sigma) d\sigma + I_0 - \bar{I} \right).$$

On the other hand, one has

$$\int_0^{\sigma_h} f(\sigma) d\sigma = \sigma_h + S_h \log \left( \frac{S_h}{S_0} \right) = I_h - I_0$$

which finally gives the expression (7.7).  $\square$

Then, the best admissible NSN control can be given as follows.

**Corollary 7.4.1.** *When  $Q \leq \frac{I_h - I_0}{\beta S_h I_0}$ , the smallest  $\bar{I} \in [I_0, I_h]$  for which the solution with the NSN strategy is admissible, is given by the value*

$$\bar{I}^*(Q) := \frac{I_h}{Q\beta S_h + 1} \quad (7.9)$$

and one has

$$\mathcal{L}(\bar{I}^*(Q)) = Q. \quad (7.10)$$

We give now our main result that shows that the NSN strategy is optimal.

**Proposition 7.4.2.** *Let Assumptions 3 and 4 be fulfilled. Then, the NSN feedback is optimal with*

$$\bar{I} = \begin{cases} \bar{I}^*(Q), & Q < \frac{I_h - I_0}{\beta S_h I_0} \\ I_0, & Q \geq \frac{I_h - I_0}{\beta S_h I_0} \end{cases}$$

where  $\bar{I}^*(Q)$  is defined in (7.9), and  $\bar{I}$  is the optimal value of problem (7.2).

*Proof.* When  $Q \geq \frac{I_h - I_0}{\beta S_h I_0}$ , the NSN strategy is admissible and the corresponding solution verifies

$$\max_{t \geq 0} I(t) = I_0$$

which is thus optimal.

Consider now  $Q < \frac{I_h - I_0}{\beta S_h I_0}$ . Let  $(S^*(\cdot), I^*(\cdot), C^*(\cdot))$  be the solution generated by the NSN strategy with  $\bar{I} = \bar{I}^*(Q)$ , and denote  $u^*(\cdot)$  the corresponding control. Let

$$\bar{S} := S^*(\bar{t}) \text{ where } \bar{t} = \inf\{t > 0, I^*(t) = \bar{I}\}$$

and

$$t_h^* := \inf\{t > \bar{t}, S^*(t) = S_h\}.$$

We consider in the  $(S, I)$  plane the curve

$$\mathcal{C}^* := \{(S^*(t), I^*(t)); t \in [0, t_h^*]\}$$

For  $S \geq \bar{S}$ , the control (7.6) is null and a upward normal to  $\mathcal{C}^*$  is given by the expression

$$\vec{n}(S, I) = \begin{bmatrix} \beta SI - \gamma I \\ \beta SI \end{bmatrix}, \quad (S, I) \in \mathcal{C}^* \text{ with } S \in [\bar{S}, S_0].$$

On the other hand, the vector field in the  $(S, I)$  plane of any admissible solution is

$$\vec{v}(S, I, u) = \begin{bmatrix} -\beta SI(1-u) \\ \beta SI(1-u) - \gamma I \end{bmatrix}.$$

Then, one has

$$\vec{n}(S, I) \cdot \vec{v}(S, I, u) = -\beta \gamma SI^2 u \leq 0,$$

$\forall (S, I) \in \mathcal{C}^*$  with  $S \in [\bar{S}, S_0]$ , which shows that any admissible solution is below the curve  $\mathcal{C}^*$  in the  $(S, I)$  plane for  $S \in [\bar{S}, S_0]$ . For  $S \in [S_h, \bar{S}]$ , the curve  $\mathcal{C}^*$  is an horizontal line with  $I = \bar{I}$ . Therefore, if there exists an admissible solution  $(S(\cdot), I(\cdot), C(\cdot))$  with  $\max_t I(t) < \bar{I}$ , its trajectory in the  $(S, I)$  plane has to be below the curve  $\mathcal{C}^*$  for any  $S \in [S_h, S_0]$ . Let

$$t_h := \inf\{t > 0, S(t) = S_h\}.$$

One has thus  $I(t_h) < \bar{I}$ . Define

$$T := t_h^* + \frac{1}{\gamma} \log \left( \frac{\bar{I}}{I(t_h)} \right) > t_h^*$$

and consider the non-admissible solution  $(\tilde{S}(\cdot), \tilde{I}(\cdot), \tilde{C}(\cdot))$  of (7.3) on  $[0, T]$  defined by the control

$$\tilde{u}(t) = \begin{cases} u^*(t), & t \in [0, t_h^*) \\ 1, & t \in [t_h^*, T] \end{cases}.$$

One can straightforwardly check with equations (7.3) that the solution  $(\tilde{S}(t), \tilde{I}(t), \tilde{C}(t))$  is

$$\begin{cases} (S^*(t), I^*(t), C^*(t)), & t \in [0, t_h^*) \\ (S_h, \bar{I} \exp(-\gamma(t - t_h^*)), C^*(t_h^*) + t_h^* - t), & t \in [t_h^*, T]. \end{cases}$$

Remind, from Corollary 7.4.1, that one has  $C^*(t_h^*) = 0$  by equation (7.10)). Clearly, one has  $(\tilde{S}(T), \tilde{I}(T)) = (S_h, I(t_h))$  and  $\tilde{C}(T) < 0$ . We consider now in the  $(S, I)$  plane the simple closed curve  $\Gamma$  which is the concatenation of the trajectory  $(\tilde{S}(\cdot), \tilde{I}(\cdot))$  on forward time with the trajectory  $(S(\cdot), I(\cdot))$  in backward time:

$$\Gamma := \{(\tilde{S}(\tau), \tilde{I}(\tau)), \tau \in [0, T]\} \cup \{(S(T + t_h - t), I(T + t_h - t)), \tau \in [T, T + t_h]\}$$

that is anticlockwise oriented by  $\tau \in [0, T + t_h]$ . Then one has

$$\tilde{C}(T) - C(t_h) = \oint_{\Gamma} dC.$$

From equations (7.3), one gets

$$dC = -\frac{dS}{\beta SI} - dt = -\frac{dS}{\beta SI} + \frac{dS + dI}{\gamma I} = \left(1 - \frac{S_h}{S}\right) \frac{dS}{\gamma I} + \frac{dI}{\gamma I}$$

and thus

$$\tilde{C}(T) - C(t_h) = \oint_{\Gamma} P(S, I) dS + Q(S, I) dI$$

with

$$P(S, I) = \left(1 - \frac{S_h}{S}\right) \frac{1}{\gamma I}, \quad Q(S, I) = \frac{1}{\gamma I}.$$

By the Green's Theorem, one obtains

$$\begin{aligned} \tilde{C}(T) - C(t_h) &= \iint_{\mathcal{D}} \left( \frac{\partial Q}{\partial S}(S, I) - \frac{\partial P}{\partial I}(S, I) \right) dS dI \\ &= \iint_{\mathcal{D}} \left(1 - \frac{S_h}{S}\right) \frac{1}{\gamma I^2} dS dI \\ &> 0 \end{aligned}$$

where  $\mathcal{D}$  is the domain bounded by  $\Gamma$  (see Figure 7.1 as an illustration). This implies  $C(t_h) < \tilde{C}(T) < 0$  and thus a contradiction with the admissibility condition (7.4) of the solution  $(S(\cdot), I(\cdot), C(\cdot))$ . We conclude that  $(S^*(\cdot), I^*(\cdot), C^*(\cdot))$  is optimal.  $\square$

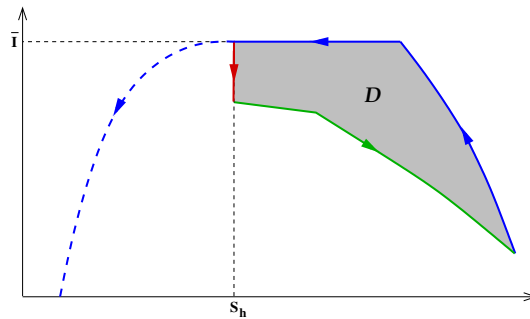


Figure 7.1: The closed curve  $\Gamma$  is composed of the trajectory  $(S^*(\cdot), I^*(\cdot))$  in blue up to to the point  $(S_h, \bar{I})$ , the additional part  $(\tilde{S}(\cdot), \tilde{I}(\cdot))$  in red and the hypothetical better trajectory  $(S(\cdot), I(\cdot))$  in backward time in green.

## 7.5 Numerical illustrations and discussion

We have considered the same parameters and initial condition as in [83] (see Table 7.1). For these values,

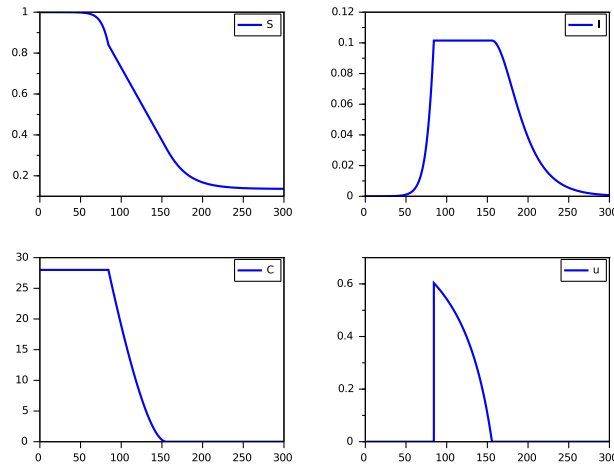
$\beta$	$\gamma$	$S(0)$	$I(0)$
0.21	0.07	$1 - 10^{-6}$	$10^{-6}$

Table 7.1: SIR parameters and initial condition

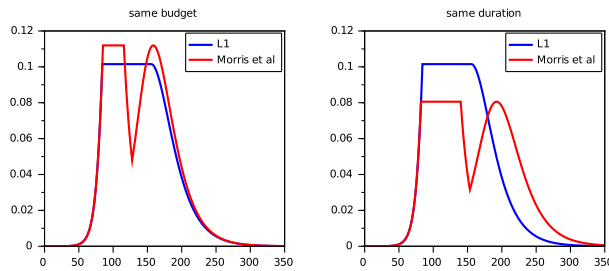
one computes

$$\mathcal{R}_0 = 3, \quad S_h = \frac{1}{3}, \quad I_h \simeq 0.3.$$

Figure 7.2 presents a simulation of the optimal solution for the budget  $Q = 28$ , as an example (the minimum peak is reached for  $\bar{I} \simeq 0.1015$ ). As a comparison, the optimal strategy obtained by Morris et al. in [83] for a fixed time duration of interventions without consideration of any budget is quite different (see Figure 7.3). It consists in four phases: no intervention, maintain  $I$  constant, apply the maximal control

Figure 7.2: Optimal solution for  $Q = 28$ .

(i.e.  $u = 1$ ) and stop the intervention. This control presents thus three switches and relies on a full break of the transmission, differently to the NSN strategy which presents only one switch (see Remark 7.3.1) and does not require a full break (see the maximal value of the control given in Lemma 7.3.1). Applying

Figure 7.3: Comparison of the time evolution of the infected population  $I$  between the optimal NSN strategy and the optimal one of Morris et al.

an NSN strategy appears thus less restrictive to be applied in practice. The strategy proposed by Morris et al. induces also a second peak: after the third phase, the prevalence  $I$  increases again up to a peak which has to be equal to the level maintained during the second phase if it is optimally chosen. But this second peak turns out to be non robust under a mischoice (or mistiming) of the second phase (see [83] for more details). Comparatively, the NSN is naturally robust with respect to a bad choice of  $\bar{I}$ : the maximum value of  $I$  is always guaranteed to be equal to  $\bar{I}$ . However, a mischoice of  $\bar{I}$  has an impact on the budget of the NSN strategy, given by expression (7.7) and illustrated in Table 7.2 (for model parameters given in Table 7.1 and  $Q = 28$ ).

In case of a new epidemic among a large population, one can consider that the initial number of infected individuals is very low, while all the remaining population is susceptible. Therefore, one has  $S_0 + I_0 = 1$  with  $I_0$  very small, and the optimal value of  $\bar{I}$  can be well approximated by its limiting

$\bar{I} - \bar{I}^*$	-10%	-5%	-1%	+5%	+10%
$\mathcal{L}(\bar{I}) - Q$	+17%	+8%	+1.5%	-7%	-14%

Table 7.2: Variation of the control budget of the NSN strategy under a mischoice of  $\bar{I}$ 

expression for  $I_0 = 0$ , that is

$$\bar{I}_\ell := \frac{1 - S_h + S_h \log(S_h)}{Q\beta S_h + 1}. \quad (7.11)$$

From property (7.5), one also gets an approximation of the value  $\bar{S}_\ell$  of  $S$  when  $I$  reaches  $\bar{I}_\ell$  with  $u = 0$ , as the solution of the equation

$$\bar{S}_\ell + \bar{I}_\ell - S_h \log(\bar{S}_\ell) = 1,$$

and then an approximation of the duration of the intervention is given by

$$d_\ell := \frac{S_h - \bar{S}_\ell}{\gamma \bar{I}_\ell},$$

(one can easily check that along the singular arc  $I = \bar{I}$ , one has  $\dot{S} = -\gamma \bar{I}$ ). For the parameters of Table 7.1, one obtains the limiting values given in Table 7.3. This means that depending on the budget  $Q$  only, one can determine the minimal peak and the optimal strategy to apply, without the knowledge of the initial size of the infected population, provided that parameters  $\beta$  and  $\gamma$  of the disease are known.

$\bar{I}_\ell$	$\bar{S}_\ell$	$d_\ell$
0.1015	0.8406	71.39

Table 7.3: The limiting optimal values for arbitrarily small  $I_0$  (with  $Q = 28$ )

The question of parameters estimation in the SIR model from data is out of the scope of the present work. However, while reaching  $I = \bar{I}_\ell$  without intervention, one may expect refinement of the estimates and thus an adjustment of the value of  $\bar{I}_\ell$ .

Note that if it is rather the height of the peak  $\bar{I}$  that is imposed, the corresponding effort can be determined with expression (7.11), that is

$$Q = \frac{1}{\beta S_h} \left( \frac{1 - S_h}{\bar{I}} - 1 \right),$$

as well with the duration of the intervention.

To have a better insight of the impacts of the available budget  $Q$  on the course of the epidemic, we have considered four characteristics numbers:

- $t_i$ : the starting date of the intervention,
- $d$ : the duration of the intervention,
- $\bar{I}$ : the height of the peak,
- $u_{max}$ : the maximal value of the control,



of the optimal solution, depicted on Figure 7.4 as a function of  $Q$  for  $I_0 = 10^{-6}$  and  $S_0 + I_0 = 1$ . Let us note that the maximal budget  $Q$  under which it is not possible to immediately slow down the progress of the epidemic is given, according to Proposition 7.4.2, by

$$Q_{max} := \frac{I_h - I_0}{\beta S_h I_0} \simeq 4.3 \cdot 10^6,$$

which is quite high. Moreover, the maximal value of the control is bounded by the value

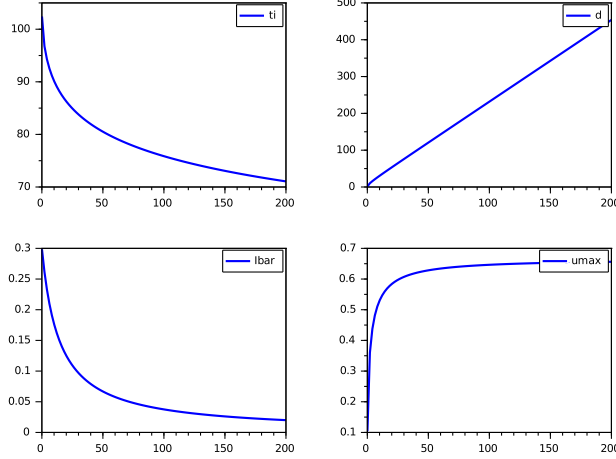


Figure 7.4: Characteristics numbers as functions of  $Q$ .

$$u_{max}(\bar{I}) \leq 1 - S_h = \frac{2}{3},$$

far from the value 1 (that would consist in a total lockdown of the population). On Figure 7.4, one can see that the peak  $\bar{I}$  can be drastically reduced under a reasonable budget, and that taking larger budgets slows down the decrease of the peak, while the duration of the intervention carries on increasing, almost linearly. Indeed, remind that one has  $d = (\bar{S} - S_h)/(\gamma \bar{I})$  and for an optimal value of  $\bar{I}$ , one has  $Q = (I_h - \bar{I})/(\gamma \bar{I})$  from (7.9). Then one gets

$$d = \frac{\bar{S} - S_h}{I_h - \bar{I}} Q,$$

but for large values of  $Q$ ,  $\bar{I}$  is small and  $\bar{S}$  closed to one, which gives an approximation of  $d$  as the linear function of  $Q$

$$d \simeq \frac{1 - S_h}{I_h} Q \simeq 2.194 Q.$$

This implies that for a long duration, fixing the budget  $Q$  or the duration  $d$  tends to be equivalent. Therefore, for a same large duration, the optimal peak gets closed from the optimal one of the strategy of Morris et al. which constraints the duration only, but the difference of the budgets of these two strategies gets increasing with always a lower one for the NSN strategy, as one can see on Figure 7.5.

Finally, this analysis highlights (as already mentioned in [83, 76]) the importance to do not intervene too early (unless one has a very large budget) and to choose the "right" time to launch interventions. We believe that curves as in Figure 7.4 might be of some help for decision makers.

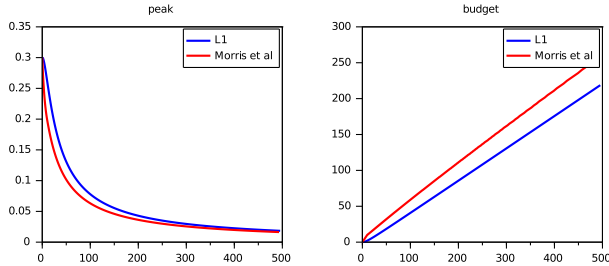


Figure 7.5: Comparison of the performances of the optimal strategies with same duration (in abscissa).



# Chapter 8

## Conclusion and perspectives

The objective of this thesis was to apply optimal control techniques in resource management problems. In particular we worked in two contexts, mining and epidemiology. Now, we will present the main results obtained and open problems which we identified.

We begin with the mining context corresponding to Part 1 of this document.

- **Conclusions:** In Chapter 3, we proposed a formulation based on the continuous framework of [3], where the only change was to reduce the functional space of profiles from continuous to absolutely continuous functions. Then, we characterized analytically optimal profiles of  $(FOP)$  in 2D (Theorem 3.2.1) and 3D (Theorem 3.3.1), both being, main results of that section.

In Chapter 4 we showed  $(FOP)$  and  $(SOP)$  versions in a 2D case assuming continuity in densities  $g$  and  $e$ . Then we recovered Theorem 3.2.1 for these hypotheses. Moreover, we proposed a new version of the sequential problem  $(SOP)$  that we called *Semi continuous formulation* consisting in discretizing the space domain  $\Omega$  and in each point of this discretization take a continuous function characterizing the depth at that point. This formulation is applied on 2D  $((SOP)_{SC}^{2D})$  and 3D cases  $((SOP)_{SC}^{3D})$  and allows us to manipulate the 3D case numerically.

We showed numerical experiments for the 2D case, comparing both formulations and methods of resolution (local optimization and HBJ). Besides, we present, for the first time in literature to our knowledge, numerical solutions of  $(SOP)$  in the 3D case.

- **Perspectives:** The hypothesis in Theorem 3.3.1 are a bit strong, therefore, the search for weaker hypotheses remains an open problem. An option would be work with profiles parameterized in each coordinate by another variable, reducing then the dimension of the optimal control problem space.

In the analytical part, another open problem is to explore optimality conditions for  $((SOP)_{SC}^{2D})$  and  $((SOP)_{SC}^{3D})$  which could give extra information of optimal profiles.

With respect to the numerical part, there does not exist a density function  $g$  in literature built from real data. That information is found in blocks as it was presented in the Introduction. A challenge is to design an efficient interpolation tool to pass from a block model to a continuous function  $g$ , and then use the same optimization tools used in Chapter 4 to solve an academic example.

Now, the epidemiology context corresponding to Part 2 of this document.

- **Conclusions:** In Chapter 6 we worked with general optimal control problems consisting of minimizing the maximum value of a state which appear in epidemiology to minimize the peak of infected individuals after an outbreak disease. We presented four different equivalent formulations

as a Mayer optimal control problem with and without state constraints. These formulations let us apply classical optimal control tools.

We showed the numerical experiment in an academic example and in a more realistic SIR problem. We compared performance among each formulation and  $L^p$  approximation, used to approximate  $L^\infty$  norm. The main advantages and disadvantages in a numerical sense are summarized in the following table.

Formulation	$\mathcal{P}_0$	$\mathcal{P}_1$ or $\mathcal{P}_2$	$\mathcal{P}_3$	$\mathcal{P}_3^\theta$
suitable to direct methods	yes	yes	no	yes
suitable to Hamilton-Jacobi-Bellman methods	no	yes	yes	yes
suitable to shooting methods without constraint	no	no	no	yes
provides approximations from below	no	no	no	yes

Table 8.1: Comparison of the different formulations

In Chapter 7 we worked the particular problem of minimizing the peak of infected individuals over a SIR model with a  $L^1$  cost in the control. We found the analytical expression for the optimal strategy which we called NSN (null-singular-null) and which corresponds to the feedback control (6.5.2).

Finally we presented numerical experiment and a parametric analysis of solutions comparing the budget  $Q$  with the duration of the intervention and height of the optimal peak.

- **Perspectives:** Chapter 6 was focused on modeling and numerical experiments, therefore, there is still work to be done in finding optimality conditions using one of the four formulations.

The strategy NSN presented in Chapter 7 seems possible to extend to more general planar optimal control problems and it is a work in progress. Besides it would be interesting investigate the performance of this strategy with other related problems, for example, minimizing the final value of susceptible  $S(T)$  as in [21].

The compartment model introduced in section 5.2 has not been exploited in an optimal control sense which would be the following step to take. For example, it would be interesting to study what happens with results of Chapters 6 and 7 using this new model rather than the SIR one or indeed consider the vaccination rate as a control to be optimize.

# Appendix A

## Codes global optimization (BocopHJB)

### A.1 1D FOP - continuous formulation

#### A.1.1 Problem definition

```
1 # This file defines all dimensions and parameters for your problem:
2 # Dimensions:
3 state.dimension 2
4 control.dimension 1
5 constant.dimension 2
6 brownian.dimension 0
7
8 # Variable Names :
9 state.0 x
10 state.1 c
11 control.0 u
12
13 # Constants :
14 constant.0 effort_max 1e9
15 constant.1 K 0
16
17 # Time discretization :
18 time.initial 0
19 time.final 1230
20 time.steps 123
21
22 # State discretization: uniform grid
23 state.0.lowerbound 0
24 state.0.upperbound 500
25 state.0.steps 50
26 state.1.lowerbound 0
27 state.1.upperbound 1e6
28 state.1.steps 210
29
30 # value function for points outside the state grid:
31 # final cost ; projection; infinity; user_function
32 valueFunction.out.of.grid infinity
33
34 # Control discretisation :
35 # uniform ;
36 # components_user_file; components_user_function; control_set_user_file;
37 # control_set_user_function; control_set_user_function_state_dependent
```

```

38 control.set uniform
39 # uniform case
40 control.0.lowerbound -1
41 control.0.upperbound 1
42 control.0.steps 100
43
44 # System modes (>=1) and admissible transitions for switching between modes
45 # all_transitions; user_file; user_function
46 # state jumps at transitions: none; user_function
47 system.modes 1
48 admissible.transitions all_transitions
49 state.jumps none
50
51 # Value function save files:
52 # Previous steps type: resume; overwrite; ask
53 # Output format type: text; binary; none
54 # Output path: . --> here; other repertoire
55 valueFunction.previous.steps overwrite
56 valueFunction.output.format text
57 valueFunction.output.path valueFunction/
58
59 # SimulatedTrajectory:
60 # Computation: none; after_valueFunction; read_valueFunction
61 # Noise: none; gaussian; user_function
62 # Starting mode: best_mode; user_function; value:[0, nbmode-1]
63 # Output path: . -->here; other repertoire
64 simulatedTrajectory.computation after_valueFunction
65 simulatedTrajectory.output.path trajectory/
66 simulatedTrajectory.noise gaussian
67 simulatedTrajectory.starting.mode 0
68 simulatedTrajectory.starting.state.0 0.001
69 simulatedTrajectory.starting.state.1 0.1
70 simulatedTrajectory.other_output 0
71
72 # ProcessLaw:
73 # Computation: true; false
74 # Output Path: . -->here; other repertoire
75 processLaw.computation false
76 processLaw.output.path processLaw/
77 processLaw.initial.path processLaw/initialDistribution/
78
79 HD.option false

```

## A.1.2 Objective

```

1 // This code is published under the Eclipse Public License
2 // Authors: Daphne Giorgi, Benjamin Heymann, Jinyan Liu, Pierre Martinon, Olivier
   Tissot
3 // Inria Saclay and Cmap Ecole Polytechnique
4 // 2014-2017
5
6
7 // Function for the running cost
8 // Input :
9 // time : current time t
10 // initial_time : t0
11 // final_time : tf
12 // state : vector of state variables x
13 // control : vector of control variables u
14 // mode : mode of the system i
15 // constants : vector of constants

```

```

16 // dim_constant : dimension of the vector constants
17 // Output :
18 // running_cost : running cost l(t,x,u,i)
19 #include "dependencies.hpp"
20 #include "publicTools.hpp"
21
22 #include "header_runningCost"
23 {
24 double depth = state[0];
25 double distance = time;
26 double p0 = interpolation(distance, distance_grid, p0_values);
27 running_cost = - G_depth_integral(p0,depth,distance);
28 }
29
30
31 // Function for the final cost
32 // Input :
33 // initial_time : t0
34 // final_time : tf
35 // final_state : vector of state variables x_f
36 // final_mode : final mode of the system i_f
37 // constants : vector of constants
38 // dim_constant : dimension of the vector constants
39 // Output :
40 // final_cost : final cost g(t0,tf,x_f,i_f)
41 #include "header_finalCost"
42 {
43
44 // final condition xf = 0
45 double xf = final_state[0];
46 if (abs(xf) <= 1e0)
47     final_cost = 0e0;
48 else
49     final_cost = 1e20;
50 }
51
52
53 // Function for the switching cost
54 // Input :
55 // current_mode : current mode
56 // next_mode : next mode
57 // constants : vector of constants
58 // dim_constant : dimension of the vector constants
59 // Output :
60 // switching_cost : switching cost s(i_k,i_k+1)
61 #include "header_switchingCost"
62 {
63     switching_cost = 0e0;
64 }

```

### A.1.3 Dynamics

```

1 // This code is published under the Eclipse Public License
2 // Authors: Daphne Giorgi, Benjamin Heymann, Jinyan Liu, Pierre Martinon, Olivier
   Tissot
3 // Inria Saclay and Cmap Ecole Polytechnique
4 // 2014-2017
5
6 // General dynamics
7 // dy/dt = drift(t,y,u)dt + volatility(y,u)dWt where Wt is the standard Brownian
   motion

```



```

8
9 // Function for the drift (deterministic dynamics)
10 // Input :
11 // time : current time t
12 // initial_time : t0
13 // final_time : tf
14 // state : vector of state variables x
15 // control : vector of control variables u
16 // mode : mode of the system i
17 // constants : vector of constants
18 // dim_constant : dimension of the vector constants
19 // Output :
20 // state_dynamics : drift f(t,x,u,i) ie deterministic dynamics
21 #include "publicTools.hpp"
22 #include "header_drift"
23 {
24     double distance = time;
25     double depth = state[0];
26     double u = control[0];
27
28     double p0 = interpolation(distance, distance_grid, p0_values);
29     double K = constants[1];
30     if (K == 0)
31         K = interpolation2D(distance, depth, distance_grid, depth_grid, K_values);
32     state_dynamics[0] = u * K;
33     state_dynamics[1] = depth - p0;
34 }
35
36
37 // Function for the volatility (stochastic dynamics)
38 // Input :
39 // time : current time t
40 // initial_time : t0
41 // final_time : tf
42 // state : vector of state variables x
43 // control : vector of control variables u
44 // mode : mode of the system i
45 // constants : vector of constants
46 // dim_constant : dimension of the vector constants
47 // Output :
48 // volatility_dynamics : vector giving the volatility expression of the volatility
49 // Remember that this is a matrix of dimension dim_state x dim_brownian and you
50 // have to fill every coefficient.
51 #include "header_volatility"
52 {
53
54 void generic_noise_dynamics (const double time, const vector<double>& state, const
55     vector<double>& control,
56     const int mode, const vector<double> noise, const
57     vector<double>& constants, vector<double>& dynamics)
58 {
59 }

```

#### A.1.4 Constraints

```

1 // This code is published under the Eclipse Public License
2 // Authors: Daphne Giorgi, Benjamin Heymann, Jinyan Liu, Pierre Martinon, Olivier
3 // Tissot
4 // Inria Saclay and Cmap Ecole Polytechnique
5 // 2014-2017

```

```
5
6 #include "publicTools.hpp"
7
8 // Function for the state admissibility
9 // Input :
10 // time : current time (t)
11 // state : vector of state variables (x)
12 // mode : current mode of the system (i)
13 // constants : vector of constants
14 // dim_constant : dimension of the vector constants
15 // Output :
16 // true if the state is admissible
17 // false if it is not
18 #include "header_checkAdmissibleState"
19 {
20     double depth = state[0];
21     double distance = time;
22     double p0 = interpolation(distance, distance_grid, p0_values);
23     if (depth < p0)
24         return false;
25     else
26         return true;
27 }
28
29
30 // Function for the (control, state) admissibility
31 // Input :
32 // time : current time (t)
33 // state : vector of state variables (x)
34 // control: vector of control variables (u)
35 // mode : current mode of the system (i)
36 // constants : vector of constants
37 // dim_constant : dimension of the vector constants
38 // Output :
39 // true if the (control, state) pair is admissible
40 // false if it is not
41 #include "header_checkAdmissibleControlState"
42 {
43     return true;
44 }
45
46
47 // Function for the final state admissibility
48 // Input :
49 // time : current time (t)
50 // final_state : vector of state variables (x)
51 // control: vector of control variables (u)
52 // mode : current mode of the system (i)
53 // constants : vector of constants
54 // dim_constant : dimension of the vector constants
55 // Output :
56 // true if the (control, state) pair is admissible
57 // false if it is not
58 #include "header_checkAdmissibleFinalState"
59 {
60     double final_effort = final_state[1];
61     double effort_max = constants[0];
62     if (final_effort > effort_max)
63         return false;
64     else
65         return true;
66 }
```

### A.1.5 Other functions

```

1 #include "publicTools.hpp"
2 #include "dependencies.hpp"
3 using namespace std;
4
5
6 #include "header_preProcessing"
7 {
8
9 // initialize interpolations for p0, G and K
10 readFileToVector("data/distance.data", distance_grid);
11 readFileToVector("data/p0.data", p0_values);
12 readFileToVector("data/depth.data", depth_grid);
13 readCVSToMatrix("data/gain.csv", G_values, ',', 0, 1);
14 readCVSToMatrix("data/slope.csv", K_values, ',', 0, 1);
15
16 return 0;
17 }

1 // This code is published under the Eclipse Public License
2 // Authors: Daphne Giorgi, Benjamin Heymann, Jinyan Liu, Pierre Martinon, Olivier
   Tisot
3 // Inria Saclay and Cmap Ecole Polytechnique
4 // 2014-2017
5
6
7 // Function to define the value function outside of the state grid
8 // Input :
9 // time : current time (t)
10 // state : vector of state variables
11 // constants : vector of constants
12 // dim_constant : dimension of the vector constants
13 // Output :
14 // result : double representing the user interpolation formula
15 #include "header_userOutOfGridValueFunction"
16 {
17     result = 1e20;
18 }
19
20
21 /** User function for component-wise control discretization */
22 #include "header_userControlDiscretization"
23 {
24     vector <vector<double> > control_discretization;
25     return control_discretization;
26 }
27
28
29 /** User function for the set of discretized controls*/
30 #include "header_userControlSet"
31 {
32     vector< vector<double> > control_set;
33     return control_set;
34 }
35
36
37 /** User function for the set of discretized controls (state dependent) */
38 #include "header_userControlSetStateDependent"
39 {
40     vector< vector<double> > control_set;

```

```

41     return control_set;
42 }
43
44
45 /** User function for the set of admissible transitions between system modes */
46 #include "header_userAdmissibleTransition"
47 {
48     vector<vector<int> > admissibleTransitionSet;
49     return admissibleTransitionSet;
50 }
51
52
53 /** User function for state jump */
54 #include "header_stateJumpAtSwitching"
55 {
56     // vector 'state' can be modified here
57     // example: if state[0] counts the number of switchings
58     // if (current_mode != next_mode)
59     //     state[0] = state[0] + 1;
60 }
61
62
63 /** User noise */
64 #include "header_userNoise"
65 {
66     vector<double> noise;
67     return noise;
68 }
69
70
71 void other_outputs(double time, vector<double> state, vector<double> control,
72                  vector<double> noise, vector<double> constants, vector<double>& output)
73 {
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```
27 double G0 = interpolation2D(distance, d0, distance_grid, depth_grid, G_values);
28 double G1 = interpolation2D(distance, d1, distance_grid, depth_grid, G_values);
29 g_sum += 0.5e0 * (G0 + G1) * (d1 - d0);
30
31 // complete intervals (depth_grid[j],depth_grid[j+1])
32 for (int j=j0+1;j<j1;j++)
33 {
34     d0 = depth_grid[j];
35     d1 = depth_grid[j+1];
36     G0 = interpolation2D(distance, d0, distance_grid, depth_grid, G_values);
37     G1 = interpolation2D(distance, d1, distance_grid, depth_grid, G_values);
38     g_sum += 0.5e0 * (G0 + G1) * (d1 - d0);
39 }
40
41 // partial last interval (depth_grid[j1],detph)
42 d0 = depth_grid[j1];
43 d1 = depth;
44 G0 = interpolation2D(distance, d0, distance_grid, depth_grid, G_values);
45 G1 = interpolation2D(distance, d1, distance_grid, depth_grid, G_values);
46 g_sum += 0.5e0 * (G0 + G1) * (d1 - d0);
47
48 // rescale
49 return g_sum / 1e6;
50 }
```

# Appendix B

## Codes local optimization (Bocop)

### B.1 1D SOP - continuous formulation

#### B.1.1 Problem definition

```
1 # Definition file
2
3 # Dimensions
4 dim.state 3
5 dim.control 1
6 dim.boundaryconditions 6
7 dim.pathconstraints 1
8 dim.parameters 0
9 dim.constants 2
10
11 # Time interval
12 initial.time 0
13 final.time 1230
14
15 # Constants
16 constant.0 1 # slope type (value or '0' for table)
17 constant.1 0 # quadratic regularization. effect not clear...
18
19 # Time discretisation NB; singular case, CV not necessarily better when increasing
    steps from 1000 to 2000
20 time.steps 1000
21
22 # Bounds for constraints
23 boundarycond.0.lowerbound 0
24 boundarycond.0.upperbound 0
25 boundarycond.1.lowerbound 0
26 boundarycond.1.upperbound 0
27 boundarycond.2.lowerbound 0
28 boundarycond.2.upperbound 0
29 boundarycond.3.lowerbound 0
30 boundarycond.3.upperbound 0
31 boundarycond.4.lowerbound 0
32 boundarycond.5.upperbound 2e4
33 pathconstraint.0.lowerbound 0
34
35 # Bounds for variables
36 state.0.upperbound 500
```

```

37 control.0.lowerbound -1
38 control.0.upperbound 1
39
40 # Initialization for discretized problem
41 state.0.init 0.1
42 state.1.init 0.1
43 state.2.init 0.1
44 control.0.init 0.1
45
46 # Names
47 state.0.name depth
48 state.1.name gain
49 state.2.name effort
50
51 # Ipopt
52 ipoptIntOption.print_level 5
53 ipoptIntOption.max_iter 2000
54 ipoptStrOption.mu_strategy monotone
55 ipoptNumOption.tol 1e-12
56
57 # Misc
58 ad.retape 1

```

## B.1.2 Problem functions

```

1 // +++DRAFT+++ This class implements the OCP functions
2 // It derives from the generic class bocop30CPBase
3 // OCP functions are defined with templates since they will be called
4 // from both the NLP solver (double arguments) and AD tool (ad_double arguments)
5 #include <OCP.h>
6
7 // data for interpolations
8 std::vector <double> depth_grid, p0_values, distance_grid;
9 std::vector <std::vector <double> > G_values;
10 std::vector <std::vector <double> > K_values;
11
12 template<typename Variable> Variable getG(const double distance, const Variable
13     depth)
14 {
15     // basic 2D linear interpolation
16     int verbose = 0;
17     Variable distance_ad = distance;
18     Variable G = bcp::interpolation2Dbilinear(distance_ad, depth, distance_grid,
19         depth_grid, G_values, verbose);
20
21     // rescale
22     return G / 1e6;
23 }
24
25 template<typename Variable> Variable getK(const double distance, const Variable
26     depth)
27 {
28     // basic 2D linear interpolation
29     int verbose = 0;
30     Variable distance_ad = distance;
31     Variable K = bcp::interpolation2Dbilinear(distance_ad, depth, distance_grid,
32         depth_grid, K_values, verbose);
33     return K ;
34 }

```

```

33 template<typename Variable> Variable G_depth_integral(const Variable p0, const
    Variable depth, const double distance)
34 {
35     // compute integral of g over [p0,depth] at x=distance +++ use rectangle instead
    // ? ('right'/implicit rectangle ?)
36     Variable g_sum = 0e0;
37
38     // limit indices for integral along depth
39     int verbose = 0;
40     int j0 = bcp::locate(p0,depth_grid,verbose);
41     int j1 = bcp::locate(depth,depth_grid,verbose);
42     double d0, d1;
43     Variable G0, G1;
44
45     // partial first interval (p0,depth_grid[j0+1])
46     d1 = depth_grid[j0+1];
47     G0 = getG(distance, p0);
48     G1 = getG(distance, d1);
49     g_sum = g_sum + (G0 + G1) / 2e0 * (d1 - p0);
50
51     // complete intervals (depth_grid[j],depth_grid[j+1])
52     for (int j=j0+1;j<j1;j++)
53     {
54         d0 = depth_grid[j];
55         d1 = depth_grid[j+1];
56         G0 = getG(distance, d0);
57         G1 = getG(distance, d1);
58         g_sum = g_sum + (G0 + G1) / 2e0 * (d1 - d0);
59     }
60
61     // partial last interval (depth_grid[j1],depth)
62     d0 = depth_grid[j1];
63     G0 = getG(distance, d0);
64     G1 = getG(distance, depth);
65     g_sum = g_sum + (G0 + G1) / 2e0 * (depth - d0);
66
67     return g_sum;
68 }
69
70
71 template<typename Variable> Variable E_depth_integral(const Variable depth0, const
    Variable depth1)
72 {
73     // NB. the first branch (then) blocks convergence --- o0
74     //if (depth1 <= depth0)
75     // return 0e0;
76     //else
77     return depth1 - depth0;
78 }
79
80 // //////////////////////////////////////
81
82 template <typename Variable>
83 inline void OCP::finalCost(double initial_time, double final_time, const Variable
    *initial_state, const Variable *final_state, const Variable *parameters, const
    double *constants, Variable &final_cost)
84 {
85     // maximize integral of G (sum of all 2nd final state of each timeframe)
86     final_cost = - final_state[1] - final_state[4];
87 }
88
89 template <typename Variable>

```



```

90 inline void OCP::dynamics(double time, const Variable *state, const Variable *
    control, const Variable *parameters, const double *constants, Variable *
    state_dynamics)
91 {
92     // LAYOUT: 3 states and 1 control
93     // - depth profile for current timeframe
94     // - gain from previous timeframe
95     // - effort from previous timeframe
96     // - control is slope derivative for the depth profile
97
98     double distance = time;
99     Variable depth = state[0];
100     Variable u = control[0];
101
102     int verbose = 0;
103     Variable p0 = bcp::interpolation1Dlinear(distance, distance_grid, p0_values,
        verbose);
104
105     Variable K = constants[0];
106     if (K == 0)
107         K = getK(distance, depth);
108     double eps_reg = constants[1];
109
110     state_dynamics[0] = u * K;
111     state_dynamics[1] = G_depth_integral(p0, depth, distance) - eps_reg*u*u;
112     state_dynamics[2] = E_depth_integral(p0, depth);
113
114 }
115
116 template <typename Variable>
117 inline void OCP::boundaryConditions(double initial_time, double final_time, const
    Variable *initial_state, const Variable *final_state, const Variable *
    parameters, const double *constants, Variable *boundary_conditions)
118 {
119     // initial and final conditions for depth, gain, effort
120     size_t dim_state = 3;
121     for (int i=0; i<dim_state; i++)
122     {
123         boundary_conditions[i] = initial_state[i];
124         boundary_conditions[dim_state+i] = final_state[i];
125     }
126 }
127
128 template <typename Variable>
129 inline void OCP::pathConstraints(double time, const Variable *state, const
    Variable *control, const Variable *parameters, const double *constants,
    Variable *path_constraints)
130 {
131     // initial profile constraint  $x \geq p_0$ 
132     int verbose = 0;
133     double distance = time;
134     double p0 = bcp::interpolation1Dlinear(distance, distance_grid, p0_values,
        verbose);
135     path_constraints[0] = state[0] - p0;
136 }
137
138
139 void OCP::preProcessing()
140 {
141     // initialize interpolations for p0 and G
142     bcp::readFileToVector("data/distance.data", distance_grid);
143     bcp::readFileToVector("data/p0.data", p0_values);

```

```

144 bcp::readFileToVector("data/depth.data", depth_grid);
145 bcp::readCSVToMatrix("data/gain.csv",G_values,',',0,1);
146 bcp::readCSVToMatrix("data/slope.csv",K_values,',',0,1);
147 }
148
149
150 // //////////////////////////////////////
151 // explicit template instantiation for template functions, with double and
    double_ad
152 // +++ could be in an included separate file ?
153 // but needs to be done for aux functions too ? APPARENTLY NOT !
154 template void OCP::finalCost<double>(double initial_time, double final_time, const
    double *initial_state, const double *final_state, const double *parameters,
    const double *constants, double &final_cost);
155 template void OCP::dynamics<double>(double time, const double *state, const double
    *control, const double *parameters, const double *constants, double *
    state_dynamics);
156 template void OCP::boundaryConditions<double>(double initial_time, double
    final_time, const double *initial_state, const double *final_state, const
    double *parameters, const double *constants, double *boundary_conditions);
157 template void OCP::pathConstraints<double>(double time, const double *state, const
    double *control, const double *parameters, const double *constants, double *
    path_constraints);
158
159 template void OCP::finalCost<double_ad>(double initial_time, double final_time,
    const double_ad *initial_state, const double_ad *final_state, const double_ad
    *parameters, const double *constants, double_ad &final_cost);
160 template void OCP::dynamics<double_ad>(double time, const double_ad *state, const
    double_ad *control, const double_ad *parameters, const double *constants,
    double_ad *state_dynamics);
161 template void OCP::boundaryConditions<double_ad>(double initial_time, double
    final_time, const double_ad *initial_state, const double_ad *final_state,
    const double_ad *parameters, const double *constants, double_ad *
    boundary_conditions);
162 template void OCP::pathConstraints<double_ad>(double time, const double_ad *state,
    const double_ad *control, const double_ad *parameters, const double *
    constants, double_ad *path_constraints);

```

## B.2 1D SOP - semicontinuous formulation

### B.2.1 Problem definition

```

1 # Definition file
2
3 # Dimensions
4 dim.state 100
5 dim.control 99
6 dim.boundaryconditions 101
7 dim.pathconstraints 101
8 dim.parameters 0
9 dim.constants 3
10
11 # Time interval
12 initial.time 0
13 final.time 1
14
15 # Constants
16 constant.0 100
17 constant.1 20000.0
18 constant.2 0.1

```

```

19
20 # Time discretisation
21 ode.discretization euler_implicit
22 time.steps 1
23
24 # Bounds for constraints
25 boundarycond.0.lowerbound 0
26 boundarycond.0.upperbound 0
27 ...
28 boundarycond.99.lowerbound 0
29 boundarycond.99.upperbound 0
30 boundarycond.100.lowerbound 5000.0
31 pathconstraint.0.lowerbound -1
32 pathconstraint.0.upperbound 1
33 ...
34 pathconstraint.99.lowerbound -1
35 pathconstraint.99.upperbound 1
36 pathconstraint.100.lowerbound 0
37
38 # Bounds for variables
39 state.0.lowerbound 0
40 state.0.upperbound 500
41 ...
42 state.98.upperbound 500
43 state.99.lowerbound 0
44 control.0.lowerbound 0
45 control.0.upperbound 500
46 ...
47 control.99.lowerbound 0
48 control.99.upperbound 500
49
50 # Initialization for discretized problem
51 state.0.init 100
52 ...
53 state.99.init 100
54 control.0.init 0.1
55 ...
56 control.98.init 0.1
57
58 # Names
59
60 # Ipopt
61 ipoptIntOption.print_level 5
62 ipoptIntOption.max_iter 10000
63 ipoptStrOption.mu_strategy monotone
64 ipoptNumOption.tol 1e-6
65
66 # Misc
67 ad.retape 1

```

## B.2.2 Problem functions

```

1 // +++DRAFT+++ This class implements the OCP functions
2 // It derives from the generic class bocop30CPBase
3 // OCP functions are defined with templates since they will be called
4 // from both the NLP solver (double arguments) and AD tool (ad_double arguments)
5 #include <OCP.h>
6
7 // gloabl N, DX, T ?
8

```

```

9 // data for interpolations
10 std::vector <double> depth_grid, p0_values, distance_grid;
11 std::vector <std::vector <double> > G_values;
12
13 template<typename Variable> Variable getG(const double distance, const Variable
    depth)
14 {
15     // basic 2D linear interpolation
16     int verbose = 0;
17     Variable distance_ad = distance;
18     Variable G = bcp::interpolation2Dbilinear(distance_ad, depth, distance_grid,
        depth_grid, G_values, verbose);
19
20     // rescale
21     return G / 1e6;
22 }
23
24 template<typename Variable> Variable G_depth_integral(const Variable p0, const
    Variable depth, const double distance)
25 {
26     // compute integral of g over [p0,depth] at x=distance +++ use rectangle instead
        ? ('right'/implicit rectangle ?)
27     Variable g_sum = 0e0;
28
29     // limit indices for integral along depth
30     int verbose = 0;
31     int j0 = bcp::locate(p0, depth_grid, verbose);
32     int j1 = bcp::locate(depth, depth_grid, verbose);
33     double d0, d1;
34     Variable G0, G1;
35
36     // partial first interval (p0,depth_grid[j0+1])
37     d1 = depth_grid[j0+1];
38     G0 = getG(distance, p0);
39     G1 = getG(distance, d1);
40     g_sum = g_sum + (G0 + G1) / 2e0 * (d1 - p0);
41
42     // complete intervals (depth_grid[j],depth_grid[j+1])
43     for (int j=j0+1;j<j1;j++)
44     {
45         d0 = depth_grid[j];
46         d1 = depth_grid[j+1];
47         G0 = getG(distance, d0);
48         G1 = getG(distance, d1);
49         g_sum = g_sum + (G0 + G1) / 2e0 * (d1 - d0);
50     }
51
52     // partial last interval (depth_grid[j1],depth)
53     d0 = depth_grid[j1];
54     G0 = getG(distance, d0);
55     G1 = getG(distance, depth);
56     g_sum = g_sum + (G0 + G1) / 2e0 * (depth - d0);
57
58     return g_sum;
59 }
60
61
62 template<typename Variable> Variable gain_at_timeframe(const Variable *state,
    const Variable *control, const int N)
63 {
64     // NB. if using implicit euler then state is at the end of the time step aka P
        ~k+1

```

```

65 // previous state  $P^k$  is recomputed via the control
66
67 // gain for current timeframe
68 //  $G = \sum DX (g_i^k + g_{i+1}^k) / 2$ 
69 // with  $g_i^k = \int_{P_i^k}^{P_{i+1}^k} G(X_i, p) dp$ 
70
71 double DX = 1230 / N;
72 Variable gain = 0e0;
73 Variable Gi, Giplus, depth_start, depth_end;
74 double distance;
75
76 for (int i=-1; i<=N-2; i++)
77 {
78 //  $G_i = \int_{P_i^k}^{P_{i+1}^k} G(X_i, p) dp$ 
79 if (i==-1)
80     Gi = 0e0;
81 else
82     {
83         depth_end = state[i];
84         depth_start = depth_end - control[i];
85         distance = DX * (i+1);
86         Gi = G_depth_integral(depth_start, depth_end, distance);
87     }
88 //  $G_{i+1} = \int_{P_{i+1}^k}^{P_{i+2}^k} G(X_{i+1}, p) dp$ 
89 if (i==N-2)
90     Giplus = 0e0;
91 else
92     {
93         depth_end = state[i+1];
94         depth_start = depth_end - control[i+1];
95         distance = DX * (i+2);
96         Giplus = G_depth_integral(depth_start, depth_end, distance);
97     }
98     gain += DX / 2 * (Gi + Giplus);
99 }
100 return gain;
101 }
102
103
104 // //////////////////////////////////////
105
106 template <typename Variable>
107 inline void OCP::finalCost(double initial_time, double final_time, const Variable
108 *initial_state, const Variable *final_state, const Variable *parameters, const
109 double *constants, Variable &final_cost)
110 {
111 // maximize final gain
112 final_cost = - final_state[stateSize()-1];
113 }
114
115 template <typename Variable>
116 inline void OCP::dynamics(double time, const Variable *state, const Variable *
117 control, const Variable *parameters, const double *constants, Variable *
118 state_dynamics)
119 {
120 int N = (int) constants[0];
121
122 // digging progress:  $p_{i+1}^k - p_i^k = u_i^k$  (not for extremities which are
123 // fixed to 0)
124 for (int i=0; i<=N-2; i++)
125     state_dynamics[i] = control[i];
126

```

```

122 // gain at timeframe TIME STEP SHOULD BE EQUAL TO 1 (or gain will be scaled
    wrongly)
123 // NB. if using implicit euler the state will correctly be the profile at the
    end of the time step (NOT if using midpoint !)
124 double alpha = constants[2];
125 state_dynamics[N-1] = gain_at_timeframe(state, control, N) / pow(1e0+alpha,time
    -1e0);
126 }
127
128 template <typename Variable>
129 inline void OCP::boundaryConditions(double initial_time, double final_time, const
    Variable *initial_state, const Variable *final_state, const Variable *
    parameters, const double *constants, Variable *boundary_conditions)
130 {
131     int N = (int) constants[0];
132
133     // initial profile P(t0) = P0
134     double distance, P0;
135     double DX = 1230 / N;
136     int verbose = 1;
137     for (int i=0; i<=N-2; i++)
138     {
139         distance = DX * (i+1);
140         P0 = bcp::interpolation1Dlinear(distance, distance_grid, p0_values, 0, verbose
            );
141         boundary_conditions[i] = initial_state[i] - P0;
142     }
143
144     // gain
145     boundary_conditions[N-1] = initial_state[N-1];
146     boundary_conditions[N] = final_state[N-1];
147 }
148
149 template <typename Variable>
150 inline void OCP::pathConstraints(double time, const Variable *state, const
    Variable *control, const Variable *parameters, const double *constants,
    Variable *path_constraints)
151 {
152     int N = (int) constants[0];
153     double DX = 1230 / N;
154     double kappa = 1e0; // max slope (+++ use table interpolation here)
155
156     // N profile constraints according to space discretization
157     // at time t_k for 0<=k<=N-1 (p_{i+1}^{k+1} - p_i^{k+1})/DX/kappa = s_i^k with p_0 =
    p_N = 0 at extremities
158     // IMPORTANT: note that constraint involves 'next' state p^{k+1} since p^0 is
    fixed by bounday conditions
159     // last call of pathcond occurs at penultimate time t_{T-1}, so constraint will be
    properly enforced for last state p^T
160
161     // recompute next state at end of time step
162     Variable next_state[N-1];
163     for (int i=0; i<=N-1; i++)
164         next_state[i] = state[i] + control[i];
165
166     // slope constraints
167     path_constraints[0] = (next_state[0] - 0e0) / DX / kappa;
168     path_constraints[N-1] = (0e0 - next_state[N-2]) / DX / kappa;
169     for (int i=1; i<=N-2; i++)
170         path_constraints[i] = (next_state[i] - next_state[i-1]) / DX / kappa;
171
172     // capacity constraint for current timeframe TIME STEP SHOULD BE EQUAL TO 1 (or

```

```

173     Cmax will be scaled wrongly)
174     // E = sum DX u_i^k
175     Variable capacity = 0e0;
176     double Cmax_per_time_step = constants[1];
177     for (int i=0; i<=N-2; i++)
178         capacity += DX * control[i];
179     path_constraints[N] = Cmax_per_time_step - capacity;
180 }
181
182 void OCP::preProcessing()
183 {
184     // initialize interpolations for p0 and G
185     bcp::readFileToVector("data/distance.data", distance_grid);
186     bcp::readFileToVector("data/p0.data", p0_values);
187     bcp::readFileToVector("data/depth.data", depth_grid);
188     bcp::readCSVToMatrix("data/gain.csv", G_values, ';', 0,1);
189
190     // +++ constants seems unaffected at this call -- (OCP::initialize())
191 }
192
193
194 // ////////////////////////////////////////
195 // explicit template instantiation for template functions, with double and
196 // double_ad
197 // +++ could be in an included separate file ?
198 // but needs to be done for aux functions too ? APPARENTLY NOT !
199 template void OCP::finalCost<double>(double initial_time, double final_time, const
200     double *initial_state, const double *final_state, const double *parameters,
201     const double *constants, double &final_cost);
202 template void OCP::dynamics<double>(double time, const double *state, const double
203     *control, const double *parameters, const double *constants, double *
204     state_dynamics);
205 template void OCP::boundaryConditions<double>(double initial_time, double
206     final_time, const double *initial_state, const double *final_state, const
207     double *parameters, const double *constants, double *boundary_conditions);
208 template void OCP::pathConstraints<double>(double time, const double *state, const
209     double *control, const double *parameters, const double *constants, double *
210     path_constraints);
211
212 template void OCP::finalCost<double_ad>(double initial_time, double final_time,
213     const double_ad *initial_state, const double_ad *final_state, const double_ad
214     *parameters, const double *constants, double_ad &final_cost);
215 template void OCP::dynamics<double_ad>(double time, const double_ad *state, const
216     double_ad *control, const double_ad *parameters, const double *constants,
217     double_ad *state_dynamics);
218 template void OCP::boundaryConditions<double_ad>(double initial_time, double
219     final_time, const double_ad *initial_state, const double_ad *final_state,
220     const double_ad *parameters, const double *constants, double_ad *
221     boundary_conditions);
222 template void OCP::pathConstraints<double_ad>(double time, const double_ad *state,
223     const double_ad *control, const double_ad *parameters, const double *
224     constants, double_ad *path_constraints);

```

## B.3 Local optimization (bocop) for 2D SOP - semicontinuous formulation

### B.3.1 Problem definition

```
1 # Definition file
2
3 # Dimensions
4 dim.state 172
5 dim.control 171
6 dim.boundaryconditions 173
7 dim.pathconstraints 401
8 dim.parameters 0
9 dim.constants 4
10
11 # Time interval
12 initial.time 0
13 final.time 2
14
15 # Constants
16 constant.0 20
17 constant.1 10
18 constant.2 1000000.0
19 constant.3 0.1
20
21 # Time discretisation
22 ode.discretization euler_implicit
23 time.steps 2
24
25 # Bounds for constraints
26 boundarycond.0.lowerbound 0
27 boundarycond.0.upperbound 0
28 ...
29 boundarycond.171.lowerbound 0
30 boundarycond.171.upperbound 0
31 boundarycond.172.lowerbound 1000.0
32 pathconstraint.0.lowerbound -1
33 pathconstraint.0.upperbound 1
34 ...
35 pathconstraint.399.lowerbound -1
36 pathconstraint.399.upperbound 1
37 pathconstraint.400.lowerbound 0
38
39 # Bounds for variables
40 state.0.lowerbound 0
41 state.0.upperbound 500
42 ...
43 state.170.upperbound 500
44 state.171.lowerbound 0
45 control.0.lowerbound 0
46 control.0.upperbound 500
47 ...
48 control.170.lowerbound 0
49 control.170.upperbound 500
50
51 # Initialization for discretized problem
52 state.0.init 10
53 ...
54 state.171.init 10
55 control.0.init 1
56 ...
57 control.170.init 1
58
59 # Names
60
61 # Ipopt
62 ipoptIntOption.print_level 5
```



```

63 ipoptIntOption.max_iter 5000
64 ipoptStrOption.mu_strategy monotone
65 ipoptNumOption.tol 1e-06
66
67 # Misc
68 ad.retape 1

1 # .def file generation for mine problem ('PDE formulation') 3D CASE
2 import bocop
3
4 # parameters
5 filename = 'problem.def'
6 N = 20
7 M = 10
8 T = 2
9 steps = T
10 Cmax_per_time_unit = 1e6
11 objective_lowerbound = 1e3 #1e5
12 alpha = 0.1
13 snorm = 1
14 # ipopt
15 maxiter = 5000
16 tol = 1e-6
17
18 # values
19 dim_state = (N-1)*(M-1) + 1 # interior grid points for profile + gain
20 dim_control = (N-1)*(M-1) # digging effort for interior grid points
21 dim_boundaryconditions = (N-1)*(M-1) + 2 # initial interior profile + initial
    gain + final gain
22 dim_pathconstraints = 2*M*N + 1 # X/Y slopes for points except farther boundaries
    + capacity limit
23
24 # later use default options ?
25 with open(filename, 'w') as deffile:
26     deffile.write('# Definition file\n\n')
27
28     deffile.write('# Dimensions\n')
29     deffile.write('dim.state ' + str(dim_state) + '\n')
30     deffile.write('dim.control ' + str(dim_control) + '\n')
31     deffile.write('dim.boundaryconditions ' + str(dim_boundaryconditions) + '\n')
32     deffile.write('dim.pathconstraints ' + str(dim_pathconstraints) + '\n')
33     deffile.write('dim.parameters 0\n')
34     deffile.write('dim.constants 4\n')
35
36     deffile.write('\n# Time interval\n')
37     deffile.write('initial.time 0\n')
38     deffile.write('final.time ' + str(T) + '\n')
39
40     deffile.write('\n# Constants\n')
41     deffile.write('constant.0 ' + str(N) + '\n')
42     deffile.write('constant.1 ' + str(M) + '\n')
43     deffile.write('constant.2 ' + str(Cmax_per_time_unit * T / steps) + '\n')
44     deffile.write('constant.3 ' + str(alpha) + '\n')
45
46     deffile.write('\n# Time discretisation\n')
47     deffile.write('ode.discretization euler_implicit\n')
48     deffile.write('time.steps ' + str(steps) + '\n')
49
50     deffile.write('\n# Bounds for constraints\n')
51     # initial profile and gain
52     for i in range(dim_boundaryconditions - 1):
53         deffile.write('boundarycond.' + str(i) + '.lowerbound 0\n')

```

```

54     deffile.write('boundarycond.'+str(i)+'.upperbound 0\n')
55     # final gain
56     deffile.write('boundarycond.'+str(dim_boundaryconditions-1)+'.lowerbound '+str
(objective_lowerbound)+'\n')
57     # profile slope constraints
58     for i in range(dim_pathconstraints-1):
59         deffile.write('pathconstraint.'+str(i)+'.lowerbound -'+str(snorm)+'\n')
60         deffile.write('pathconstraint.'+str(i)+'.upperbound '+str(snorm)+'\n')
61     # capacity constraint
62     deffile.write('pathconstraint.'+str(dim_pathconstraints-1)+'.lowerbound 0\n')
63
64     deffile.write('\n# Bounds for variables\n')
65     # profile bounds
66     for i in range(dim_state-1):
67         deffile.write('state.'+str(i)+'.lowerbound 0\n')
68         deffile.write('state.'+str(i)+'.upperbound 500\n')
69     # gain
70     deffile.write('state.'+str(dim_state-1)+'.lowerbound 0\n')
71     # digging bounds
72     for i in range(dim_control):
73         deffile.write('control.'+str(i)+'.lowerbound 0\n')
74         deffile.write('control.'+str(i)+'.upperbound 500\n')
75
76     deffile.write('\n# Initialization for discretized problem\n')
77     for i in range(dim_state):
78         deffile.write('state.'+str(i)+'.init 10\n')
79     for i in range(dim_control):
80         deffile.write('control.'+str(i)+'.init 1\n')
81
82     deffile.write('\n# Names\n\n# Ipopt\n')
83     deffile.write('ipoptIntOption.print_level 5\n')
84     deffile.write('ipoptIntOption.max_iter '+str(maxiter)+'\n')
85     deffile.write('ipoptStrOption.mu_strategy monotone\n')
86     deffile.write('ipoptNumOption.tol '+str(tol)+'\n')
87
88     deffile.write('\n# Misc\n')
89     deffile.write('ad.retape 1\n')

```

### B.3.2 Problem functions

```

1 // +++DRAFT+++ This class implements the OCP functions
2 // It derives from the generic class bocop3OCPBase
3 // OCP functions are defined with templates since they will be called
4 // from both the NLP solver (double arguments) and AD tool (ad_double arguments)
5 //#pragma once
6
7 #include <OCP.h>
8
9 // ////////////////////////////////////////
10
11 // space domain is discretized on a X/Y grid with N x M intervals
12 // this means (N+1)(M+1) points on the grid including the boundaries
13 // since profile is equal to 0 on the boundary, only the (N-1)(M-1) interior
14 // points are modeled as state variables
15 // this matrix is stored by columns ie 1D index is  $l = j * (N-1) + i$ 
16
17 // constants: [N, M, Cmax_per_time_unit, alpha]
18 double X_length = 1230;
19 double Y_length = 400;

```

```

20 std::vector<double> distanceX_grid, distanceY_grid, distanceZ_grid;
21 std::vector<std::vector<double>> p0_2D_values, G_blob;
22 std::vector<std::vector<std::vector<double>>> G_3D_values;
23
24 template<typename Variable> Variable getG(const double distanceX, const double
    distanceY, const Variable depth)
25 {
26
27 // custom 3D linear interpolation: first two interp2D on depth Di and Di+1 then
    interpolate between these two
28 int verbose = 0;
29 /* int k = bcp::locate(depth, distanceZ_grid, verbose);
30 double G0 = bcp::interpolation2Dbilinear(distanceX, distanceY, distanceX_grid,
    distanceY_grid, G_3D_values[k], verbose);
31 double G1 = bcp::interpolation2Dbilinear(distanceX, distanceY, distanceX_grid,
    distanceY_grid, G_3D_values[k+1], verbose);
32 Variable r = (depth - distanceZ_grid[k]) / (distanceZ_grid[k+1] -
    distanceZ_grid[k]);
33 Variable G = (1e0 - r) * G0 + r * G1;
34
35 */
36 Variable r = pow(pow(distanceX - 600,2)+pow(distanceY - 200,2)+pow(depth -
    350,2),0.5);
37 Variable G = -r+1000;
38
39 // rescale
40 return G/1e6 ;
41 }
42
43
44 template<typename Variable> Variable G_depth_integral(const Variable p0, const
    Variable depth, const double distanceX, const double distanceY)
45 {
46 // compute integral of g over [p0,depth] at x=distance and y=width +++ use
    rectangle instead ? ('right'/implicit rectangle ?)
47 Variable g_sum = 0e0;
48
49 // limit indices for integral along depth
50 int verbose = 0;
51 int j0 = bcp::locate(p0, distanceZ_grid, verbose);
52 int j1 = bcp::locate(depth, distanceZ_grid, verbose);
53 double d0, d1;
54 Variable G0, G1;
55
56 // partial first interval (p0,distanceZ_grid[j0+1])
57 d1 = distanceZ_grid[j0+1];
58 G0 = getG(distanceX, distanceY, p0);
59 G1 = getG(distanceX, distanceY, d1);
60 g_sum = g_sum + (G0 + G1) / 2e0 * (d1 - p0);
61
62 // complete intervals (distanceZ_grid[j],distanceZ_grid[j+1])
63 for (int j=j0+1; j<j1; j++)
64 {
65 d0 = distanceZ_grid[j];
66 d1 = distanceZ_grid[j+1];
67 G0 = getG(distanceX, distanceY, d0);
68 G1 = getG(distanceX, distanceY, d1);
69 g_sum = g_sum + (G0 + G1) / 2e0 * (d1 - d0);
70 }
71
72 // partial last interval (distanceZ_grid[j1],depth)
73 d0 = distanceZ_grid[j1];

```

```

74  G0 = getG(distanceX, distanceY, d0);
75  G1 = getG(distanceX, distanceY, depth);
76  g_sum = g_sum + (G0 + G1) / 2e0 * (depth - d0);
77
78  return g_sum;
79 }
80
81
82 template<typename Variable> Variable G_depth_integral_indices(const Variable *
    state, const Variable *control, const int i, const int j, const int N, const
    int M)
83 {
84     // note: on boundary the depth profile is always 0 so the integral of G along
    // depth is 0 as well
85     Variable G = 0e0;
86
87     // full grid is (N+1)(M+1) points with i|j indices from 0 to N|M
88     // interior is (N-1)(M-1) points with i|j indices from 1 to N-1|M-1
89     // interior point
90     if ((i > 0) && (i < N) && (j > 0) && (j < M))
91     {
92         int l = (j-1) * (N-1) + (i-1);
93         Variable depth_end = state[l];
94         Variable depth_start = depth_end - control[l];
95         double distanceX = X_length / N * i;
96         double distanceY = Y_length / M * j;
97         G = G_depth_integral(depth_start, depth_end, distanceX, distanceY);
98     }
99
100 /*     if ((i>-1) and (j>-1) and (i<N-2) and (j<M-2))
101     {
102         double DX = X_length / N;
103         double DY = Y_length / M;
104         int l = j * (N-1) + i;
105         Variable depth_end = state[l];
106         Variable depth_start = depth_end - control[l];
107         double distanceX = DX*(i+1);
108         double distanceY = DY*(j+1);
109         G = G_depth_integral(depth_start, depth_end, distanceX, distanceY);
110     }
111 */
112     return G;
113 }
114
115 template<typename Variable> Variable gain_at_timeframe(const Variable *state,
    const Variable *control, const int N, const int M)
116 {
117     // NB. if using implicit euler then state is at the end of the time step aka P
    // ^k+1
118     // previous state P^k is recomputed via the control
119
120     // gain for current timeframe +++ CHECK THIS ONE
121     // G = DX DY sum (g_ij^k + g_i+1j^k + g_i+1j+1^k + g_ij+1^k) / 4
122     // with g_i^k = int_Pik^Pik+1 G(Xi,p) dp
123     Variable gain = 0e0;
124
125     double DX = X_length / N;
126     double DY = Y_length / M;
127     Variable G00, G01, G10, G11;
128     double distanceX, distanceY;
129
130     // full grid is (N+1)(M+1) points with i|j indices from 0 to N|M

```

```

131 // interior is (N-1)(M-1) points with i|j indices from 1 to N-1|M-1
132 // summation for gain is done for indices i=0..N-1 and j=0..M-1 since we take
133 [i,i+1]x[j,j+1] cells
134 //+++ use something more centered here, with loop on all interior points
135 exactly ?
136 //for (int i=-1; i<=N-2; i++)
137 //    for (int j=-1; j<=M-2; j++)
138 for (int i=0; i<=N-1; i++)
139     for (int j=0; j<=M-1; j++)
140     {
141         // G00 ie g_i_j
142         G00 = G_depth_integral_indices(state, control, i, j, N, M);
143         // G01 ie g_i_j+1
144         G01 = G_depth_integral_indices(state, control, i, j+1, N, M);
145         // G10 ie g_i+1_j
146         G10 = G_depth_integral_indices(state, control, i+1, j, N, M);
147         // G11 ie g_i+1_j+1
148         G11 = G_depth_integral_indices(state, control, i+1, j+1, N, M);
149         // update gain with current cell integral
150         gain += DX * DY/ 4 * (G00 + G01 + G11 + G10);
151     }
152 }
153
154 // get profile at grid point (i,j) ie X= i DX and Y= j DY
155 template<typename Variable> Variable getPij(const int i, const int j, std::vector<
156 Variable> state, const int N, const int M)
157 {
158     // profile is 0 on boundary
159     Variable Pij = 0e0;
160
161     // interior point: retrieve profile value from 1D vector
162     // NB. first state (index 0) is grid point (1,1) since boundary is omitted !
163     // full grid is (N+1)(M+1) points with i|j indices from 0 to N|M
164     // interior is (N-1)(M-1) points with i|j indices from 1 to N-1|M-1
165     if ((i > 0) && (i < N) && (j > 0) && (j < M))
166         Pij = state[(j-1) * (N-1) + (i-1)];
167
168     return Pij;
169 }
170
171 template <typename Variable>
172 inline void OCP::finalCost(double initial_time, double final_time, const Variable
173 *initial_state, const Variable *final_state, const Variable *parameters, const
174 double *constants, Variable &final_cost)
175 {
176     // maximize final gain
177     final_cost = - final_state[stateSize()-1];
178 }
179
180 template <typename Variable>
181 inline void OCP::dynamics(double time, const Variable *state, const Variable *
182 control, const Variable *parameters, const double *constants, Variable *
183 state_dynamics)
184 {
185     int N = (int) constants[0];
186     int M = (int) constants[1];
187
188     // 2D (N-1)x(M-1) grid stored by columns ie 1D index is l = j * (N-1) + i

```

```

186 // digging progress:  $p_{ij}^{k+1} - p_{ij}^k = u_{ij}^k$  (not for boundaries which are
      fixed to 0)
187 for (int l=0; l<(N-1)*(M-1); l++)
188     state_dynamics[l] = control[l];
189
190 // gain at timeframe TIME STEP SHOULD BE EQUAL TO 1 (or gain will be scaled
      wrongly)
191 // NB. if using implicit euler the state will correctly be the profile at the
      end of the time step (NOT if using midpoint !)
192 double alpha = constants[3];
193 state_dynamics[(N-1)*(M-1)] = gain_at_timeframe(state, control, N, M) / pow(1e0+
      alpha,time-1e0);
194 }
195
196
197 template <typename Variable>
198 inline void OCP::boundaryConditions(double initial_time, double final_time, const
      Variable *initial_state, const Variable *final_state, const Variable *
      parameters, const double *constants, Variable *boundary_conditions)
199 {
200     int N = (int) constants[0];
201     int M = (int) constants[1];
202     double DX = X_length / N;
203     double DY = Y_length / M;
204
205     // initial profile  $P(t_0) = P_0$ 
206     double distanceX, distanceY, P0;
207     int verbose = 1;
208     int l;
209     for (int i=0; i<=N-2; i++)
210         for(int j=0; j<=M-2; j++)
211             {
212                 distanceX = DX * (i+1);
213                 distanceY = DY * (j+1);
214                 P0 = bcp::interpolation2Dbilinear(distanceX, distanceY, distanceX_grid,
                distanceY_grid, p0_2D_values, verbose);
215                 l = j*(N-1)+i;
216                 boundary_conditions[l] = initial_state[l] - P0;
217             }
218
219     // gain CI and CF
220     boundary_conditions[(N-1)*(M-1)] = initial_state[(N-1)*(M-1)];
221     boundary_conditions[(N-1)*(M-1)+1] = final_state[(N-1)*(M-1)];
222 }
223
224
225 template <typename Variable>
226 inline void OCP::pathConstraints(double time, const Variable *state, const
      Variable *control, const Variable *parameters, const double *constants,
      Variable *path_constraints)
227 {
228     // profile constraint: normalized slopes at each point must be in [-1,1] (NB.
      2 slopes per point, so twice the constraints !)
229     double kappa = 1e0; // max slope (+++ use table interpolation here)
230
231     // compute the two X,Y slopes at each point  $X_i, Y_j$ 
232     int N = (int) constants[0];
233     int M = (int) constants[1];
234     double DX = X_length / N;
235     double DY = Y_length / M;
236
237     // recompute next state at end of time step

```

```

238 //Variable next_state[(N-1)*(M-1)];
239 std::vector <Variable> next_state((N-1)*(M-1));
240
241 // note: could also use two i,j loops here...
242 for (int l=0; l<(N-1)*(M-1); l++)
243     next_state[l] = state[l] + control[l];
244
245 // slopes by forward difference ie (Pi+1,j - Pij) / Dx, same for j and Dy
246 // all points except boundaries i=N and j=M, ie i=0..N-1 x j=0..M-1
247 // total 2MN slopes constraints (some of which are trivial since P=0 on the i
248 // =0,j=0 boundaries, but simpler this way)
248 int k = 0;
249 for (int i=0; i<=N-1; i++)
250     for (int j=0; j<=M-1; j++)
251     {
252         // X slope
253         path_constraints[k++] = (getPij(i+1, j ,next_state, N, M) - getPij(i,
254         j,next_state, N, M)) / DX / kappa;
255         // Y slope
256         path_constraints[k++] = (getPij(i, j+1 ,next_state, N, M) - getPij(i,
257         j,next_state, N, M)) / DY / kappa;
258     }
259
260 // capacity constraint for current timeframe TIME STEP SHOULD BE EQUAL TO 1 (
261 // or Cmax will be scaled wrongly)
262 // E = sum DX DY u_ij^k
263 Variable capacity = 0e0;
264 double Cmax_per_time_step = constants[2];
265 for (int l=0; l<(N-1)*(M-1); l++)
266     capacity += DX * DY * control[l];
267 path_constraints[2*M*N] = Cmax_per_time_step - capacity;
268 }
269
270 void OCP::preProcessing()
271 {
272     // NB. constants are not available here --
273
274     // P0 is now 2D X,Y
275     std::cout << "read distanceX" << std::endl;
276     bcp::readFileToVector("data/distanceX.data", distanceX_grid);
277     std::cout << "read distanceY" << std::endl;
278     bcp::readFileToVector("data/distanceY.data", distanceY_grid);
279     std::cout << "read p0 2D" << std::endl;
280     bcp::readCSVToMatrix("data/p0_2D.csv",p0_2D_values,',',0,1);
281
282     // G is now 3D Z,X,Y
283     std::cout << "read distanceZ" << std::endl;
284     bcp::readFileToVector("data/distanceZ.data", distanceZ_grid);
285
286     std::cout << "read gain 3D" << std::endl;
287     bcp::readCSVToMatrix("data/gain_3D.csv",G_blob,',',0,1);
288     // reshape G in 3D
289     G_3D_values.resize(distanceZ_grid.size());
290     for (int k = 0; k<distanceZ_grid.size(); k++)
291     {
292         G_3D_values[k].resize(distanceX_grid.size());
293         for (int i = 0; i < distanceX_grid.size(); i++)
294         {
295             G_3D_values[k][i].resize(distanceY_grid.size());
296             for (int j=0; j < distanceY_grid.size(); j++)

```

```

296         G_3D_values[k][i][j] = G_blob[k*distanceX_grid.size() + i][j];
297     }
298 }
299 std::cout << "Gain 3D is " << G_3D_values.size() << " x " << G_3D_values[0].
size() << " x " << G_3D_values[0][0].size() << std::endl;
300
301 }
302
303 // ////////////////////////////////////////
304 // explicit template instantiation for template functions, with double and
double_ad
305 // +++ could be in an included separate file ?
306 // but needs to be done for aux functions too ? APPARENTLY NOT !
307
308 template void OCP::finalCost<double>(double initial_time, double final_time, const
double *initial_state, const double *final_state, const double *parameters,
const double *constants, double &final_cost);
309 template void OCP::dynamics<double>(double time, const double *state, const double
*control, const double *parameters, const double *constants, double *
state_dynamics);
310 template void OCP::boundaryConditions<double>(double initial_time, double
final_time, const double *initial_state, const double *final_state, const
double *parameters, const double *constants, double *boundary_conditions);
311 template void OCP::pathConstraints<double>(double time, const double *state, const
double *control, const double *parameters, const double *constants, double *
path_constraints);
312
313 template void OCP::finalCost<double_ad>(double initial_time, double final_time,
const double_ad *initial_state, const double_ad *final_state, const double_ad
*parameters, const double *constants, double_ad &final_cost);
314 template void OCP::dynamics<double_ad>(double time, const double_ad *state, const
double_ad *control, const double_ad *parameters, const double *constants,
double_ad *state_dynamics);
315 template void OCP::boundaryConditions<double_ad>(double initial_time, double
final_time, const double_ad *initial_state, const double_ad *final_state,
const double_ad *parameters, const double *constants, double_ad *
boundary_conditions);
316 template void OCP::pathConstraints<double_ad>(double time, const double_ad *state,
const double_ad *control, const double_ad *parameters, const double *
constants, double_ad *path_constraints);

```

too long





# Bibliography

- [1] M. Aguiar, J. Van-Dierdonck, J. Mar, and N. Stollenwerk. “The role of mild and asymptomatic infections on COVID-19 vaccines performance: a modeling study”. In: *Journal of Advanced Research* 39 (2022), pp. 157–166.
- [2] A. Altarovici, O. Bokanowski, and H. Zidani. “A general Hamilton-Jacobi framework for non-linear state-constrained control problems”. In: *ESAIM: Control, Optimisation and Calculus of Variations* 19.2 (2013), pp. 337–357.
- [3] F. Alvarez, J. Amaya, A. Griewank, and N. Strogies. “A continuous framework for open pit mine planning”. In: *Mathematical methods of operations research* 73.1 (2011), pp. 29–54.
- [4] J. Amaya, C. Hermosilla, and E. Molina. “Optimality conditions for the continuous model of the final open pit problem”. In: *Optimization Letters* 15.3 (2021), pp. 991–1007.
- [5] R. Anderson. “Discussion: the Kermack-McKendrick epidemic threshold theorem”. In: *Bulletin of mathematical biology* 53.1 (1991), pp. 1–32.
- [6] G. Angelov, R. Kovacevic, N. Stilianakis, and V. Veliov. “Optimal vaccination strategies using a distributed epidemiological model applied to COVID-19”. In: (2021).
- [7] *Anuario de la minería de Chile*. Tech. rep. Sernageomin, 2020.
- [8] R. Arbel et al. “Effectiveness of a second BNT162b2 booster vaccine against hospitalization and death from COVID-19 in adults aged over 60 years”. In: *Nature medicine* (2022), pp. 1–5.
- [9] M. Aronna, R. Guglielmi, and L. Moschen. “A model for COVID-19 with isolation, quarantine and testing as control measures”. In: *Epidemics* 34 (2021), p. 100437.
- [10] U. Ascher, R. Mattheij, and R. Russell. *Numerical solution of boundary value problems for ordinary differential equations*. SIAM, 1995.
- [11] J.-P. Aubin. *Viability Theory*. Birkhäuser, Boston, MA, 2009.
- [12] F. Avram, L. Freddi, and D. Goreac. “Optimal control of a SIR epidemic with ICU constraints and target objectives”. In: *Applied Mathematics and Computation* 418 (2022), p. 126816.
- [13] M. Bardi and I. Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Vol. 12. Springer, 1997.
- [14] E. Barron. “The Pontryagin maximum principle for minimax problems of optimal control”. In: *Nonlinear Analysis: Theory, Methods & Applications* 15.12 (1990), pp. 1155–1165.
- [15] E. Barron and H. Ishii. “The Bellman equation for minimizing the maximum cost”. In: *Nonlinear Analysis: Theory, Methods & Applications* 13.9 (1989), pp. 1067–1090.
- [16] E. Barron, R. Jensen, and W. Liu. “The  $L^\infty$  control problem with continuous control functions”. In: *Nonlinear Analysis: Theory, Methods & Applications* 32.1 (1998), pp. 1–14.

- [17] H. Behncke. “Optimal control of deterministic epidemics”. In: *Optimal control applications and methods* 21.6 (2000), pp. 269–285.
- [18] B. M. Bell. “CppAD: a package for C++ algorithmic differentiation”. In: *Computational Infrastructure for Operations Research* (2012).
- [19] J. T. Betts. *Practical methods for optimal control using nonlinear programming*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2001.
- [20] P.-A. Bliman and M. Duprez. “How best can finite-time social distancing reduce epidemic final size?” In: *Journal of theoretical biology* 511 (2021), p. 110557.
- [21] P.-A. Bliman, M. Duprez, Y. Privat, and N. Vauchelet. “Optimal immunity control and final size minimization by social distancing for the sir epidemic model”. In: *Journal of Optimization Theory and Applications* 189.2 (2021), pp. 408–436.
- [22] N. Boland, C. Fricke, and G. Froyland. “A strengthened formulation for the open pit mine production scheduled problem”. In: *Preprint, University of Melbourne, Parkville, VIC 3010* (2006).
- [23] L. Bolzoni, E. Bonacini, R. Della Marca, and M. Groppi. “Optimal control of epidemic size and duration with limited resources”. In: *Mathematical biosciences* 315 (2019), p. 108232.
- [24] L. Bolzoni, E. Bonacini, C. Soresina, and M. Groppi. “Time-optimal control strategies in SIR epidemic models”. In: *Mathematical biosciences* 292 (2017), pp. 86–96.
- [25] F. Bonnans and A. Hermant. “Revisiting the analysis of optimal control problems with several state constraints”. In: *Control Cybernet* 38.4A (2009), pp. 1021–1052.
- [26] F. Bonnans, P. Martinon, D. Giorgi, V. Grelard, S. Maindrault, and O. Tissot. *BOCOP - A toolbox for optimal control problems*. <http://bocop.org>. 2019.
- [27] F. Bonnans et al. *Bocop – A collection of examples*. Tech. rep. INRIA, 2017.
- [28] R. Bulirsch, J. Stoer, and J. Stoer. *Introduction to numerical analysis*. Vol. 3. Springer, 2002.
- [29] L. Caccetta. “Application of optimisation techniques in open pit mining”. In: *Handbook of operations research in natural resources*. Springer, 2007, pp. 547–559.
- [30] L. Caccetta and S. P. Hill. “An application of branch and cut to open pit mine scheduling”. In: *Journal of global optimization* 27.2 (2003), pp. 349–365.
- [31] J.-B. Caillaud, O. Cots, and J. Gergaud. “Differential pathfollowing for regular optimal control problems”. In: *Optim. Methods Softw.* 27.2 (2012), pp. 177–196.
- [32] I. Capuzzo-Dolcetta and P.-L. Lions. “Hamilton-Jacobi equations with state constraints”. In: *Transactions of the American mathematical society* 318.2 (1990), pp. 643–683.
- [33] J. Caulkins et al. “The optimal lockdown intensity for COVID-19”. In: *Journal of Mathematical Economics* 93 (2021), p. 102489.
- [34] L. Cesari. *Optimization—theory and applications: problems with ordinary differential equations*. Vol. 1. Springer, New York, NY, 1983.
- [35] F. Clarke. *Functional analysis, calculus of variations and optimal control*. Vol. 264. Springer, 2013.
- [36] F. Clarke. *Optimization and nonsmooth analysis*. SIAM, 1990.
- [37] F. Clarke and R. Vinter. “The relationship between the maximum principle and dynamic programming”. In: *SIAM Journal on Control and Optimization* 25.5 (1987), pp. 1291–1311.
- [38] E. Cristiani and P. Martinon. “Initialization of the shooting method via the Hamilton-Jacobi-Bellman approach”. In: *Journal of Optimization Theory and Applications* 146.2 (2010), pp. 321–346.

- [39] B. Denby and D. Schofield. “Open-pit design and scheduling by use of genetic algorithms”. In: *Transactions of the Institution of Mining and Metallurgy. Section A. Mining Industry* 103 (1994).
- [40] S. Di Marco and R. González. “A numerical procedure for minimizing the maximum cost”. In: *System Modelling and Optimization*. Springer, 1996, pp. 285–291.
- [41] S. Di Marco and R. González. “Minimax optimal control problems. Numerical analysis of the finite horizon case”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 33.1 (1999), pp. 23–54.
- [42] A. Dontchev and R. Rockafellar. *Implicit functions and solution mappings*. Vol. 543. Springer, 2014.
- [43] I. Ekeland and M. Queyranne. “Optimal pits and optimal transportation”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 49.6 (2015), pp. 1659–1670.
- [44] N. Espejo, P. Nancel-Penard, and N. Morales. “A methodology for automatic ramp design in open pit mines”. In: *Journal of Mining Engineering and Research* 1.2 (2019), pp. 87–93.
- [45] D. Espinoza, M. Goycoolea, E. Moreno, and A. Newman. “MineLib: a library of open pit mining problems”. In: *Annals of Operations Research* 206.1 (2013), pp. 93–114.
- [46] M. Falcone and R. Ferretti. *Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations*. SIAM, 2013.
- [47] J. Ferland, J. Amaya, and M. Djuimo. “Application of a particle swarm algorithm to the capacitated open pit mining problem”. In: *Autonomous robots and agents*. Springer, 2007, pp. 127–133.
- [48] W. Fleming and R. Rishel. *Deterministic and stochastic optimal control*. Vol. 1. Springer Science & Business Media, 2012.
- [49] L. Freddi. “Optimal control of the transmission rate in compartmental epidemics”. In: *Mathematical Control & Related Fields* (2021).
- [50] R. Gamkrelidze. “Discovery of the maximum principle”. In: *Journal of dynamical and control systems* 5.4 (1999), pp. 437–451.
- [51] J. Gianatti, L. Aragone, P. Lotito, and L. Parente. “Solving minimax control problems via nonsmooth optimization”. In: *Operations Research Letters* 44.5 (2016), pp. 680–686.
- [52] M. Giaquinta and S. Hildebrandt. *Calculus of variations I*. Vol. 311. Springer Science & Business Media, 2013.
- [53] G. Giordano et al. “Modeling vaccination rollouts, SARS-CoV-2 variants and the requirement for non-pharmaceutical interventions in Italy”. In: *Nature medicine* 27.6 (2021), pp. 993–998.
- [54] R. González and L. Aragone. “A Bellman’s equation for minimizing the maximum cost”. In: *Indian Journal of Pure and Applied Mathematics* 31.12 (2000), pp. 1621–1632.
- [55] A. Griewank and N. Strogies. “Duality results for stationary problems of open pit mine planning in a continuous function framework”. In: *Computational & Applied Mathematics* 30.1 (2011), pp. 197–215.
- [56] E. Grigorieva, E. Khailov, and A. Korobeinikov. “Optimal quarantine strategies for COVID-19 control models”. In: *arXiv preprint arXiv:2004.10614* (2020).
- [57] H. Hermes and J. La Salle. *Functional Analysis and Time Optimal Control*. Vol. 56. Academic Press, New York, NY, 1969.
- [58] D. Hochbaum and A. Chen. “Performance analysis and best implementations of old and new algorithms for the open-pit mining problem”. In: *Operations Research* 48.6 (2000), pp. 894–914.

- [59] W. Hustrulid, M. Kuchta, and R. Martin. *Open pit mine planning and design, two volume set & CD-ROM pack*. CRC Press, 2013.
- [60] E. Jélvez, N. Morales, and H. Askari-Nasab. “A new model for automated pushback selection”. In: *Computers & Operations Research* 115 (2020), p. 104456.
- [61] T. Johnson. *Optimum open pit mine production scheduling*. University of California, Berkeley, 1968.
- [62] T. B. Johnson and W. R. Sharp. *A Three-dimensional dynamic programming method for optimal ultimate open pit design*. Vol. 7553. Bureau of Mines, US Department of the Interior, 1971.
- [63] J. Jones. “Notes on R0”. In: *California: Department of Anthropological Sciences* 323 (2007), pp. 1–19.
- [64] M. Jones and M. M. Peet. “A generalization of Bellman’s equation with application to path planning, obstacle avoidance and invariant set estimation”. In: *Automatica* 127 (2021), p. 109510.
- [65] M. Jones and M. M. Peet. “Extensions of the dynamic programming framework: Battery scheduling, demand charges, and renewable integration”. In: *IEEE Transactions on Automatic Control* 66.4 (2020), pp. 1602–1617.
- [66] J. Jost, J. Jost, and X. Li-Jost. *Calculus of variations*. Vol. 64. Cambridge University Press, 1998.
- [67] M. Kantner and T. Koprucki. “Beyond just “flattening the curve”: Optimal control of epidemics with purely non-pharmaceutical interventions”. In: *Journal of Mathematics in Industry* 10.1 (2020), pp. 1–23.
- [68] J. Kennedy. “Vaccine hesitancy: a growing concern”. In: *Pediatric drugs* 22.2 (2020), pp. 105–111.
- [69] W. Kermack and A. McKendrick. “A contribution to the mathematical theory of epidemics”. In: *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* 115.772 (1927), pp. 700–721.
- [70] W. Kermack and A. McKendrick. “A contribution to the mathematical theory of epidemics”. In: *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* 115.772 (1927), pp. 700–721.
- [71] D. Ketcheson. “Optimal control of an SIR epidemic through finite-time non-pharmaceutical intervention”. In: *Journal of Mathematical Biology* 83.1 (2021), pp. 1–21.
- [72] R. Kovacevic, N. Stilianakis, and V. Veliov. “A Distributed Optimal Control Model Applied to COVID-19 Pandemic”. In: *SIAM Journal on Control and Optimization* 60.2 (2022), S221–S245.
- [73] E. Lee and L. Markus. *Foundations of optimal control theory*. Tech. rep. Minnesota Univ Minneapolis Center For Control Sciences, 1967.
- [74] H. Lerchs and I. Grossman. “Optimum Design of Open-Pit Mines”. In: (1965), pp. 47–54.
- [75] M. Li. *An introduction to mathematical modeling of infectious diseases*. Vol. 2. Springer, 2018.
- [76] C. Lobry. *Qu’est ce que le pic d’une épidémie et comment le contrôler*. Cassini, 2021.
- [77] G. Lv, J. Yuan, X. Xiong, and M. Li. “Mortality rate and characteristics of deaths following COVID-19 vaccination”. In: *Frontiers in Medicine* (2021), p. 649.
- [78] P. Maclean, A. Mentzer, T. Lambe, and J. Knight. “Why do breakthrough COVID-19 infections occur in the vaccinated?” In: *QJM: An International Journal of Medicine* 115.2 (2022), pp. 67–68.

- [79] E. Molina and J. Amaya. “Analytical Properties of the Feasible and Optimal Profiles in the Binary Programming Formulation of Open Pit”. In: *Application of Computers and Operations Research in the Mineral Industry*. Golden, Colorado USA, 2017.
- [80] E. Molina, P. Martinon, and H. Ramírez. “Optimal control approaches for Open Pit planning”. working paper or preprint. Feb. 2022.
- [81] E. Molina and A. Rapaport. “An optimal feedback control that minimizes the epidemic peak in the SIR model under a budget constraint”. In: *arXiv preprint arXiv:2203.05800* (2022).
- [82] E. Molina, A. Rapaport, and H. Ramirez. “Equivalent formulations of optimal control problems with maximum cost and applications”. In: *arXiv preprint arXiv:2202.12545* (2022).
- [83] D. H. Morris, F. W. Rossine, J. B. Plotkin, and S. A. Levin. “Optimal, near-optimal, and robust epidemic control”. In: *Communications Physics* 4.1 (2021), pp. 1–8.
- [84] A. Newman, E. Rubio, R. Caro, A. Weintraub, and K. Eurek. “A review of operations research in mine planning”. In: *Interfaces* 40.3 (2010), pp. 222–245.
- [85] J. Nocedal and S. Wright. *Numerical optimization*. Springer, 1999.
- [86] A. Palmer, Z. Zabinsky, and S. Liu. “Optimal control of COVID-19 infection rate with social costs. arXiv 2020”. In: *arXiv preprint arXiv:2007.13811* ().
- [87] L. Pontryagin, V. Boltyanski, R. Gamkrelidze, and E. Michtchenko. *The Mathematical Theory of Optimal Processes*. Wiley Interscience, New York, 1962.
- [88] A. Ramos, M. Vela-Pérez, M. Ferrández, A. Kubik, and B. Ivorra. “Modeling the impact of SARS-CoV-2 variants and vaccines on the spread of COVID-19”. In: *Communications in Nonlinear Science and Numerical Simulation* 102 (2021), p. 105937.
- [89] A. Roghani et al. “The influence of COVID-19 vaccination on daily cases, hospitalization, and death rate in Tennessee, United States: Case Study”. In: *JMIRx med* 2.3 (2021), e29324.
- [90] J. Saavedra-Rosas, E. Jeivez, J. Amaya, and N. Morales. “Optimizing open-pit block scheduling with exposed ore reserve”. In: *Journal of the Southern African Institute of Mining and Metallurgy* 116.7 (2016), pp. 655–662.
- [91] S. Sethi. *Optimal Control Theory. Applications to Management Science and Economics*. Vol. 3. Springer, Cham, 2019.
- [92] E. Sontag. *Mathematical control theory: deterministic finite dimensional systems*. Vol. 6. Springer Science & Business Media, 2013.
- [93] N. Strogies and A. Griewank. “A PDE constraint formulation of open pit mine planning problems”. In: *PAMM* 13.1 (2013), pp. 391–392.
- [94] L. Tang et al. “Asymptomatic and symptomatic SARS-CoV-2 infections after BNT162b2 vaccination in a routinely screened workforce”. In: *Jama* 325.24 (2021), pp. 2500–2502.
- [95] T. Tao. *An introduction to measure theory*. Vol. 126. American Mathematical Society Providence, 2011.
- [96] E. Trélat. *Contrôle optimal: théorie & applications*. Vol. 36. Vuibert Paris, 2005.
- [97] R. Vinter. *Optimal control*. Springer, 2010.
- [98] A. Waechter and L. T. Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical Programming Series A* 106 (2006), pp. 25–57.
- [99] W. Walter. *Ordinary Differential Equations*. Springer, New York, NY, 1998.

- 
- [100] H. Weiss. “The SIR model and the foundations of public health”. In: *Materials mathematics* (2013), pp. 0001–17.
- [101] E. A. Wright. “The use of dynamic programming for open pit mine design: some practical implications”. In: *Mining Science and Technology* 4.2 (1987), pp. 97–104.





**Abstract**

Optimal control techniques have numerous applications in engineering and real world problems. This thesis is devoted to using these techniques in two contexts, mining and epidemiology, dividing this document in two respective parts.

In the first part related to mining, we work with the continuous formulation of the Open Pit Problem consisting of finding the optimal shape of an opencast mine representing its profile by a continuous function. Optimality in this context corresponds to maximizing the profit of mineral extraction. We introduce for the first time optimal control models of this problem. We present optimality conditions of solutions along with numerical experiments using local and global methods.

Another relevant problem in this context corresponds to the Sequential version of the Open Pit Problem, which consists of scheduling an extraction program over consecutive time frames (for example, a profile each 6 months), finding nested profiles maximizing a discounted profit. We proposed a novel semi-continuous model to obtain solutions of the sequential problem and we use it to present for the first time, to the best of our knowledge, numerical solutions of a three dimensional case (a possible real world mine) including the original Open Pit Problem (case with a single time-frame).

In the second part we deal with optimal control problems minimizing the maximum value of a state. This problematic was inspired by Covid-19, where hospitals and ICU beds were overcrowded due to a high amount of simultaneous infections. We present four different reformulations of this kind of optimal control problem as a Mayer one, each one having its pros and cons. We present the numerical performance of each formulation in an academic example and in a more realistic SIR model where the problem corresponds to minimizing the peak of infectious compartment with integral constraint in the control. With respect to the latter problem, we prove analytically that the structure of the optimal control is null-singular-null and we used it to assess numerical solutions.

**Keywords:** optimal control, natural system management, mining, epidemiology

---

**Résumé**

Les techniques de contrôle optimale ont plusieurs applications dans le domaines de l'ingenierie et les problèmes de la vie pratique. Cette thèse est consacrée a l'utilisation de ces techniques dans deux contextes, l'exploitation minière et l'épidémiologie, divisant ce document en deux parties respectives.

Dans la première partie relative à l'exploitation minière, nous travaillons avec la formulation continue du problème Final Open Pit consistant à trouver la forme optimale d'une mine à ciel ouvert représentant son profil par une fonction continue. L'optimalité dans ce contexte correspond à maximiser le bénéfice de l'extraction minérale. Nous présentons pour la première fois des modèles de contrôle optimal pour ce problème. Nous présentons des conditions optimales de solutions avec des simulations numériques utilisant des méthodes locales et globales.

La version séquentielle du Final Open Pit est également très pertinent dans le contexte de cette thèse. Le problème consiste à planifier un programme d'extraction sur des périodes de temps consécutives (par exemple, un profil tous les 6 mois), en trouvant des profils imbriqués maximisant un bénéfice actualisé. Nous avons formulé un nouveau modèle semi-continu permettant d'obtenir des solutions du problème séquentiel et que nous utilisons pour proposer des solutions numériques dans un cas tridimensionnelle ( une mine possible du monde réel) y compris le problème Open Pit original (cas avec une seule période de temps). Cela, à notre connaissance, n'avait jamais été fait auparavant.

Dans la deuxième partie, nous traitons les problèmes de contrôle optimal minimisant la valeur maximale d'un état. Ce problème a été inspiré par le contexte sanitaire très difficile suite a la covid-19, où avoir de beaucoup infectés en même temps sature les hôpitaux et les lits de soin intensif. Nous présentons quatre reformulations différentes de ce type de problème comme un Mayer de contrôle optimal, ayant chacun ses avantages et ses inconvénients. Nous évaluons la performance numérique de chaque formulation dans un exemple académique et dans un modèle SIR plus réaliste où le problème correspond à minimiser le pic du compartiment infecté avec une contrainte intégrale dans le contrôle. En ce qui concerne ce dernier problème, nous prouvons analytiquement que la structure du contrôle optimal est null-singular-null et nous l'avons utilisé pour évaluer les solutions numériques.

**Mots clés :** contrôle optimal, gestion du système naturel, exploitation minière, épidémiologie

---