Hamilton-Jacobi Approach for State-Constrained Differential Games and Numerical Learning Methods for Optimal Control Problems

Nidhal Gammoudi

INSTITUT POLYTECHNIQUE DE PARIS

ENSTA IP PARIS

NNT : 2021IPPAE005

Thèse de doctorat

Hamilton-Jacobi Approach for State-Constrained Differential Games and Numerical Learning Methods for Optimal Control Problems

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Ecole Nationale Supérieure des Techniques Avancées (ENSTA)

École doctorale n° 574 de Mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques Appliquées

Thèse présentée et soutenue à Palaiseau, le 12 avril 2021, par

NIDHAL GAMMOUDI

Composition du Jury :

M. Quentin MÉRIGOT Professeur, Université Paris Saclay	Président
M. Maurizio FALCONE Professeur, Université de Rome La Sapienza	Rapporteur
M. Peter Dower Professeur, Université de Melbourne	Rapporteur
M. Nicolas Forcadel Professeur, Institut National des Sciences Appliquées de Rouen	Examinateur
M. Xavier Warin Ingénieur de recherche, EDF R&D	Examinateur
M. Olivier Bokanowski Maître de conférences, Université Paris-Diderot	Examinateur
Mme. Hasnaa Zidani Professeur, Ecole Nationale Supérieure des Techniques Avancées	Directrice de thèse

Remerciements

Ces quelques lignes sont pour moi l'occasion d'exprimer ma gratitude envers tous ceux qui m'ont accompagné dans cette thèse.

Je tiens à remercier d'abord ma directrcie de thèse Hasnaa Zidani pour sa disponibilité et la grande confiance qu'elle m'a accordée pour mener à bien ce travail de recherche. Je remercie également Olivier Bokanowski qui a participé à l'encadrement d'une grande partie de ma thèse. Je suis très reconnaissant pour le temps qu'ils ont accordé à mon suivi et à ma formation.

Mes remerciements s'adressent également aux professeurs Maurizio Falcone et Peter Dower pour m'avoir fait l'honneur d'être les rapporteurs de ma thèse, pour leurs lectures attentives du manuscrit et leurs remarques pertinentes.

Je tiens à remercier aussi les autres membres du jury, les professeurs Quentin Mérigot, Nicolas Forcadel et Xavier Warin. Un remerciement tout particulier à Xavier Warin et aussi à Nicolas Langrene pour leur collaboration et leurs remarques précieuses sur la dernière partie de mon travail.

Je remercie tous les membres de l'Unité des Mathématiques Appliquées (UMA) de l'Ecole Nationale Supérieure des Techniques Avancées (ENSTA) de Paris pour avoir contribuer à rendre mon environnement de travail particulièrement accueillant et amical. Un remerciement spécial à Frederic Jean pour son infinie disponibilité et sa grande gentillesse et à Sourour Elloumi pour ses conseils et son encouragement. Je tiens à remercier Anya Desilles pour sa disponibilité et son aide précieuse au début de ma thèse et Eliane Becache pour m'avoir aider sur le plan administratif en tant que directrice adjointe de l'Ecole Doctorale de Mathématiques Hadamard EDMH. Un grand merci à Corinne Chen, Maurice Diamantini et Christophe Mathulik pour leur aide, leur gentiellesse et leur disponibilité. Je remercie aussi Pierre Carpentier, Christophe Hazard, Zacharie Ales, Francessco Rosso et Jerome Perez avec qui les discussions étaient agréables. Un remerciement spécial aux doctorants de l'UMA pour leur soutien, leur aide et les échanges que nous avons eu ensemble pendant les pauses. Ainsi, je remercie Mahrane, Othmane, Jean-François, Veljko, Akram.

Enfin je remercie tous les membres de ma famille pour m'avoir soutenu pendant tout mon parcours en particulier durant mon projet de thèse. Je remercie également tous mes amis qui m'ont été proches.

Contents

1 Introduction **7**

1.1 Differential games . . . 10

1.2 Numerical learning methods for optimal control problems . . . 12

1.2.1 Optimistic planning approach . . . 13

1.2.2 Deep learning numerical methods for dynamic programming . . . 14

2 Unconstrained Finite Horizon Differential Games **16**

2.1 Introduction . . . 16

2.2 Definitions and hypothesis . . . 18

2.3 Unconstrained problem with delay strategies . . . 19

2.3.1 Problem formulation . . . 19

2.3.2 Some properties of value functions . . . 21

2.3.3 Characterization of value functions . . . 23

2.4 Unconstrained problem with nonanticpative strategies . . . 27

2.4.1 Problem formulation . . . 27

2.4.2 Characterization of value functions . . . 27

2.5 General comparison result . . . 31

2.6 Approximation by discrete time games and trajectory reconstruction . . . 31

2.6.1 Approximation by discrete time games . . . 32

2.6.2 Trajectory reconstruction . . . 35

2.7 A game example . . . 43

2.7.1 Problem of the first player . . . 44

2.7.2 Problem of the second player . . . 44

3 Hamilton-Jacobi Approach For Differential Game Problems With State Constraints **45**

3.1 Introduction . . . 45

3.2 Problem formulation . . . 48

3.2.1 Settings of the constrained differential game . . . 48

3.2.2 Associated auxiliary problem . . . 49

3.3 Properties of the value functions v and w . . . 50

3.4 Trajectory reconstruction based on the value function and approximation by discrete time games . . . 57

3.4.1 A general reconstruction procedure . . . 58

3.4.2 Reconstruction with a *specific* approximation . . . 63

3.5 Application to an aircraft landing problem . . . 71

3.5.1 Introduction . . . 71

3.5.2 5D differential game model . . . 72

3.6 Appendix: Properties of the auxiliary value function w . . . 76

4 Optimistic Planning Algorithms For Constrained Optimal Control Problems **82**
4.1 Introduction . . . 82
4.2 Problem formulation and discrete settings . . . 84
4.3 Preliminary results . . . 87
4.4 Optimistic planning . . . 89
4.4.1 Optimistic planning approach . . . 89
4.4.2 Optimistic Planning (*OP*) Algorithm . . . 92
4.4.3 Simultaneous Optimistic Planning (*SOP*) Algorithm . . . 93
4.4.4 Simultaneous Optimistic Planning with Multiple Steps (*SOPMS*) Algorithm . . . 94
4.4.5 Resolution procedure for a constrained problem . . . 95
4.5 Extension to infinite horizon problems . . . 96
4.6 Numerical experimentation . . . 99
4.6.1 Choice of the numerical parameters . . . 99
4.6.2 Numerical examples . . . 100
4.7 Appendix . . . 111
4.7.1 Appendix: Proofs of convergence results for *OP* and *SOP* algorithms . . . 111
4.7.2 Appendix B. Numerical parameters of example 4 . . . 117

5 Deep Learning Numerical Methods For Dynamic Programming **119**
5.1 Introduction . . . 119
5.2 Neural networks for functions approximation . . . 121
5.3 Problem settings . . . 125
5.4 Deep learning numerical methods . . . 126
5.4.1 Neural networks for dynamic programming (DP) . . . 126
5.4.2 Neural networks for partial derivatives equations (PDE) . . . 128
5.5 Numerical examples . . . 129

6 Conclusion and perspectives **147**

Abbreviations and Notations

DPP	Dynamic Programming Principle
PDE	Partial Derivatives Equation
HJ	Hamilton-Jacobi
HJB	Hamilton-Jacobi-Belman
HJI	Hamilton-Jacobi-Isaacs
u.s.c	upper semi continuous
l.s.c	lower semi continuous
DNN	Deep Neural Networks
$\mathbb{R}^p$	Euclidean p-dimensional space
$\mathbb{R}^{p\times q}$	Space of $(p \times q)$ real matrices
I_p	The identity matrix of size p
$\langle \cdot, \cdot \rangle$	The inner product in some Euclidean space
$\| \cdot \|$	The Euclidean norm
$\mathbb{B}_{\mathbb{R}^p}$	The closed unit ball of $\mathbb{R}^p$
$\mathbb{B}(x, r)$	The open ball of center x and radius $r > 0$
$\overline{\mathbb{B}}(x, r)$	The closed ball of center x and radius $r > 0$
$A \bigcap B$	The intersection of two sets A and B
$A \bigcup B$	The union of two sets A and B
$\overset{\circ}{Y}$	The interior of a set $Y \subset \mathbb{R}^p$
∂Y	The boundary of a set $Y \subset \mathbb{R}^p$
$d(\cdot, Y)$	The distance function to a set $Y \subset \mathbb{R}^p$
$d_Y(\cdot)$	The signed distance function to a set $Y \subset \mathbb{R}^p$
$D_x\phi$	The gradient of ϕ w.r.t. x
$\partial_t\phi$	The partial derivative of ϕ w.r.t. t
$\lfloor \cdot \rfloor$	The floor function
$\lceil \cdot \rceil$	The ceiling function
$a \bigvee b$	$\max(a, b)$, for $a, b \in \mathbb{R}$
$a \bigwedge b$	$\min(a, b)$, for $a, b \in \mathbb{R}$
$\|U\|$	The cardinality of a finite set U
$X \sim \mu$	The random variable X is described by the probability distribution μ

Synthèse(en français)

L'objectif de cette thèse est d'étudier des problèmes de jeux différentiels avec contraintes d'état par l'approche Hamilton-Jacobi et de développer des méthodes numériques d'apprentissage pour résoudre des problèmes de commande optimale.

La théorie de commande optimale est une branche de mathématiques appliquées qui s'intéresse à l'étude de l'évolution d'un système dynamique dans le temps afin de minimiser un coût, maximiser un gain, atteindre une cible finale ou stabiliser le système. L'évolution temporelle du système définit une trajectoire régie par des équations différentielles ordinaires pour les problèmes de commande optimale déterministes. Cette trajectoire est directement affectée par les actions d'un contrôleur dites lois de contrôle admissibles si elles satisfont certaines conditions. D'autre part, dans de nombreuses applications, l'espace d'état peut être restreint, ce qui définit certaines contraintes qui doivent être respectées par la trajectoire au cours de l'évolution du système.

Parmi les principales approches pour étudier les problèmes de commande optimale, on peut trouver dans la littérature l'approche de *Programmation Dynamique* (PD) formulée par Richard Bellman dans les années 1950, voir [18, 19], qui considère la valeur optimale du problème d'optimisation en fonction de la condition initiale définissant ainsi la *fonction valeur*.

Une classe très importante de la théorie de commande optimale est celle des jeux différentiels, liée également à la théorie des jeux. Dans ce contexte, l'évolution du système dynamique est affectée par les actions de plus d'un joueur impliqué dans le jeu et chaque joueur vise à améliorer son gain. Ainsi, les jeux différentiels constituent un cadre très commode pour étudier des problèmes caractérisés par des situations conflictuelles entre plusieurs joueurs ayant des intérêts différents. Une autre motivation pour les jeux différentiels consiste à étudier des problèmes de commande optimale où le système dynamique est affecté par des perturbations inconnues [126, 13, 66, 87].

Dans ce travail, nous nous intéressons aux jeux différentiels à somme nulle et à deux joueurs où le gain d'un joueur correspond certainement à une perte de son adversaire. Par conséquent, on peut définir pour chaque joueur sa propre fonction de valeur. D'autre part, il existe plusieurs cadres pour étudier les jeux différentiels en fonction des informations disponibles pour chaque joueur au cours du jeu. Un cadre intéressant consiste à restreindre les informations disponibles pour les deux joueurs au cours du jeu. Désormais, chaque joueur n'a aucune idée des choix futurs de son adversaire. Nous commençons par introduire au chapitre 2 quelques définitions générales et résultats de base pour les jeux différentiels à somme nulle et à deux joueurs. La principale contribution de ce chapitre est de proposer une procédure de reconstruction de stratégies et de contrôles optimaux pour les jeux différentiels à horizon fini. Le chapitre 3 est consacré à l'étude d'un jeu différentiel avec contraintes d'état et fonction de coût maximum où nous n'imposons aucune hypothèse de contrôlabilité et où les contrôles des deux joueurs peuvent être couplés dans la dynamique, les fonctions de coût et les contraintes d'état. En particulier, nous caractérisons la fonction valeur de ce problème à travers un jeu différentiel auxiliaire sans contraintes d'état. De plus, nous établissons un lien entre les stratégies optimales du problème contraint et celles du problème auxiliaire et nous présentons une approche générale permettant de construire des lois optimales approchées au jeu différentiel contraint pour les deux joueurs. Enfin, un problème d'atterrissage d'avion en présence de perturbations du vent est donné à titre d'exemple numérique illustratif.

L'approche de Programmation Dynamique est largement utilisée pour résoudre les problèmes de commande optimale en calculant une approximation de la fonction de valeur à travers plusieurs méthodes numériques telles que les méthodes *Différences finies* ([51]), *semi-lagrangien* ([61, 63]) et les méthodes *Fast Marching* ([122]). Un inconvénient de cette classe de méthodes numériques est la forte dépendance à la dimensionnalité de l'état puisque les calculs sont effectués sur une grille espace-temps. Pour cette raison, notre objectif dans le seconde partie de cette thèse était de développer des méthodes numériques

permettant de résoudre des problèmes de commande optimale avec une grande dimension d'état.

Dans cette thèse, nous exploitons des idées de l'intelligence artificielle et de l' *Optimisation Optimiste* de [106, 105, 39, 38] et nous proposons des algorithmes de *Planification Optimiste* pour résoudre des problèmes de commande optimale sous contraintes d'état en affinant l'ensemble des contrôles, au lieu de discrétiser l'espace d'état, ce qui rend cette approche très intéressante pour de nombreuses applications où la dimension de contrôle est très faible par rapport à la dimension d'état. En outre, nous établissons des résultats de convergence de ces algorithmes qui dépendent d'un budget de calcul donné. Finalement, nous étudions des méthodes numériques, basées sur l'apprentissage profond et la programmation dynamique, pour des problèmes de commande optimale déterministe avec contraintes d'état et pour des jeux différentiels à somme nulle et à deux joueurs. Une analyse numérique de ces méthodes est effectuée sur plusieurs exemples dans un espace d'état de grande dimension.

Chapter 1

Introduction

The purpose of this thesis is to study differential games under state constraints with the Hamilton-Jacobi approach and to develop numerical learning methods for solving state-constrained optimal control problems.

Strictly related with optimization, optimal control is a branch of applied mathematics that has been used in different engineering areas such as aerospace, energy, chemistry, economy

The main aim of optimal control theory is to affect the evolution of a dynamical system in time in order to minimize a cost, maximize a gain, reach a final target or to stabilize the system. The time evolution of the system defines a trajectory governed by means of ordinary differential equations for deterministic optimal control problems and by stochastic differential equations for stochastic problems. The system trajectory is directly affected by the controller actions called the admissible control inputs if satisfying some properties. On the other hand, in many applications, the state space can be restricted which defines some constraints that should be respected by the trajectory during the system evolution.

A significant interest has been accorded to deterministic optimal control since 1950's. The first motivation was for aerospace applications. Later, stochastic optimal control has appeared in 1970's to study problems in finance with a pioneer work for the portfolio optimization [102].

Among the main approaches to study optimal control problems, one can find in the literature the Pontryagin Maximum Principle that was introduced in 1956 by Lev Semenovich Pontryagin [28]. This approach consists in formulating some necessary optimality conditions for the control law. In this work, we will focus on another approach which is the *Dynamic Programming* (DP) approach formulated by Richard Bellman in 1950's, see [18, 19].

The DP approach considers the optimal value of the optimization problem as a function of the initial condition which defines the *value function*. This value function satisfies a specific equation called the *Dynamic Programming Principle* (DPP) which decomposes the optimal control problem into simpler subproblems before solving it in a recursive way. From the DPP and if the value function is smooth enough, it becomes the solution of a particular nonlinear partial differential equation called the *Hamilton-Jacobi-Bellman* (HJB) equation. Nevertheless, even for simple problems, one cannot guarantee the regularity of the value function which does not allow to characterize it as the solution of an HJB equation in a classical sense. For this reason, several theories has been appeared to define non-classical notions for solutions of *Hamilton-Jacobi* (HJ) equations. In this context, M.G. Crandall and P.L. Lions introduced, in the early 80's, a weak sense for HJ solutions, called *viscosity solutions*, which presents a very suitable framework to study existence, uniqueness and stability for a wide class of nonlinear *Partial Derivatives Equations* (PDE's) that includes HJ equations, see for instance [83, 84, 86].

On the other hand, many applications can be modeled by optimal control problems while taking into account some constraints on the system state which adds some difficulties. In particular, the value function

may become discontinuous and one cannot guarantee its uniqueness as viscosity solution to the corresponding HJ equation unless some controllability assumptions are satisfied. Among the most popular controllability assumptions, one can find the *inward pointing* condition, introduced by Soner in [124, 125], which states that, at each point of the constraints set boundary, there exists a control variable allowing the dynamical system to point in the interior of this set. Another controllability assumption, the *outward pointing* condition, was formulated in [68, 70]. This assumption imposes that each point belonging to the constraints set boundary can be hit by a trajectory coming from the interior of this set. Unfortunately, such assumptions cannot be verified in many cases. Our work here is based on an alternative technique, introduced in [5], that characterizes the constrained problem through an auxiliary optimal control problem free of state constraints.

A very important class of optimal control theory is differential games, related also with game theory. In this context, the evolution of the dynamical system is affected by the actions of more than one player involved in the game and each player aims to ameliorate his payoff. Henceforth, differential games constitute a very convenient framework to study problems characterized by conflict situations between several players having different interests. Differential games theory appeared in 1960's with a competition between the U.S.A., with a pioneer work for Isaacs [81], and the Soviet Union, represented by the Pontryagin school [109]. At that time, the aim was to study military applications and essentially pursuit evasion games, see [46, 12, 13]. Another motivation for differential games consists in investigating optimal control problems where the dynamical system is affected by some unknown disturbances [126, 13, 66, 87]. This situation can be modeled by a game where a real controller tries to counteract to the worst possible behaviors of the disturbances.

In this work, we are interested in two-person zero-sum differential games where the gain of one player corresponds certainly to a loss of his opponent. Therefore, one can define for each player its own value function. On the other hand, there are several frameworks to study differential games depending on the available information for each player during the course of the game. The *static* game corresponds to the case where each player has a complete information about his opponent future choices. In this context, one cannot guarantee the existence of a value for the game, i.e. equality between the two players value functions. Moreover, the dynamic programming approach cannot be applied here to characterize and compute the different value functions. A more interesting framework consists in restricting the available information for both players during the course of the game. Henceforth, each player has no idea about his opponent future choices. Nevertheless, an advantage of information can be accorded to only one of the two players by knowing the past and the current choices of his opponent which defines non-anticipative strategies introduced in [59, 60, 119, 129, 56]. A more restrictive class of non-anticipative strategies is delay strategies, see [45, 56]. Games can be studied in another different context, feedback strategies [49, 126, 16, 58], where one player knows the current state of the system and keep track of its past history. As for stochastic differential games, there are other information patterns in the literature such as random strategies, see [43, 44, 9]. Those different information contexts provide a convenient framework to study differential games with the dynamic programming approach and hence to characterize value functions as unique viscosity solutions to the appropriate HJ equations. Furthermore, one can find suitable conditions under which a value of the game exists. The most popular one is the Isaacs' condition, see [56]. We refer also to [67, 14] where two-person zero-sum stochastic differential games have been investigated in the viscoity solutions framework and to [32, 89] for the general theory of more than two players, the $N-$players game.

The DP approach is widely used to solve optimal control problems by computing an approximation of the value function known as the unique viscosity solution of the corresponding HJ equation. The advantage of such method is to allow to synthesize approximated optimal controls in feedback form which gives sub-optimal solutions. Moreover, several numerical methods have been proposed to approximate the solutions of first order HJ equations, derived from deterministic optimal control problems, such as *Finite*

Differences ([51]), *semi-Lagrangian* ([61, 63]) schemes and *Fast Marching* methods ([122]). For second order HJ equations, arisen in the stochastic case, one can cite *Markov chain* approximations [92] and we refer also to [14, 91]. One drawback of this class of numerical methods is the strong dependence on the state dimensionality since computations are done on a time-space grid. For this reason, solving problems with a dimension grater than 4 or 5 becomes very complex in time and requires huge memory capacities (*curse of dimensionality*).

In order to deal with the curse of dimensionality, several alternative numerical methods have been proposed in the literature. The problem can be solved by considering a simplified form which can be obtained for instance by ignoring some uncertainties in the stochastic case or by reducing the size of the state space and hence the DP approach will be applied only on a reduced subset of states and the solution of the problem will be extended by interpolation to the whole state space, see [21]. Another form of simplification consists in domain decomposition techniques for partial differential equations [110]. In [65], an approximated scheme was proposed to solve Hamilton-Jacobi equations by splitting the original problem into simpler problems on two sub-domains with a linking condition and by imposing constraints on the system state. Recently, a state-tree-structure method has appeared in order to approximate the solution of a dynamic programming equation, see [3, 120]. This approach eliminates the space discretization and constructs a tree, starting from a given initial state, by adding only the states that will be encountered by a discrete time dynamics and a finite number of controls. Then, the value function will be computed by the dynamic programming principle on the constructed tree.

Another class of methods allowing to solve the curse of dimensionality is *On-line approaches* where the optimization is done only for the current states that will be encountered during the control process. Among those methods, one can find *Rollout* algorithms using heuristic ideas, see [21], and the *Model Predictive Control* approach, see [94, 21].

Furthermore, *Neural Network Training* has been used to approximate the value function, see [21, 22] for an overview of such approach. In this context, a neural network is a parametric function depending on the system state and involving some free parameters that will be chosen in such a way to fit the value function, representing the *Target* function. This operation, called the training of the neural network, uses a set of state-value pairs known as the training set. The theoretical justification of this approach comes from the Kolmogorov-Arnold representation theorem and the universal approximation theorem, see [98, 52, 72, 99].

In this thesis we exploit ideas from artificial intelligence and *Optimistic Optimization* from [106, 105, 39, 38] and we propose *Optimistic Planning* algorithms to solve state-constrained optimal control problems by refining the set of controls, instead of discretizing the state space, which makes this approach very interesting for many applications where the control dimension is very low compared to the state dimension. Moreover, we propose a deep learning algorithm exploiting the DP approach to solve optimal control problems. The latter will be compared to another approach that tries to approximate the solutions of Hamilton-Jacobi equations.

Recall that the objective of this thesis is to apply the Hamilton-Jacobi approach to investigate state-constrained differential games and to develop learning numerical methods to solve high-dimensional optimal control problems with state constraints. First, we start by introducing in chapter 2 some general definitions and basic results for two-person zero-sum differential games. The main contribution of this chapter is to present a reconstruction procedure of optimal strategies and controls for finite-horizon differential games. Chapter 3 is devoted to study a differential game with state constraints and maximum cost function. An auxiliary differential game, free of state constraints, is introduced in order to characterize the original problem and to approximate its optimal strategies and controls. Here, controls of the two players are allowed to be coupled within the dynamics, the state constraints and the cost functions. Then in chapter 4, we develop optimistic planning algorithms to solve state-constrained optimal control problems. Moreover, we provide theoretical convergence results of our proposed algorithms. The relevance

of our approach will be illustrated even for high-dimensional problems. Finally, in chapter 5 we propose two different numerical approaches based on deep learning to solve optimal control problems under state constraints and we compare their performances.

1.1 Differential games

In chapter 2, we start by studying two-person zero-sum differential games in finite time horizon with cost functional of type *Bolza* in the context of nonanticipative strategies with delay, based on notes of P.Cardaliaguet in [45]. Then, we focus on the framework of nonanticipative strategies. In particular, some regularity properties are verified by the two players value functions. Moreover, each value function verifies a dynamic programming equation which implies its characterization as the unique viscosity solution of an appropriate Hamilton-Jacobi-Isaacs equation. Furthermore, we state a comparison result involving the different value functions defined in the context of delay and nonanticpative strategies. Finally, as we said before, the main contribution of this chapter is the introduction of a reconstruction procedure of optimal strategies and controls by means of a discrete time game that converges to the continuous time problem when the time step goes to zero, see subsection 2.6. This reconstruction procedure will be extended in chapter 3 to deal with maximum cost functions.

In chapter 3, we consider the following state-constrained differential game:

$$v(t,x) := \inf_{\alpha[\cdot]\in\Gamma} \pi(t,x;\alpha) \tag{1.1}$$

with the convention that $\inf \varnothing = +\infty$, $\alpha[\cdot] \in \Gamma$ is a nonanticipative strategy of the first player and where π is defined by:

$$\pi(t,x;\alpha) := \begin{cases} \sup\limits_{b(\cdot)\in\mathcal{B}} \left\{ \left(\max\limits_{s\in[t,T]} \phi(y_{t,x}^{\alpha[b],b}(s)) \right) \bigvee \psi(y_{t,x}^{\alpha[b],b}(T)) \right\}, \text{ if } y_{t,x}^{\alpha[b],b}(s) \in \mathcal{K},\ \forall s \in [t,T],\ \forall b(\cdot) \in \mathcal{B}, \\ +\infty, \text{ else,} \end{cases}$$

for $(t,x) \in [0,T] \times \mathbb{R}^d$ with $T > 0$ is the final time, $b(\cdot) \in \mathcal{B}$ and $\alpha[b](\cdot) \in \mathcal{A}$ are the actions of the second and the first players respectively, $\mathcal{K}$ is a closed set of $\mathbb{R}^d$ representing the set of constraints and $y_{t,x}^{\alpha[b],b}(\cdot)$ is the unique absolutely continuous solution of the following dynamical system:

$$\begin{cases} \dot{y}(s) = f(s, y(s), \alpha[b](s), b(s)), \quad a.e.\ \ s \in [t,T], \\ y(t) = x. \end{cases} \tag{1.2}$$

The cost functions $\phi : \mathbb{R}^d \to \mathbb{R}$ and $\psi : \mathbb{R}^d \to \mathbb{R}$ and the dynamics $f{:}[0,T]\times\mathbb{R}^d\times A\times B \to \mathbb{R}^d$ are continuous functions with A and B are two compact subsets of $\mathbb{R}^p$ and $\mathbb{R}^q$ respectively, $p, q \geq 1$, in which the players actions take values. Moreover, $y_{t,x}^{\alpha[b],b}(\cdot)$ which represents the trajectory of the system corresponding to $(\alpha[b](\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$, is said to be admissible if it remains in the admissible set $\mathcal{K}$. See section 3.2 for further definitions and more precise assumptions.

This problem formulation describes the situation where the first player is exploiting his information advantage and trying to find nonanticipative strategies that guarantee the admissibility of trajectories against any choice of the second player and minimize the cost functional π. This can model the case where a controller tries to counteract to unknown disturbances which can affect the system and the cost functions. One can imagine another game example where the second player objective is to maximize the cost π or to violate the state constraints.

This problem was studied in a particular case, $\mathcal{K} \equiv \mathbb{R}^d$, in [121, 15, 116]. It was also considered in the case of a single-controller in [111] through characterizing the value function epigraph by use of a *viability*

kernel.

In the general case, $\mathcal{K} \neq \mathbb{R}^d$, some difficulties appear. The value function v may become discontinuous and its uniqueness as a viscosity solution of an HJ equation requires some additional assumptions involving the dynamics f and the constraints set $\mathcal{K}$. In the case of single-controller problems, the most popular assumption, the Inward Pointing Condition, states that at each point of the boundary of $\mathcal{K}$, there exists a control value that lets the dynamics point in the interior of $\mathcal{K}$. We refer to [34, 104, 124] for this assumption and to [103, 116, 121] for weaker inward pointing assumptions. Equivalent assumptions for the case of a two-person game was also given in [46, 23, 24]. Such assumptions cannot be guaranteed for several control problems. For this reason, we follow the level set approach, introduced in [5] for single-controller optimal control problems, and we show that, even for state-constrained differential games, the value function v may be characterized by means of a locally Lipschitz continuous value function of an auxiliary differential game free of state constraints. Moreover, the auxiliary value function is the unique viscosity solution of an HJ equation with an obstacle term. Another contribution of this chapter is that controls of the two players are allowed to be coupled within the dynamics, the state constraints and the cost functions. Moreover, here we consider weaker assumptions on f, ϕ and ψ, compared to [5].

First, since $\mathcal{K}$ is a closed set it can be characterized via the signed distance $d_{\mathcal{K}}(\cdot)$, which is Lipschitz continuous, in the following form:

$$\forall y \in \mathbb{R}^d, \quad d_{\mathcal{K}}(y) \leq 0 \Leftrightarrow y \in \mathcal{K}.$$

The value function of the auxiliary problem is defined, for $t \in [0,T]$ and $(x,z) \in \mathbb{R}^d \times \mathbb{R}$, by:

$$w(t,x,z) := \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \left\{ \left(\max_{s\in[t,T]} \hat{\phi}(y_{t,x}^{\alpha[b],b}(s), z) \right) \bigvee \hat{\psi}(y_{t,x}^{\alpha[b],b}(T), z) \right\} \tag{1.3}$$

where for $(y,z) \in \mathbb{R}^d \times \mathbb{R}$, the functions $\hat{\phi}$ and $\hat{\psi}$ are given by:

$$\hat{\phi}(y,z) := (\phi(y) - z) \bigvee d_{\mathcal{K}}(y) \quad \text{and} \quad \hat{\psi}(y,z) := \psi(y) - z.$$

Moreover, the value function w is the unique viscosity solution of the following HJ equation:

$$\begin{cases} \min\left(-\partial_t w(t,x,z) + H(t,x,D_x w(t,x,z)), w(t,x,z) - \hat{\phi}(x,z) \right) = 0, & \text{in } [0,T[\times\mathbb{R}^d \times \mathbb{R}, \\ w(T,x,z) = \hat{\phi}(x,z) \bigvee \hat{\psi}(x,z), & \text{in } \mathbb{R}^d \times \mathbb{R}, \end{cases} \tag{1.4}$$

where the Hamiltonian H is given by:

$$H(t,x,p) := \min_{b\in B} \max_{a\in A} -\langle f(t,x,a,b), p\rangle, \quad \text{for} \quad (t,x,p) \in [0,T] \times \mathbb{R}^d \times \mathbb{R}^d.$$

Now, by exploiting the level sets of the auxiliary value function w and when some convexity assumption is verified by f, the value function v can be determined by the following relation:

$$v(t,x) = \inf \left\{ z \in \mathbb{R} \;\mid\; w(t,x,z) \leq 0 \right\}.$$

Moreover, we prove that an optimal strategy of the auxiliary problem (1.3), associated to a particular initial condition, remains also optimal for the constrained problem (1.1). Another contribution of this chapter is to present a reconstruction procedure to approximate the optimal strategies and controls of both players. Indeed, we exploit some ideas from [6, 118] about trajectory reconstruction and we propose algorithms to reconstruct approximated optimal trajectories for the continuous time auxiliary differential game by use of a discrete time game.

As an illustrative example, we study an aircraft landing problem in the presence of windshear. Indeed,

the best strategy to avoid a failed landing, that can occurs because of quick changes of the wind velocity, is to steer the aircraft to the maximal altitude that can be reached, during an interval of time, in order to prevent a crash on the ground. In [103, 104], a Chebyshev-type optimal control problem was proposed and an approximated solution is provided. The Hamilton-Jacobi-Bellman approach was applied in [6] to solve this problem after supposing the knowledge of the wind velocity fields. In [29], the aircraft landing problem was formulated as a nonlinear differential game with state constraints and a semi-Lagrangian scheme was applied to compute an approximation of the value function.

In this work, we propose a 5D differential game model with maximum running cost, where wind disturbances are considered as a second player and the first player tries, by use of nonanticipative strategies, to counteract to some catastrophic scenarios that can occur because of wind disturbances.

1.2 Numerical learning methods for optimal control problems

In chapters 4 and 5, we exploit ideas from reinforcement and deep learning and we propose two different approaches to solve state-constrained optimal control problems. The common aim of those numerical methods is to be able to solve problems with high state dimension.

For a fixed time horizon $T > 0$, a non empty and closed set $\mathcal{K} \subset \mathbb{R}^d$, $d \geq 1$, representing the state constraints set, and a compact subset A of $\mathbb{R}^q$, $q \geq 1$, consider the following constrained optimal control problem:

$$v(t,x) := \inf_{a(\cdot)\in\mathcal{A}} \Big\{ \int_t^T \ell(y^a_{t,x}(s), a(s))ds + \Phi(y^a_{t,x}(T)) \mid y^a_{t,x}(s) \in \mathcal{K}, \ \ \forall s \in [t,T] \Big\}. \tag{1.5}$$

for $(t,x) \in [0,T] \times \mathbb{R}^d$, where $\mathcal{A}$ is the set of measurable function $a(\cdot) : [0,T] \to A$ and $y^a_{t,x}(\cdot)$ is the continuous solution of the following dynamical system

$$\begin{cases} \dot{y}(s) = f(y(s), a(s)) \ \ a.e. \ \ s \in [t,T], \\ y(t) = x. \end{cases} \tag{1.6}$$

The functions f, ℓ and Φ are continuous, see chapters 4 and 5 for the convenient assumptions.

It is known that when the problem is free of state constraints, $\mathcal{K} = \mathbb{R}^d$, the value function v is Lipschitz continuous and can be characterized as the unique viscosity solution of the following HJ equation:

$$\begin{cases} -\partial_t v(t,x) + \max_{a\in A} \Big\{ -\langle f(x,a), D_x v(t,x)\rangle - \ell(x,a) \Big\} = 0 \quad \text{on } [0,T[\times\mathbb{R}^d, \\ v(T,x) = \Phi(x) \quad \text{on } \mathbb{R}^d. \end{cases}$$

This characterization provides a convenient framework for computing approximations of the value function v that allow to synthesize approximated optimal controls in feedback form and hence to obtain sub-optimal solutions. Nevertheless, the most popular numerical schemes for solving HJ equations (semi-Lagrangian, finite differences...) suffer from the curse of dimensionality since it approximate the solution on a grid (time and space) which reduces the ability of solving problems in high state dimension. Another difficulty is added in presence of state constraints, $\mathcal{K} \neq \mathbb{R}^d$. In fact, v may become discontinuous and the above characterization (via HJ equations) is no longer valid, unless some controllability assumptions are satisfied. For this reason, we follow again the level set approach, introduced in [5], which consists in characterizing the constrained problem through an auxiliary optimal control problem free of state constraints whose value function is Lipschitz continuous and can be characterized as the unique viscosity solution of an HJ equation.
First, since the set of constraints $\mathcal{K}$ is a closed subset of $\mathbb{R}^d$, there exists a Lipschitz continuous function g characterizing $\mathcal{K}$ in the following form:

$$\forall y \in \mathbb{R}^d, \quad g(y) \leq 0 \iff y \in \mathcal{K}.$$

The value function of the auxiliary control problem associated to the constrained problem (1.5) is defined, for $(t,x,z) \in [0,T] \times \mathbb{R}^d \times \mathbb{R}$, by:

$$w(t,x,z) := \inf_{a(\cdot)\in\mathcal{A}} \Big\{ \Big(\int_t^T \ell(y^a_{t,x}(s), a(s))ds + \Phi(y^a_{t,x}(T)) - z \Big) \bigvee \Big(\max_{s\in[t,T]} g(y^a_{t,x}(s)) \Big) \Big\}. \tag{1.7}$$

From [5], we already know that, under a convexity assumption on f, v can be characterized as follows:

$$v(t,x) = \inf\{z \in \mathbb{R} \mid w(t,x,z) \leq 0\}.$$

Moreover, an optimal solution of problem (1.7), for a particular value of the auxiliary variable z, coincides with an optimal solution for problem (1.5).

Exploiting this characterization, we shall compute an approximation of the auxiliary value function. Although w can be characterized through HJ equations, its approximation by solving numerically the corresponding HJ equation may seem unreasonable for problems with high state dimension. Indeed, the state vector of the auxiliary problem is increased by one more variable (the auxiliary variable z), compared to the state of the original problem, and it is known that classical numerical methods for solving HJ equations are grid-dependent which makes those approaches applicable only for problems where the dimension of the state variable is low. For this reason, we are looking for numerical methods to approximate w while avoiding or reducing the direct dependence between the state dimensionality and the resolution complexity.

For sake of simplicity, the initial time t is set to $t = 0$. Consider a uniform partition of $[0,T]$, $s_0 = 0, ..., s_k = kh, ..., s_N = T$, $N \in \mathbb{N}^*$, with time steps size $h := \frac{T}{N}$. Moreover, let ρ_h be an instantaneous cost that approximates the integral of ℓ over a time sub-interval $[s_k, s_{k+1}]$ and $(y^a_k)_k$ be the discrete trajectory associated to (1.6) and corresponding to a discrete control sequence $(a_k)_k \in A^N$.

1.2.1 Optimistic planning approach

In chapter 4, we propose optimistic planning algorithms that refine the set of controls instead of discretizing the state space. This approach is very interesting especially for many applications where the control dimension is very low compared to the state dimension.

The discrete auxiliary control problem, free of state constraints, is defined as follows:

$$W(x,z) := \inf_{(a_k)_k \in A^N} J(x,z,a),$$

where the cost functional J is given by:

$$J(x,z,a) = \Big(\sum_{k=0}^{N-1} \rho_h(y^a_k, a_k) + \Phi(y^a_N) - z \Big) \bigvee \Big(\max_{0\leq k\leq N} g(y^a_k) \Big).$$

It is worth to mention that $W(x,z)$ converges to $w(0,x,z)$, over compact subsets of $\mathbb{R}^d \times \mathbb{R}$, as $N \to +\infty$ (i.e. $h \to 0$), see [11, 64].

Optimistic planning algorithms are based on the principles of optimistic optimisation (see [56]). This approach requires the Lipschitz continuity of J with respect to the control sequence. In order to approximate $W(x,z)$, we will refine iteratively the search space, A^N, in an *optimistic* way into smaller subsets and for each subset, we attribute a control sequence and hence a value of the objective function J.

Our first two algorithms are an adaptation of the algorithms presented in [39, 38, 35, 75, 76] to our case in finite time horizon and with maximum cost. The first method, *Optimistic Planning*, refines optimistically the subsets minimizing a lower bound on the auxiliary value function $W(x,z)$. However, the *Simultaneous*

Optimistic Planning algorithm discovers simultaneously several subsets of the search space characterized by minimal values of the objective function J. Furthermore, we prove convergence results for those algorithms in a similar way to [39].

Finally, we propose a third algorithm, *Simultaneous Optimistic Planning with Multiple Steps*, which is designed by combining both the optimistic planning approach with ideas from the MPC (Model Predictive Control). Indeed, at each time step $k = 0, ..., N-1$, this algorithm picks the first value of the near optimal control sequence, obtained by minimization of the objective function over control sequences of length $N-k$, in order to simulate the next state system. This procedure ameliorates the precision of the algorithm and reduces significantly the computational time compared to the previous methods.

In order to show the relevance of our proposed approach, we illustrate with several numerical applications. First, we consider the Zermelo problem where a boat tries to reach a circular target at the final time T with minimal fuel consumption. We consider also two rectangular obstacles in the navigation domain that the boat should avoid. Then, we study the optimal control of the heat equation in order to show the relevance of our approach in higher dimensions, the dimension here is $d = 10^3$. Our aim is to minimize, by using a control input, the temperature in a given domain which is the solution of a partial derivatives equation. Furthermore, we add some constraints to this example where we impose that the solution should remain above the initial solution multiplied by some non-negative parameter. Finally, we consider an abort landing problem where the aim is to steer an aircraft to the maximum altitude that can be reached during an interval of time in the presence of the wind velocities for which we assume having an explicit model, see [6, 33, 34].

On the other hand, we give in chapter 6 some ideas about how to extend the optimistic planning approach in order to deal with two-person zero-sum differential games under constraints on the system state.

1.2.2 Deep learning numerical methods for dynamic programming

We propose two deep learning approaches to approximate the auxiliary value function w. The first one is based on the dynamic programming principle while the second method tries to approximate the solutions of HJ equations.

Deep neural networks have shown to be relevant in approximating a large class of complex non linear functions on finite dimensional space. This relevance can be theoretically justified by the Kolmogorov-Arnold representation theorem and the universal approximation theorem, see [98, 52, 72, 99].

It is known that the value function of an optimal control problem, under suitable assumptions, is the solution of a dynamic programming equation. In this context, one can discretize in time and then try to approximate the value function, at each time step, by neural networks after its learning on a training grid with reduced size, see [21]. For instance, the *Hyprid-Now* Algorithm, introduced in [78, 10, 77], estimates first the optimal policy by neural networks then this estimated policy is injected in a backward process to approximate the discrete value function. This approach is very interesting especially when the optimal policy is regular. In this chapter, we propose to adapt this algorithm to deterministic control problems for which the optimal control is not always regular enough. To this end, we will try to approximate only the value function by using neural networks and by exploiting the dynamic programming principle. Moreover, we extend this approach to deal with constraints on the system state.

On the other hand, DNN have been successfully used to solve some nonlinear partial differential equations (PDE) derived from physics and mathematics, see [115, 114, 113, 112, 71, 128]. Indeed, the solution of the PDE can be directly approximated by neural networks that will be learned, on a reduced training domain, in order to satisfy the boundary conditions and the given equation law. In this context, we propose to approximate the value function, solution of some HJ equation, by use of spatio-temporal neural networks while computing its derivatives by means of automatic differentiation, see [17].

A first-order approximation of the auxiliary value function w at time s_k, for $k = 0, ..., N$, can be given by:

$$W_k(x,z) = \min_{(a_i)_i \in A^{N-k}} \Big\{ \Big(\sum_{i=k}^{N-1} \rho_h(y_i^a, a_i) + \Phi(y_N^a) - z \Big) \bigvee \Big(\max_{k \le i \le N} g(y_i^a) \Big) \Big\}.$$

In fact, $(W_k)_{k=0}^N$ is the unique solution of the following discrete dynamic programming equation:

$$\begin{cases} W_N(x,z) = (\Phi(x) - z) \bigvee g(x), \\ W_k(x,z) = \min\limits_{a \in A} \Big\{ W_{k+1}(\hat{F}_h(\hat{x}, a)) \bigvee g(x) \Big\}, \quad \text{for } \ k = N-1, ..., 1, 0, \end{cases}$$

where $\hat{F}_h$ is a discrete approximation of the augmented dynamics $\hat{f}(x,a) := \begin{pmatrix} f(x,a) \\ -\ell(x,a) \end{pmatrix}$.

By backward induction and by using $\widehat{W}_{k+1}$, an approximation of W_{k+1} for $k = 0, ..., N-1$, we first compute $\widehat{W}_k$ on a generated training grid. Then, the latter approximation, computed on a reduced domain, will be used as an input data to extend $\widehat{W}_k$, by use of neural networks and stochastic optimization, to the whole computational domain which defines an approximation of W_k. This approach will be compared to another one that consists in approximating w, at any time instant $t \in [0,T]$, by training neural networks, on a reduced domain, in order to satisfy the following HJ equation whose unique solution is w:

$$\begin{cases} \min \Big(-\partial_t w(t,\hat{x}) + H(x, D_{\hat{x}} w(t,\hat{x})), w(t,\hat{x}) - g(x) \Big) = 0, \quad \text{on } [0,T[\times \mathbb{R}^{d+1}, \\ w(T,\hat{x}) = (\Phi(x) - z) \bigvee g(x), \quad \text{on } \mathbb{R}^{d+1}, \end{cases}$$

for $\hat{x} := (x,z) \in \mathbb{R}^d \times \mathbb{R}$ and where H is the Hamiltonian function given by:

$$H(x,p) = \max_{a \in A} \Big\{ -\langle \hat{f}(x,a), p \rangle \Big\}, \ \ \forall (x,p) \in \mathbb{R}^d \times \mathbb{R}^{d+1}.$$

Furthermore, both approaches can be extended to handle state-constrained two-person zero-sum differential games.

Chapter 2

Unconstrained Finite Horizon Differential Games

2.1 Introduction

This chapter is devoted to recall some definitions about differential games theory and basic results concerning the application of the Hamilton-Jacobi approach to solve differential games.

Differential games theory can be seen as an intersection of game theory, where more than one player are involved, and of optimal control theory, as each player looks for the best possible decisions to influence the evolution of a dynamical system in such a way to ameliorate his payoff. On the other hand, differential games are considered as a convenient framework to investigate conflict situations for dynamical systems controlled by several agents while having different interests.

The first main motivation of differential games is to model conflict situations between two players having opposite interests. The most popular application concerns the study of pursuit-evasion games, see [46, 12, 13]. Another very useful motivation is the analysis of a controlled system with some unknown disturbances [126, 13, 66, 87]. It is known that the most widely used approach for solving such problems is to establish a statistical model for disturbances and to optimize the expected value of the objective function. Nevertheless, optimizing the expected value does not guarantee a good performance of the system against some dangerous behaviours of the disturbances. Moreover, it is not always possible to find an efficient statistical model for disturbances. Henceforth, this problem can be modeled by a game where a real controller, representing the first player, tries to counteract to the worst possible actions of the disturbances, considered as a second player of the game.

In this chapter, we are interested in two-person zero-sum differential games without state constraints, with finite time horizon and where the objective function is of type Bolza. The first player, by choosing a control input $a(\cdot) \in \mathcal{A}$, tries to minimize an objective function $J(t, x, a, b)$, where x is the initial position of the system at the first time instant $t \in [0, T]$ for $T > 0$ and $b(\cdot) \in \mathcal{B}$ is the second player decision. $\mathcal{A}$ and $\mathcal{B}$ denote respectively the actions sets of the first and the second players. Conversely, $-J(t, x, a, b)$ corresponds to the cost that the second player should pay. In other words, the loss of one player coincides with the gain of his opponent.

According to the classical game theory, both players should optimize over $\mathcal{A}$ and $\mathcal{B}$ which defines a *static* game with a lower and an upper value functions defined by:

$$v_s^-(t,x) := \sup_{b(\cdot)\in\mathcal{B}} \inf_{a(\cdot)\in\mathcal{A}} J(t,x,a,b) \leq v_s^+(t,x) := \inf_{a(\cdot)\in\mathcal{A}} \sup_{b(\cdot)\in\mathcal{B}} J(t,x,a,b)$$

It is worth mentioning that the lower value function v_s^- describes the case where the second player chooses his action $b(\cdot) \in \mathcal{B}$ based on the knowledge of the first player future decisions. Conversely, from the def-

inition of v_s^+, the first player has a complete information about his opponent future choice. On the other hand, the dynamic programming approach cannot be applied to compute the value functions v_s^- and v_s^+. Moreover, one cannot guarantee the existence of a value for the game in this context, i.e. equality between the lower and the upper value functions.

A more interesting setting for differential games is to restrict the available information for both players during the course of the game which can be modeled by the notion of game strategies where each player has no idea about his opponent future choices. This information pattern allow us to investigate differential games by means of the dynamic programming approach and hence the Hamilton-Jacobi approach. Moreover, under some suitable assumptions, one can prove the existence of a value for the game. Among the most popular strategies in the literature, we cite nonanticipative strategies, introduced in [59, 60, 119, 129, 56], where only one of the two players knows the past and current choices of his opponent without any idea about his future decisions. A particular type of non-anticipative strategies is delay strategies, see [45, 56]. Some practical examples cannot be described by the notion of non-anticipative strategies. To this end, the game can be studied in another context: feedback strategies. In this case, one player knows the current state of the system and keep track of its past history, see [49, 126, 16, 58]. Another example of information pattern is the random strategies where each player has a part of the information concerning the objective function to optimize. The latter class of strategies was introduced to deal with differential games with incomplete information, see [43, 44, 9].

In this chapter, we will start by giving some basic results established for differential games in the context of delay strategies. Then, we will move to a more general information pattern which is nonanticipative strategies.

The main question now is how to characterize the players' value functions of a differential game. In 1950's, Isaacs observed, for the first time, that if the value function of a differential game is sufficiently regular, then it is solution to some non-linear first order PDE called the Hamilton-Jacobi-Isaacs (HJI) equation, see [82]. However, for many examples, the value function is not smooth enough to be a PDE solution in a classical sense. Later in 1980's, M. G. Crandall and P.L. Lions have introduced the *theory of viscosity* which gives a weaker sense for a solution of an Hamilton-Jacobi equation [50]. Not only that but they have also proved, under suitable conditions, the uniqueness of such solutions. Then, it was shown in [60], that a differential game value function, defined via nonanticipative strategies, is the unique viscosity solution of an appropriate Hamilton-Jacobi-Isaacs equation. An adaptation of those results for the infinite horizon can be found for instance in [11, Chapter VIII]. On the other hand, the fundamental tool to characterize a value function as a viscosity solution of an HJ equation is the dynamic programming approach introduced in [18, 20, 19].

The main contribution of this chapter is to extend the results concerning the approximation of optimal strategies and controls for both players, presented in [11, Chapter VIII] for infinite horizon problems, to our case with finite time horizon. Indeed, we construct a specific approximation of the value function verifying a discrete dynamic programming principle and corresponding to the value function of some discrete time game. Then, we show the existence of optimal strategies and controls for this discrete time game that will be used later to compute approximated optimal strategies and control for the continuous time game.

This chapter is organized as follows. After presenting in section 2.2 the main definitions and hypothesis that will be used along this chapter, we study in section 2.3 differential games in the context of nonanticpative strategies with delay. First, the different value functions are defined and some regularity proprieties satisfied by those functions are presented. Then, we show how to characterize them as semi-viscosity solutions of the appropriate Hamilton-Jacobi-Isaacs equations. In addition to that, we introduce a comparison principle result to compare sub and super-solutions of Hamilton-Jacobi equations in the viscosity sense. In section 2.4, we consider differential games in another context: nonanticpative strategies. First, we define the value functions corresponding to the finite time horizon. Furthermore, we characterize those value functions as the unique viscosity solutions of the appropriate Hamilton-Jacobi-Isaacs equations. In

addition to that, we present a feedback reconstruction procedure based on the knowledge of an approximation of the value function. Finally, we propose a game example for which one can compute optimal strategies and controls and get explicit expressions of both value functions. In order to conclude this chapter, section 2.5 gives a comparison result between all the value functions introduced in sections 2.3 and 2.4.

2.2 Definitions and hypothesis

Let $T > 0$ be the finite time horizon and A and B be two compacts sets of $\mathbb{R}^p$ and $\mathbb{R}^q$ $(p, q \geq 1)$ in which actions of the first and the second players take values respectively. The set of admissible control functions of the first player, $\mathcal{A}$, can be defined as follows:

$$\mathcal{A} := \{a(\cdot) : [0, T] \to A, \text{ measurable}\}.$$

In a similar way, the set of admissible controls of the second player is given by:

$$\mathcal{B} := \{b(\cdot) : [0, T] \to B, \text{ measurable}\}.$$

In the literature, there exist different notions of game strategies. One can mention nonanticipative and delay strategies [59, 60, 119, 129, 56, 45], feedback strategies [49, 126, 16, 58] for deterministic differential games and random strategies for differential games with incomplete information, see [43, 44, 9]. In this chapter, we will focus on delay and nonanticipative strategies.

Definition 2.2.1 (Nonanticipative strategies)**.** *A nonanticipative (or casual) strategy for the first player is a map $\alpha[\cdot] : \mathcal{B} \to \mathcal{A}$, s.t. for any $t \leq T$ and $b(\cdot), b'(\cdot) \in \mathcal{B}$, if $b(s) = b'(s)$ for almost every $s \leq t$, then $\alpha[b](\cdot) = \alpha[b'](\cdot)$ almost everywhere in $[0, t]$.*

A nonanticipative strategy for the second player is defined in a symmetric way. We denote by Γ (resp. Δ) the set of nonanticipative strategies of the first player (resp. second player).

In other words, the player using nonanticipative strategies takes his control decision at each time instant with the knowledge of the past and current choices of his opponent and without any idea about his future decisions.

Now, we define delay strategies which constitute a restrictive type of nonanticipative strategies.

Definition 2.2.2 (Delay strategies)**.** *A map $\alpha[\cdot] : \mathcal{B} \to \mathcal{A}$ is a delay strategy (nonanticipative strategy with delay) for the first player if there is a delay $\tau_\alpha > 0$ such that for any two controls $b(\cdot), b'(\cdot) \in \mathcal{B}$, and any $t \geq 0$, if $b(\cdot) = b'(\cdot)$ almost everywhere in $[0, t]$, then $\alpha[b](\cdot) = \alpha[b'](\cdot)$ almost everywhere in $[0, (t + \tau_\alpha) \bigwedge T]$.*

A delay strategy for the second player is defined in a symmetric way. We denote by Γ_d (resp. Δ_d) the set of delay strategies for the first player (resp. second player).

Remark 2.2.3. *The last definition implies that if $\alpha[\cdot]$ is a delay strategy, $\alpha[b](\cdot)$ does not depend on $b(\cdot)$ on the interval $[0, \tau_\alpha \bigwedge T]$. Indeed, for any $b(\cdot), b'(\cdot) \in \mathcal{B}$, $b(\cdot) = b'(\cdot)$ almost everywhere at 0. Therefore, $\alpha[b](\cdot) = \alpha[b'](\cdot)$ almost everywhere in $[0, \tau_\alpha \bigwedge T]$.*

The following remark comes from the definitions of delay and nonanticipative strategies and will be useful later.

Remark 2.2.4. *Delay strategies are nonanticipative strategies $i.e.$ $\Gamma_d \subset \Gamma$ and $\Delta_d \subset \Delta$.*

In all the control problems that will be studied in this chapter, consider a dynamic f, a distributed cost ℓ and a final cost Ψ satisfying the following assumptions:

(H2.1) *The dynamics* $f : [0,T] \times \mathbb{R}^d \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^d$ *is a continuous function and there exits* $L_1 > 0$*, such that for any* $t, s \in [0,T]$*, for any* $x, y \in \mathbb{R}^d$ *and for any* $(a,b) \in A \times B$*:*

$$\|f(t,x,a,b) - f(s,y,a,b)\| \leq L_1(|t-s| + \|x-y\|).$$

(H2.2) *The distributed cost* $\ell : [0,T] \times \mathbb{R}^d \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ *is a continuous function and there exits* $L_2 > 0$*, such that for any* $t, s \in [0,T]$*, for any* $x, y \in \mathbb{R}^d$ *and for any* $(a,b) \in A \times B$*:*

$$|\ell(t,x,a,b) - \ell(s,y,a,b)| \leq L_2(|t-s| + \|x-y\|).$$

(H2.3) *The final cost* $\Psi : \mathbb{R}^d \to \mathbb{R}$ *is a Lipschitz continuous function with Lipschitz constant* $L_3 > 0$*.*

Consider the following nonlinear dynamical system:

$$\begin{cases} \dot{y}(s) = f(s, y(s), a(s), b(s)), \quad a.e. \quad s \in [t,T], \\ y(t) = x, \end{cases} \tag{2.1}$$

where $x \in \mathbb{R}^d$ is the initial system state and $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$ are the actions of the first and the second players respectively. The corresponding absolutely continuous solution of (2.1) is denoted by $y_{t,x}^{a,b}(\cdot)$ and represents the system trajectory.

Finally, let J be the cost functional, of type Bolza, that the first player wants to minimize and the second player wants to maximize:

$$J(t,x,a,b) = \int_t^T \ell(s, y_{t,x}^{a,b}(s), a(s), b(s))ds + \Psi(y_{t,x}^{a,b}(T)) \tag{2.2}$$

for $(t,x) \in [0,T] \times \mathbb{R}^d$ and $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$.

2.3 Unconstrained problem with delay strategies

2.3.1 Problem formulation

We start by presenting the most important property of delay strategies which allows us to put the game in the so-called *normal form*. This result corresponds to [45, Lemma 2.3].

Lemma 2.3.1. *Let* $(\alpha[\cdot], \beta[\cdot]) \in \Gamma \times \Delta$ *be two nonanticipative strategies. Assume that either* $\alpha[\cdot]$ *or* $\beta[\cdot]$ *is a delay strategy. Then there is a unique pair of controls* $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$ *such that*

$$a(\cdot) = \alpha[b](\cdot) \quad \textit{and} \quad b(\cdot) = \beta[a](\cdot) \quad \textit{almost everywhere on} \quad [0,T]. \tag{2.3}$$

Proof. Without loss of generality, consider a delay strategy of the first player $\alpha[\cdot] \in \Gamma_d$, with a delay $\tau_\alpha > 0$, and a nonanticipative strategy of the second player $\beta[\cdot] \in \Delta$.

We claim that for any integer $k \geq 1$, there exists a unique pair of measurable maps $(a_k(\cdot), b_k(\cdot)) : [0, k\tau_\alpha] \to A \times B$ s.t. $\alpha[b_k](\cdot) = a_k(\cdot)$ and $\beta[a_k](\cdot) = b_k(\cdot)$ on $[0, k\tau_\alpha]$. We will prove this claim by induction.

For $k = 1$, let's pick a control $b(\cdot) \in \mathcal{B}$ and set $a_1(\cdot) = \alpha[b](\cdot) \in \mathcal{A}$. Then let $b_1(\cdot) = \beta[a_1](\cdot)$. From remark 2.2.3 and since $\alpha[\cdot]$ is a delay strategy, the restriction of $\alpha[b](\cdot)$ to $[0, \tau_\alpha]$ is independent of $b(\cdot)$. Hence, $\alpha[b_1](\cdot) = \alpha[b](\cdot) = a_1(\cdot)$ on $[0, \tau_\alpha]$. Therefore the claim holds for $k = 1$.

Suppose now that the claim is true for any $k \geq 1$ and let's prove it for $k+1$.

There is a unique pair $(a_k(\cdot), b_k(\cdot)) : [0, k\tau_\alpha] \to A \times B$ s.t. $\alpha[b_k](\cdot) = a_k(\cdot)$ and $\beta[a_k](\cdot) = b_k(\cdot)$ on $[0, k\tau_\alpha]$. We extend $a_k(\cdot)$ and $b_k(\cdot)$ to arbitrary controls on $[0, T]$. For this, we set $a_{k+1}(\cdot) = \alpha[b_k](\cdot)$ and $b_{k+1}(\cdot) = \beta[a_{k+1}](\cdot)$. Then from this construction, $a_{k+1}(\cdot) = a_k(\cdot)$ a.e. on $[0, k\tau_\alpha]$. Since $\beta[\cdot]$ is a nonanticipative strategy, we get $b_k(\cdot) = \beta[a_k](\cdot) = \beta[a_{k+1}](\cdot) = b_{k+1}(\cdot)$ a.e. on $[0, k\tau_\alpha]$. Now since $\alpha[\cdot]$ is a delay strategy with delay τ_α, then $a_{k+1}(\cdot) = \alpha[b_k](\cdot) = \alpha[b_{k+1}](\cdot)$ $a.e.$ on $[0, (k+1)\tau_\alpha]$. This completes the proof of the claim by induction. □

Thanks to the above Lemma, the cost functional J defined in (2.2) can be extended to any couple of delay strategies $(\alpha[\cdot], \beta[\cdot]) \in \Gamma_d \times \Delta_d$ as follows:

$$J(t, x, \alpha, \beta) = J(t, x, a, b),$$

where $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$ is the unique pair defined in Lemma 2.3.1.

Now, we can define the lower and the upper values of the game in the context of delay strategies.

Definition 2.3.2. *The upper value function is given by:*

$$v^+(t, x) := \inf_{\alpha[\cdot] \in \Gamma_d} \sup_{\beta[\cdot] \in \Delta_d} J(t, x, \alpha, \beta), \tag{2.4}$$

while the lower value is defined by:

$$v^-(t, x) := \sup_{\beta[\cdot] \in \Delta_d} \inf_{\alpha[\cdot] \in \Gamma_d} J(t, x, \alpha, \beta). \tag{2.5}$$

Remark 2.3.3. *From Definition 2.3.2, we can already compare the value functions v^+ and v^-. The following inequality is always true : $v^- \leq v^+$.*

Thanks again to Lemma 2.3.1, the value functions v^+ and v^- can be expressed differently.

Lemma 2.3.4 (Equivalent definitions of the value functions)**.** *We have:*

$$v^+(t, x) := \inf_{\alpha[\cdot] \in \Gamma_d} \sup_{b(\cdot) \in \mathcal{B}} J(t, x, \alpha[b], b), \tag{2.6}$$

and

$$v^-(t, x) := \sup_{\beta[\cdot] \in \Delta_d} \inf_{a(\cdot) \in \mathcal{A}} J(t, x, a, \beta[a]). \tag{2.7}$$

Proof. We will prove the first equality for v^+, the second equality can be obtained by the same arguments.

Let $\alpha_0[\cdot] \in \Gamma_d$ and consider a constant delay strategy of the second player $\beta_0[\cdot] \in \Delta_d$ such that, for any $b(\cdot) \in \mathcal{B}$, $\beta_0[b](\cdot) = b_0(\cdot)$ for some fixed $b_0(\cdot) \in \mathcal{B}$. By Lemma 2.3.1, we have:

$$J(t, x, \alpha_0, b_0) = J(t, x, \alpha_0[b_0], b_0).$$

Since the controls in $\mathcal{B}$ can be considered as constant strategies of the second player, i.e. $\mathcal{B} \subset \Delta_d$, we get for any $\alpha[\cdot] \in \Gamma_d$

$$\sup_{b(\cdot) \in \mathcal{B}} J(t, x, \alpha, b) \leq \sup_{\beta[\cdot] \in \Delta_d} J(t, x, \alpha, \beta),$$

henceforth

$$\inf_{\alpha[\cdot] \in \Gamma_d} \sup_{b(\cdot) \in \mathcal{B}} J(t, x, \alpha[b], b) \leq v^+(t, x).$$

By Lemma 2.3.1, for any $(\alpha[\cdot], \beta[\cdot]) \in \Gamma_d \times \Delta_d$, there is a unique pair of controls $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$ such that (2.3) holds. Then

$$J(t,x,\alpha,\beta) = J(t,x,\alpha[b],b) \leq \sup_{b'(\cdot)\in\mathcal{B}} J(t,x,\alpha[b'],b').$$

Therefore

$$\sup_{\beta[\cdot]\in\Delta_d} J(t,x,\alpha,\beta) \leq \sup_{b'(\cdot)\in\mathcal{B}} J(t,x,\alpha[b'],b').$$

Taking the infimum over $\alpha[\cdot] \in \Gamma_d$ in the last inequality completes the proof. □

Remark 2.3.5. *Notice that*

$$-v^-(t,x) = \inf_{\beta\in\Delta_d} \sup_{a\in\mathcal{A}} -J(t,x,a,\beta[a]),$$

which means that the value function $(-v^-)$ can be seen as an upper value of a game with running cost $-\ell$, terminal payoff $-\Psi$, and where the first player is the maximizer while the second player is the minimizer. Henceforth, any result verified by v^+ can be directly deduced for $(-v^-)$ and hence for v^-.

2.3.2 Some properties of value functions

In this subsection, we discuss and prove some properties verified by the value functions. Thanks to remark 2.3.5, we give the proofs only for the upper value function v^+. The proofs for v^- can be obtained by similar arguments.

We start by the following dynamic programming principle which holds for v^+.

Theorem 2.3.6. *Assume* **(H2.1)**, **(H2.2)** *and* **(H2.3)**. *For any $h \in [0, T-t]$, we have:*

$$v^+(t,x) = \inf_{\alpha[\cdot]\in\Gamma_d} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \int_t^{t+h} \ell(s, y_{t,x}^{\alpha[b],b}(s), \alpha[b](s), b(s))ds + v^+(t+h, y_{t,x}^{\alpha[b],b}(t+h)) \Big\}. \tag{2.8}$$

Theorem 2.3.6 is a classical result and we refer to [45, Theorem 3.4] for its proof.

Now, we cite the following result concerning the regularity of the value functions.

Proposition 2.3.7. *The value functions v^+ and v^- are Lipschitz continuous on $[0,T] \times \Omega$ for any compact set $\Omega \subset \mathbb{R}^d$.*

In order to prove Proposition 2.3.7, we will use the following result:

Lemma 2.3.8. *Let U and V be two arbitrary sets and let g, $h : U \times V \to \mathbb{R}$ be two maps. Assume that there is a real constant $k \geq 0$ such that*

$$\sup_{u\in U, v\in V} |g(u,v) - h(u,v)| \leq k.$$

Therefore

$$|\inf_{u\in U} \sup_{v\in V} g(u,v) - \inf_{u\in U} \sup_{v\in V} h(u,v)| \leq k,$$

as soon as the inf-sup of g or h is finite.

The proof of Lemma 2.3.8 is not complicated and can be found in [45].

Proof. We start by proving that v^+ is Lipschitz continuous w.r.t. the x variable uniformly in the time variable. To this end, let $(t, x_1, x_2) \in [0,T] \times \mathbb{R}^d \times \mathbb{R}^d$ and $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$ be fixed. We set $y_1(\cdot) := y^{a,b}_{t,x_1}(\cdot)$ and $y_2(\cdot) := y^{a,b}_{t,x_2}(\cdot)$. Since f is Lipschitz continuous, Gronwall's Lemma implies that for any $s \in [t,T]$:

$$\|y_1(s) - y_2(s)\| \leq e^{L_1(s-t)}\|x_1 - x_2\|.$$

By using the Lipschitz continuity of ℓ and Ψ and the Gronwall's Lemma, we get:

$$\begin{aligned}
|J(t,x_1,a,b) - J(t,x_2,a,b)| &\leq \int_t^T |\ell(s,y_1(s),a(s),b(s)) - \ell(s,y_2(s),a(s),b(s))|ds \\
&\quad + |\Psi(y_1(T)) - \Psi(y_2(T))| \\
&\leq L_2 \int_t^T \|y_1(s) - y_2(s)\|ds + L_3\|y_1(T) - y_2(T)\| \\
&\leq C\|x_1 - x_2\|,
\end{aligned}$$

where the real constant $C > 0$ depends only on L_1, L_2, L_3 and T.

The above inequality holds for any pair of controls $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$. From Lemma 2.3.1, we get for any $(\alpha[\cdot], \beta[\cdot]) \in \Gamma_d \times \Delta_d$:

$$|J(t,x_1,\alpha,\beta) - J(t,x_2,\alpha,\beta)| \leq C\|x_1 - x_2\|,$$

henceforth

$$\sup_{(\alpha[\cdot],\beta[\cdot])\in\Gamma_d\times\Delta_d} |J(t,x_1,\alpha,\beta) - J(t,x_2,\alpha,\beta)| \leq C\|x_1 - x_2\|,$$

and finally by using Lemma 2.3.8, we get the Lipschitz continuity with respect to the space variable.

Now let $(t,t',x) \in [0,T] \times [0,T] \times \mathbb{R}^d$ be fixed and without loss of generality assume that $t' > t$. We fix also $\epsilon > 0$, there exists $\alpha_\epsilon[\cdot] \in \Gamma$ and $b_\epsilon(\cdot) \in \mathcal{B}$ such that:

$$\begin{aligned}
v^+(t,x) &\geq \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \int_t^T \ell(s, y^{\alpha_\epsilon[b],b}_{t,x}(s), \alpha_\epsilon[b](s), b(s))ds + \Psi(y^{\alpha_\epsilon[b],b}_{t,x}(T)) \Big\} - \epsilon \\
&\geq \int_t^T \ell(s, y^{\alpha_\epsilon[b_\epsilon],b_\epsilon}_{t,x}(s), \alpha_\epsilon[b_\epsilon](s), b_\epsilon(s))ds + \Psi(y^{\alpha_\epsilon[b_\epsilon],b_\epsilon}_{t,x}(T)) \Big\} - \epsilon,
\end{aligned}$$

and

$$\begin{aligned}
v^+(t',x) &\leq \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \int_{t'}^T \ell(s, y^{\alpha_\epsilon[b],b}_{t',x}(s), \alpha_\epsilon[b](s), b(s))ds + \Psi(y^{\alpha_\epsilon[b],b}_{t',x}(T)) \Big\} \\
&\leq \int_{t'}^T \ell(s, y^{\alpha_\epsilon[b_\epsilon],b_\epsilon}_{t',x}(s), \alpha_\epsilon[b_\epsilon](s), b_\epsilon(s))ds + \Psi(y^{\alpha_\epsilon[b_\epsilon],b_\epsilon}_{t',x}(T)) \Big\} + \epsilon.
\end{aligned}$$

From the two above inequalities, we deduce that:

$$\begin{aligned}
v^+(t',x) - v^+(t,x) &\leq 2\epsilon + \int_t^{t'} |\ell(s, y^{\alpha_\epsilon[b_\epsilon],b_\epsilon}_{t,x}(s), \alpha_\epsilon[b_\epsilon](s), b_\epsilon(s))|ds \\
&\quad + \int_{t'}^T |\ell(s, y^{\alpha_\epsilon[b_\epsilon],b_\epsilon}_{t',x}(s), \alpha_\epsilon[b_\epsilon](s), b_\epsilon(s)) - \ell(s, y^{\alpha_\epsilon[b_\epsilon],b_\epsilon}_{t,x}(s), \alpha_\epsilon[b_\epsilon](s), b_\epsilon(s))|ds \\
&\quad + |\Psi(y^{\alpha_\epsilon[b_\epsilon],b_\epsilon}_{t',x}(T)) - \Psi(y^{\alpha_\epsilon[b_\epsilon],b_\epsilon}_{t,x}(T))|.
\end{aligned}$$

Moreover, the trajectory $y_{t,x}^{\alpha_\epsilon[b_\epsilon],b_\epsilon}(\cdot)$ coincides with $y_{t',x'}^{\alpha_\epsilon[b_\epsilon],b_\epsilon}(\cdot)$ for $s \geq t'$ where $x' := y_{t,x}^{\alpha_\epsilon[b_\epsilon],b_\epsilon}(t')$. Henceforth, the last inequality becomes:

$$v^+(t',x) - v^+(t,x) \leq 2\epsilon + M|t'-t| + L_2 \int_{t'}^{T} \|y_{t',x}^{\alpha_\epsilon[b_\epsilon],b_\epsilon}(s) - y_{t',x'}^{\alpha_\epsilon[b_\epsilon],b_\epsilon}(s)\| ds \\ + L_3 \|y_{t',x}^{\alpha_\epsilon[b_\epsilon],b_\epsilon}(T) - y_{t',x'}^{\alpha_\epsilon[b_\epsilon],b_\epsilon}(T)\|,$$

where M is an upper bound of $s \mapsto \ell(s, y_{t,x}^{\alpha_\epsilon[b_\epsilon],b_\epsilon}(s), \alpha_\epsilon[b_\epsilon](s), b_\epsilon(s))$, for $s \in [t,t']$.

Since f is Lipschitz continuous and by using Gronwall's Lemma, there exist $C_1 > 0$ and $C_2 > 0$ such that:

$$\|y_{t',x}^{\alpha_\epsilon[b_\epsilon],b_\epsilon}(s) - y_{t',x'}^{\alpha_\epsilon[b_\epsilon],b_\epsilon}(s)\| \leq C_1 \|x - x'\|, \ \ \forall s \in [t,t'] \quad \text{and} \quad \|x - x'\| \leq C_2 |t'-t|.$$

More precisely, the exact expressions of C_1 and C_2 are given by:

$$C_1 := e^{L_1 T} \quad \text{and} \quad C_2 := e^{L_1 T}(L_1 \|x\| + C_f),$$

where $C_f := \max\Big\{ \|f(s,0,a,b)\| \quad \text{for} \quad (s,a,b) \in [0,T] \times A \times B \Big\}$.

Combining the above inequalities implies the existence of $C_3 > 0$, depending on M, L_2, L_3, C_1 and C_2, such that:

$$v^+(t',x) - v^+(t,x) \leq 2\epsilon + C_3|t'-t|.$$

In a similar way, one can prove that:

$$v^+(t,x) - v^+(t',x) \leq 2\epsilon + C_3|t'-t|,$$

which ends the proof since ϵ is chosen arbitrarily.

□

2.3.3 Characterization of value functions

For this game, one can define the two following Hamilton-Jacobi-Isaacs equations:

$$\begin{cases} -\partial_t V + H^-(t,x,D_x V) = 0, \quad t \in [0,T[, \ x \in \mathbb{R}^d \\ V(T,x) = \Psi(x), \quad x \in \mathbb{R}^d, \end{cases} \tag{2.9}$$

where the hamiltonian H^- is given by:

$$H^-(t,x,p) := \max_{a \in A} \ \min_{b \in B} \Big\{ -\langle f(t,x,a,b), p\rangle - \ell(t,x,a,b) \Big\}, \tag{2.10}$$

and

$$\begin{cases} -\partial_t V + H^+(t,x,D_x V) = 0, \quad t \in [0,T[, \ x \in \mathbb{R}^d \\ V(T,x) = \Psi(x), \quad x \in \mathbb{R}^d, \end{cases} \tag{2.11}$$

where H^+ is defined by:

$$H^+(t,x,p) := \min_{b \in B} \ \max_{a \in A} \Big\{ -\langle f(t,x,a,b), p\rangle - \ell(t,x,a,b) \Big\}. \tag{2.12}$$

The following Lemma gives a half-characterization of the value functions v^+ and v^-. Its proof can be found in [45, Lemma 3.15]

Lemma 2.3.9 (Viscosity solutions)**.** *Assume* **(H2.1)**, **(H2.2)** *and* **(H2.3)**. *The upper value v^+ is a viscosity sub-solution of (2.9), while the lower value v^- is a super-solution of (2.11).*

Now we present a comparison principle result which gives an order between a sub-solution and a super-solution of an Hamilton-Jacobi equation. Consider the following HJ equation:

$$\begin{cases} -\partial_t V + H(t,x,D_x V) = 0, \quad t \in [0,T[, x \in \mathbb{R}^d \\ V(T,x) = g(x), \quad x \in \mathbb{R}^d \end{cases} \tag{2.13}$$

where g is a given continuous function and the hamiltonian H:$[0,T]\times\mathbb{R}^d\times\mathbb{R}^d \to \mathbb{R}$ is a continuous function satisfying the following Lipschitz properties:

$$\|H(t_2,x_2,p) - H(t_1,x_1,p)\| \leq C(1+\|p\|)(|t_2-t_1| + \|x_2-x_1\|) \tag{2.14}$$

and

$$\|H(t,x,p_2) - H(t,x,p_1)\| \leq C\|p_2-p_1\| \tag{2.15}$$

for any $t_1, t_2 \in [0,T]$, $x_1, x_2, p_1, p_2 \in \mathbb{R}^d$ and where $C > 0$.

Theorem 2.3.10 (Comparison principle)**.** *Under assumptions (2.14) and (2.15), let V_1 be an $u.s.c.$ sub-solution of (2.13) and let V_2 be a $l.s.c.$ super-solution of the same HJ equation. Assume that $V_1(T,x) \leq V_2(T,x)$ for any $x \in \mathbb{R}^d$, then*

$$V_1(t,x) \leq V_2(t,x), \ \ \forall (t,x) \in [0,T]\times\mathbb{R}^d.$$

In order to prove Theorem 2.3.10, we will need the following result from [45]. We postpone its proof to the end of the proof of Theorem 2.3.10.

Lemma 2.3.11. *Assume that H is continuous and satisfies (2.15). If u is an u.s.c function and a sub-solution of (2.13) on $[0,T]\times\mathbb{R}^d$ (resp. is a l.s.c. function and a super-solution of (2.13) on $[0,T]\times\mathbb{R}^d$), then for any $(t_0,x_0) \in [0,T[\times\mathbb{R}^d$, u is also a sub-solution (resp. super-solution) in the cone*

$$C_{t_0,x_0} := \Big\{(t,x) \in [t_0,T]\times\mathbb{R}^d, \|x-x_0\| \leq C(t-t_0)\Big\}.$$

In other words, if a function ϕ of class C^1 such that $u-\phi$ is maximal (resp. minimal) on C_{t_0,x_0} at some point (t,x) with $t<T$, then

$$-\partial_t\phi(t,x) + H(t,x,D_x\phi(t,x)) \leq 0, \quad (\textit{resp.} \ \geq 0).$$

The aim of Lemma 2.3.11 is to restrict V_1 and V_2 to bounded domains while preserving the property of being sub and super-solution of (2.13). In other words, this lemma claims that a solution on a set is also a solution on a bounded subset of that set.

Proof. Now, we can start the proof of Theorem 2.3.10. We claim that:

$$\text{for any } \ \sigma > 0, \ \ \forall (t,x) \in [0,T]\times\mathbb{R}^d, \ \ V_1(t,x) - V_2(t,x) - \sigma(T-t) \leq 0. \tag{2.16}$$

The claim (2.16) can be shown by contradiction. Hence, suppose that there exists $\sigma_0 > 0$ and $(t_0,x_0) \in [0,T[\times\mathbb{R}^d$ such that

$$M := \sup_{(t,x)\in C_{t_0,x_0}} V_1(t,x) - V_2(t,x) - \sigma_0(T-t) \geq V_1(t_0,x_0) - V_2(t_0,x_0) - \sigma_0(T-t_0) > 0.$$

Now, we use the *doubling variable technique*. Indeed, for $\epsilon > 0$ we set:

$$\phi_\epsilon((t,x),(s,y)) = V_1(t,x) - V_2(s,y) - \frac{1}{2\epsilon}\|(s,y)-(t,x)\|^2 - \sigma_0(T-s)$$

and consider the following optimization problem:

$$M_\epsilon := \sup_{(t,x),(s,y)\in C_{t_0,x_0}} \phi_\epsilon((t,x),(s,y)). \tag{2.17}$$

Note that $M_\epsilon \geq M$. Since V_1 and $-V_2$ are u.s.c. functions in C_{t_0,x_0}, so is the map ϕ_ϵ. On the other hand, C_{t_0,x_0} is a compact set, therefore problem (2.17) has a maximum point denoted by $((t_\epsilon, x_\epsilon),(s_\epsilon, y_\epsilon))$.

Since the map $(t,x) \to \phi_\epsilon((t,x),(s_\epsilon,y_\epsilon))$ has a maximum at the point (t_ϵ, x_ϵ) on C_{t_0,x_0}, we have for any $(t,x) \in C_{t_0,x_0}$:

$$V_1(t,x) \leq \phi(t,x) := V_1(t_\epsilon,x_\epsilon) + \frac{1}{2\epsilon}\Big(\|(s_\epsilon,y_\epsilon)-(t,x)\|^2 - \|(s_\epsilon,y_\epsilon)-(t_\epsilon,x_\epsilon)\|^2\Big).$$

Notice that ϕ is a smooth function which coincides with V_1 at (t_ϵ, x_ϵ). Therefore, $V_1 - \phi$ has a maximum at (t_ϵ, x_ϵ) on the set C_{t_0,x_0}. Since V_1 is a sub-solution of (2.13) and $t_\epsilon < T$, Lemma 2.3.11 implies that

$$-\partial_t\phi(t_\epsilon,x_\epsilon) + H(t_\epsilon,x_\epsilon,D_x\phi) \leq 0,$$

which means that

$$-\frac{t_\epsilon - s_\epsilon}{\epsilon} + H\Big(t_\epsilon, x_\epsilon, \frac{x_\epsilon - y_\epsilon}{\epsilon}\Big) \leq 0. \tag{2.18}$$

In a symmetric way, since the map $(s,y) \to \phi_\epsilon((t_\epsilon,x_\epsilon),(s,y))$ has a maximum at the point (s_ϵ, y_ϵ) on C_{t_0,x_0}, we get, for any $(s,y) \in C_{t_0,x_0}$:

$$V_2(s,y) \geq V_2(s_\epsilon,y_\epsilon) - \frac{1}{2\epsilon}\Big(\|(s,y)-(t_\epsilon,x_\epsilon)\|^2 - \|(s_\epsilon,y_\epsilon)-(t_\epsilon,x_\epsilon)\|^2\Big) + \sigma_0(s-s_\epsilon)$$

and since V_2 is a super-solution of (2.13), we obtain again by Lemma 2.3.11:

$$-\frac{t_\epsilon - s_\epsilon}{\epsilon} - \sigma_0 + H\Big(s_\epsilon, y_\epsilon, \frac{x_\epsilon - y_\epsilon}{\epsilon}\Big) \geq 0. \tag{2.19}$$

The difference between (2.18) and (2.19) gives

$$-\sigma_0 + H\Big(s_\epsilon, y_\epsilon, \frac{x_\epsilon - y_\epsilon}{\epsilon}\Big) - H\Big(t_\epsilon, x_\epsilon, \frac{x_\epsilon - y_\epsilon}{\epsilon}\Big) \geq 0.$$

Now, from assumption (2.15) on H, we obtain:

$$-\sigma_0 - C\Big(1 + \frac{\|x_\epsilon - y_\epsilon\|}{\epsilon}\Big)\|x_\epsilon - y_\epsilon\| \geq 0.$$

Finally, by letting $\epsilon \to 0^+$ and using Lemma 2.3.12, we get a contradiction since we have supposed that $\sigma_0 > 0$. □

The following Lemma, used in the proof of Theorem 2.3.10, gives some estimates on $((t_\epsilon, x_\epsilon),(s_\epsilon, y_\epsilon))$.

Lemma 2.3.12. *We have:*

(i) $\lim_{\epsilon\to 0^+} M_\epsilon = M.$

(ii) $\lim_{\epsilon\to 0^+} \frac{1}{\epsilon}\|(s_\epsilon,y_\epsilon)-(t_\epsilon,x_\epsilon)\|^2 = 0.$

(iii) *For ϵ small enough, $s_\epsilon < T$ and $t_\epsilon < T$.*

Proof of Lemma 2.3.11. Here, we give only the proof of the result concerning sub-solutions. In order to prove the result verified by a super-solution, one can consider its opposite, which is a sub-solution of some modified HJ equation.

Let u be an u.s.c function and a sub-solution of (2.13) on $[0,T]\times\mathbb{R}^d$ and let ϕ be a function of class C^1 s.t. $u-\phi$ has a strict local maximum on C_{t_0,x_0} at some point (t,x). For $\sigma>0$, we consider the map ϕ_σ defined, for $(s,y)\in C(t_0,x_0)$, by:

$$\phi_\sigma(s,y)=u(s,y)-\phi(s,y)+\frac{\sigma}{2}\log(C^2(s-t_0)^2-\|y-x_0\|^2)$$

Since C_{t_0,x_0} is a compact set and $\phi_\sigma(s,y)\to-\infty$ as soon as (s,y) goes to some point at the boundary of C_{t_0,x_0}, ϕ_σ has a maximum point (s_σ,y_σ) on C_{t_0,x_0}.

Notice that $(s_\sigma,y_\sigma)\to(t,x)$ when $\sigma\to0^+$. Henceforth, for σ small enough, we get $s_\sigma<T$ because we have considered $t<T$ in Lemma 2.3.11.

On the other hand, since u is a sub-solution of (2.13), we obtain:

$$\partial_t\phi(s_\sigma,y_\sigma)-\sigma C^2\frac{s_\sigma-t_0}{A_\sigma}+H(s_\sigma,y_\sigma,D_x\phi(s_\sigma,y_\sigma)+\sigma\frac{y_\sigma-x_0}{A_\sigma})\geq0,$$

where $A_\sigma:=C^2(s_\sigma-t_0)^2-\|y_\sigma-x_0\|^2$. By using assumption (2.15), we deduce from the above inequality

$$\partial_t\phi(s_\sigma,y_\sigma)-\sigma C\frac{C(s_\sigma-t_0)-\|y_\sigma-x_0\|}{A_\sigma}+H(s_\sigma,y_\sigma,D_x\phi(s_\sigma,y_\sigma))\geq0.$$

Now, the fact that $A_\sigma>0$ and $C(s_\sigma-t_0)-\|y_\sigma-x_0\|>0$ gives:

$$\partial_t\phi(s_\sigma,y_\sigma)+H(s_\sigma,y_\sigma,D_x\phi(s_\sigma,y_\sigma))\geq0.$$

As a conclusion, since $\partial_t\phi$ and H are continuous (ϕ is a function of class C^1), and by letting $\sigma\to0^+$, we obtain the desired result. □

Proof of Lemma 2.3.12. We already know that $M_\epsilon\geq M$. Let K be an upper bound for V_1-V_2 on C_{t_0,x_0}. Therefore,

$$\begin{aligned}0<M\leq M_\epsilon&=V_1(t_\epsilon,x_\epsilon)-V_2(s_\epsilon,y_\epsilon)-\frac{1}{2\epsilon}\|(s_\epsilon,y_\epsilon)-(t_\epsilon,x_\epsilon)\|^2-\sigma(T-s_\epsilon)\\&\leq K-\frac{1}{2\epsilon}\|(s_\epsilon,y_\epsilon)-(t_\epsilon,x_\epsilon)\|^2.\end{aligned}$$

This implies that $\frac{1}{2\epsilon}\|(s_\epsilon,y_\epsilon)-(t_\epsilon,x_\epsilon)\|^2$ is bounded and therefore $\Big((s_\epsilon,y_\epsilon)-(t_\epsilon,x_\epsilon)\Big)\to0$ when $\epsilon\to0^+$.

Now, let (t,x) be a cluster point of the bounded sequences $(s_\epsilon,y_\epsilon)_\epsilon$ and $(t_\epsilon,x_\epsilon)_\epsilon$. Since V_1-V_2 is an u.s.c function, we get:

$$M\leq\liminf_{\epsilon\to0}M_\epsilon\leq\limsup_{\epsilon\to0}\quad V_1(t_\epsilon,x_\epsilon)-V_2(s_\epsilon,y_\epsilon)-\sigma(T-s_\epsilon)\leq V_1(t,x)-V_2(t,x)-\sigma(T-t)\leq M$$

Henceforth, $M_\epsilon\to M$ when $\epsilon\to0$ and

$$\lim_{\epsilon\to0}\quad V_1(t_\epsilon,x_\epsilon)-V_2(s_\epsilon,y_\epsilon)-\sigma(T-s_\epsilon)=V_1(t,x)-V_2(t,x)-\sigma(T-t)=M,$$

which implies that:

$$\lim_{\epsilon\to0}\quad\frac{1}{2\epsilon}\|(s_\epsilon,y_\epsilon)-(t_\epsilon,x_\epsilon)\|^2=\lim_{\epsilon\to0}\quad V_1(t_\epsilon,x_\epsilon)-V_2(s_\epsilon,y_\epsilon)-\sigma(T-s_\epsilon)-M=0.$$

Finally, we will show that $t<T$. Suppose that $t=T$, therefore

$$M\leq V_1(T,x)-V_2(T,x)\leq0$$

which is impossible since we have supposed that $M>0$. Henceforth $s_\epsilon<T$ and $t_\epsilon<T$ for ϵ small enough, which completes the proof of this Lemma. □

Now, we introduce a sufficient condition for which the game has a value which means that the lower value v^- coincides with the upper value v^+. This condition is called the Isaacs' condition, which holds when:

$$H^+(t,x,p) = H^-(t,x,p), \quad \forall (t,x,p) \in [0,T] \times \mathbb{R}^d \times \mathbb{R}^d. \tag{2.20}$$

Theorem 2.3.13 (Value of the game)**.** *Assume* **(H2.1)**, **(H2.2)** *and* **(H2.3)**. *If Isaacs' condition* (2.20) *holds, then the game has a value $v = v^+ = v^-$. Moreover, this value is the unique viscosity solution of the Hamilton-Jacobi-Isaacs' equation* (2.9) *(or* (2.11)*, since* (2.9) *and* (2.11) *are reduced to the same HJI equation with the same hamiltonian function $H = H^+ = H^-$).*

Proof. If Isaacs' condition holds, v^- which is a viscosity super-solution of (2.11), from Lemma 2.3.9, becomes a viscosity super-solution of (2.9). Moreover, from Lemma 2.3.9, v^+ is a viscosity sub-solution of (2.9). Henceforth, from Theorem 2.3.10, we obtain $v^- \geq v^+$. On the other hand, we have already $v^+ \geq v^-$ (see remark 2.3.3), which concludes the proof. □

2.4 Unconstrained problem with nonanticpative strategies

In this section, we focus on the context of nonanticipative strategies. In particular, we extend the results of Capuzzo-Dolcetta and Bardi in [11, Chapter VIII], concerning the trajectory reconstruction for both players, where they studied a differential game with infinite time horizon.

2.4.1 Problem formulation

We consider again a problem of type Bolza where the cost function J is defined as in (2.2) and we use the same hypothesis introduced in section 2.1. We define two value functions of this game according to the player having the advantage of information.

The value function of the first player corresponds to the case when he uses nonanticpative strategies $\alpha[\cdot] \in \Gamma$ to minimize the objective function J. It can be defined as

$$v^\sharp(t,x) := \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} J(t,x,\alpha[b],b). \tag{2.21}$$

By the same way, we define the value function of the second player which corresponds to the case when he uses nonanticpative strategies $\beta[\cdot] \in \Delta$ to maximize the objective function J:

$$v^\flat(t,x) := \sup_{\beta[\cdot]\in\Delta} \inf_{a(\cdot)\in\mathcal{A}} J(t,x,a,\beta[a]). \tag{2.22}$$

Remark 2.4.1. *From this definition, we cannot compare directly the value functions $v^\sharp$ and $v^\flat$ since the infimum and the supremum are not taken over the same sets of controls and strategies. Nevertheless, we will be able to give an order between the two value functions after their characterisation as unique viscosity solutions to two different Hamilton-Jacobi-Isaacs equations and by exploiting the comparison principle Theorem 2.3.10.*

2.4.2 Characterization of value functions

The aim of this part is to characterize the value functions $v^\sharp$ and $v^\flat$ by means of HJI equations. To this end, we start by presenting a dynamic programming principle and a regularity property verified by both value functions.

Theorem 2.4.2. *Assume* **(H2.1)**, **(H2.2)** *and* **(H2.3)**.

(i) *For* $t \in [0,T]$, $x \in \mathbb{R}^d$ *and* $h \in [0, T-t]$, *we have*

$$v^\sharp(t,x) = \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \left\{ \int_t^{t+h} \ell(s, y_{t,x}^{\alpha[b],b}(s), \alpha[b](s), b(s))ds + v^\sharp(t+h, y_{t,x}^{\alpha[b],b}(t+h)) \right\}, \tag{2.23}$$

and

$$v^\flat(t,x) = \sup_{\beta[\cdot]\in\Delta} \inf_{a(\cdot)\in\mathcal{A}} \left\{ \int_t^{t+h} \ell(s, y_{t,x}^{a,\beta[a]}(s), a(s), \beta[a](s))ds + v^\flat(t+h, y_{t,x}^{a,\beta[a]}(t+h)) \right\}. \tag{2.24}$$

(ii) *Both value functions* $v^\sharp$ *and* $v^\flat$ *are Lipschitz continuous on* $[0,T]\times\Omega$, *for any compact set* $\Omega \subset \mathbb{R}^d$.

The proof of (i) can be done by using the same arguments as in [11, Chapter VIII]. As for (ii), it can be proven in a similar way to Proposition 2.3.7.

From the previous result, we get the following characterization of the two players value functions $v^\sharp$ and $v^\flat$.

Theorem 2.4.3. $v^\sharp$ *is the unique viscosity solution of* (2.11) *while* $v^\flat$ *is the unique viscosity solution of* (2.9).

In order to prove Theorem 2.4.3, we will need the following result whose proof is postponed to the end.

Lemma 2.4.4. *Let* ϕ *be a function of class* C^1 *such that* $-\partial_t\phi(t,x) + H^+(t,x,D_x\phi(t,x)) = \delta > 0$. *Then, there exists a strategy* $\alpha^*[\cdot] \in \Gamma$, *such that for any* $b(\cdot) \in \mathcal{B}$ *and* $h > 0$ *small enough:*

$$\int_t^{t+h} \Big(\partial_t\phi(s,y^*(s)) + \langle f(s,y^*(s),a^*(s),b(s)), D_x\phi(s,y^*(s))\rangle + \ell(s,y^*(s),a^*(s),b(s)) \Big) ds \le -\frac{\delta h}{4},$$

where $a^*(\cdot) := \alpha^*[b](\cdot) \in \mathcal{A}$ *and* $y^*(\cdot) := y_{t,x}^{a^*,b}(\cdot)$.

Proof. We will prove the result only for $v^\sharp$, and by the same way, one can get the proof of the result for $v^\flat$.

We start by showing that $v^\sharp$ is a viscosity super-solution of (2.11). Let ϕ be a function of class C^1 and (t,x) be a local minimum for $v^\sharp - \phi$ such that $(v^\sharp - \phi)(t,x) = 0$. Assume that there exists $\delta > 0$ such that:

$$-\partial_t\phi(t,x) + H^+(t,x,D_x\phi(t,x)) = -\delta < 0.$$

Therefore, there exists $b^* \in B$, such that for any $a \in A$,

$$-\partial_t\phi(t,x) - \langle f(t,x,a,b^*), D_x\phi(t,x)\rangle - \ell(t,x,a,b^*) \le -\delta.$$

Now, let's fix $h > 0$ small enough and $\alpha[\cdot] \in \Gamma$. Denote by $a^*(\cdot) := \alpha[b^*](\cdot) \in \mathcal{A}$ and by $y^*(\cdot) := y_{t,x}^{a^*,b^*}$. For any $s \in [t,T]$, we have $a^*(s) \in A$. From the last inequality, we deduce that

$$-\partial_t\phi(t,x) - \langle f(t,x,a^*(s),b^*), D_x\phi(t,x)\rangle - \ell(t,x,a^*(s),b^*) \le -\delta, \quad \text{for any} \quad s \in [t,T].$$

By continuity of f,ℓ, $\partial_t\phi$ and $D_x\phi$, we get for any $s \in [t, t+h]$:

$$-\partial_t\phi(s,y^*(s)) - \langle f(s,y^*(s),a^*(s),b^*), D_x\phi(s,y^*(s))\rangle - \ell(s,y^*(s),a^*(s),b^*) \le -\frac{\delta}{2},$$

henceforth

$$\int_t^{t+h} \Big(-\partial_t\phi(s,y^*(s)) - \langle f(s,y^*(s),a^*(s),b^*), D_x\phi(s,y^*(s))\rangle - \ell(s,y^*(s),a^*(s),b^*) \Big) ds \le -\frac{\delta h}{2},$$

which implies

$$\phi(t,x) - \phi(t+h, y^*(t+h)) - \int_t^{t+h} \ell(s, y^*(s), a^*(s), b^*)ds \leq -\frac{\delta h}{2}.$$

On the other hand, for any $s \in [t, t+h]$, we have:

$$(v^\sharp - \phi)(s, y^*(s)) \geq (v^\sharp - \phi)(t,x).$$

From the two above inequalities, we deduce that:

$$\frac{\delta h}{2} + v^\sharp(t,x) \leq v^\sharp(t+h, y^*(t+h)) + \int_t^{t+h} \ell(s, y^*(s), a^*(s), b^*)ds.$$

Therefore we get

$$\frac{\delta h}{2} + v^\sharp(t,x) \leq \sup_{b(\cdot)\in\mathcal{B}} \left\{ v^\sharp(t+h, y_{t,x}^{\alpha[b],b}(t+h)) + \int_t^{t+h} \ell(s, y_{t,x}^{\alpha[b],b}(s), \alpha[b](s), b(s))ds \right\}.$$

The last inequality holds for any $\alpha[\cdot] \in \Gamma$, hence

$$\frac{\delta h}{2} + v^\sharp(t,x) \leq \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \left\{ v^\sharp(t+h, y_{t,x}^{\alpha[b],b}(t+h)) + \int_t^{t+h} \ell(s, y_{t,x}^{\alpha[b],b}(s), \alpha[b](s), b(s))ds \right\} = v^\sharp(t,x),$$

which contradicts (2.23). Hence, we conclude that:

$$-\partial_t\phi(t,x) + H^+(t,x,D_x\phi(t,x)) \geq 0,$$

which means that $v^\sharp$ is a viscosity super-solution of (2.11).

Now, we will prove that $v^\sharp$ is a viscosity sub-solution of (2.11). Let ϕ be a function of class C^1 and (t,x) be a local maximum for $v^\sharp - \phi$ such that $(v^\sharp - \phi)(t,x) = 0$.

Moreover, suppose that there exists $\delta > 0$, such that $-\partial_t\phi(t,x) + H^+(t,x,D_x\phi(t,x)) = \delta$. From Lemma 2.4.4, we deduce the existence of a strategy $\alpha^*[\cdot] \in \Gamma$, such that for any $b(\cdot) \in \mathcal{B}$ and $h > 0$ small enough:

$$\phi(t+h, y_{t,x}^{\alpha^*[b],b}(t+h)) - \phi(t,x) + \int_t^{t+h} \ell(s, y_{t,x}^{\alpha^*[b],b}(s), \alpha^*[b](s), b(s))ds \leq -\frac{\delta h}{4}.$$

Since (t,x) is a local maximum of $v^\sharp - \phi$, the last inequality becomes:

$$\sup_{b(\cdot)\in\mathcal{B}} \left\{ \int_t^{t+h} \ell(s, y_{t,x}^{\alpha^*[b],b(s)}(s), \alpha^*[b](s), b(s))ds + v^\sharp(t+h, y_{t,x}^{\alpha^*[b],b}(t+h)) \right\} - v^\sharp(t,x) \leq -\frac{\delta h}{4},$$

which implies that

$$\inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \left\{ v^\sharp(t+h, y_{t,x}^{\alpha[b],b}(t+h)) + \int_t^{t+h} \ell(s, y_{t,x}^{\alpha[b],b}, \alpha[b](s), b(s))ds \right\} - v^\sharp(t,x) \leq -\frac{\delta h}{4} < 0.$$

From (2.23), the last inequality is impossible. Therefore

$$-\partial_t\phi(t,x) + H^+(t,x,D_x\phi(t,x)) \leq 0,$$

which means that $v^\sharp$ is a viscosity sub-solution of (2.11).

The uniqueness of $v^\sharp$ as a viscosity solution of 2.11 comes from the comparison Theorem 2.3.10. Indeed, let u be a viscosity solution of (2.11). Since u and $v^\sharp$ are respectively a sub and a super-solution of (2.11), we deduce from Theorem 2.3.10 that $u \leq v^\sharp$. Conversely, we get $v^\sharp \leq u$. □

Proof of Lemma 2.4.4. We set for $(s,z,a,b) \in [0,T] \times \mathbb{R}^d \times A \times B$,

$$G(s,z,a,b) := -\partial_t \phi(s,z) - \langle f(s,z,a,b), D_x\phi(s,z)\rangle - \ell(s,z,a,b).$$

We recall that $H^+(t,x,D_x\phi(t,x)) = \min_{b\in B}\max_{a\in A}\Big\{ - \langle f(t,x,a,b), D_x\phi(t,x)\rangle - \ell(t,x,a,b)\Big\}$. Henceforth, we obtain

$$\min_{b\in B}\max_{a\in A} G(t,x,a,b) = \delta.$$

Therefore, for any $b \in B$, there exists $a \in A$ (depending on b) such that:

$$G(t,x,a,b) \geq \delta.$$

On the other hand, $G(t,x,a,\cdot)$ is uniformly continuous since it is continuous over the compact set B. Thus, for any $b \in B$,

$$G(t,x,a,\bar{b}) \geq \frac{3\delta}{4}, \quad \forall\ \bar{b} \in \mathbb{B}(b,r)\bigcap B,$$

for some $r = r(b) > 0$. Moreover, since B is compact, there exist a finite number m of points $b_1,...,b_m \in B$ and radius $r_1,...,r_m > 0$ such that

$$B \subset \bigcup_{i=1}^{m} \mathbb{B}(b_i,r_i),$$

and for $a_i = a(b_i)$, $i \in \{1,...,m\}$,

$$G(t,x,a_i,\bar{b}) \geq \frac{3\delta}{4}, \quad \forall\ \bar{b} \in \mathbb{B}(b_i,r_i)\bigcap B.$$

Now, consider the mapping $\Lambda : B \to A$, such that for any $b \in B$, $\Lambda(b) := a_k$, where $1 \leq k \leq m$ is the smallest index verifying $b \in \mathbb{B}(b_k,r_k)$.

For any $b(\cdot) \in \mathcal{B}$, $\Lambda(b(\cdot))$ is measurable, thus we can define the strategy $\alpha^*[\cdot] \in \Gamma$ as follows:

$$\alpha^*[b](s) := \Lambda(b(s)).$$

By definition of Λ,

$$G(t,x,\Lambda(b),b) \geq \frac{3\delta}{4}, \quad \forall\ b \in B.$$

By continuity of G and the trajectory $y_{t,x}^{\alpha^*[b],b}$, for any $b(\cdot) \in \mathcal{B}$, there exists $h > 0$ small enough such that for any $b(\cdot) \in \mathcal{B}$:

$$G(s, y_{t,x}^{\alpha^*[b],b}(s), \alpha^*[b](s), b(s)) \geq \frac{\delta}{2}, \quad \forall s \in [t,t+h].$$

Finally, it is enough to integrate the last inequality between t and $t+h$ to get the desired result. □

After characterizing the value functions $v^\sharp$ and $v^\flat$ as unique viscosity solutions to Hamilton-Jacobi-Isaacs equations in Theorem 2.4.2 and by using the comparison principle Theorem 2.3.10, we can now compare the values of the two players.

Corollary 2.4.5. *In general, we have, for any* $(t,x) \in [0,T] \times \mathbb{R}^d$,

$$v^\sharp(t,x) \leq v^\flat(t,x).$$

Proof. Since $v^\flat$ is a viscosity solution of (2.9), it is also a super-solution of the same equation. Therefore

$$-\partial_t v^\flat + H^-(t, x, D_x v^\flat) \geq 0,$$

in the viscosity sens. Now since $H^+ \geq H^-$, we obtain:

$$-\partial_t v^\flat + H^+(t, x, D_x v^\flat) \geq 0,$$

which means that $v^\flat$ is a super-solution of (2.11). On the other hand, $v^\sharp$ is a sub-solution of (2.11). The comparison principle Theorem 2.3.10 concludes the result. □

In a similar way to Theorem 2.3.13, one can prove, in this context, the equality between $v^\flat$ and $v^\sharp$ if the Isaacs' condition (2.20) holds:

Corollary 2.4.6 (Value of the game)**.** *Assume that* **(H2.1)**, **(H2.2)**, **(H2.3)** *and condition* (2.20) *hold. Therefore, the game has a value* $v = v^\flat = v^\sharp$.

Proof. If the Isaacs' condition holds, (2.9) and (2.11) become reduced to the same HJ equation with an hamiltonian $H = H^+ = H^-$. In this case, $v^\sharp$ becomes a super-solution of equation (2.9). Since $v^\flat$ is the unique viscosity solution of (2.9), it is also a sub-solution of the same equation. From the comparison principle Theorem 2.3.10, we deduce that $v^\flat \leq v^\sharp$. Moreover, we know already from Corollary 2.4.5 that $v^\sharp \geq v^\flat$ which ends the proof. □

2.5 General comparison result

The aim of this part is to compare the different value functions v^-, v^+, $v^\sharp$ and $v^\flat$ defined is sections 2.3 and 2.4. This result is deduced essentially from the theorems of characterization of those different value functions and the comparison principle Theorem 2.3.10 for Hamilton-Jacobi equations.

Corollary 2.5.1. *We have the following order:*

$$v^\sharp \leq v^- \leq v^+ \leq v^\flat.$$

Proof. From Lemma 2.3.9, v^- is a super-solution of (2.11). Since $v^\sharp$ is the unique viscosity solution of (2.11), $v^\sharp$ is the smallest super-solution of this equation. Therefore, we have $v^\sharp \leq v^-$.
On the other hand, aging from Lemma 2.3.9, v^+ is a sub-solution of (2.9) whose unique viscosity solution is $v^\flat$. By the comparison theorem, we get $v^+ \leq v^\flat$. The fact that $v^- \leq v^+$ ends the proof. □

Remark 2.5.2. *The equality between all the value functions holds when the Isaacs' condition* (2.20) *is verified.*

2.6 Approximation by discrete time games and trajectory reconstruction

In this section, we present a method to reconstruct the optimal strategies and controls for both players. Without loss of generality, we will just focus on the problem of the first player (2.21). We extend the results presented in [11, Chapter VIII] for infinite time horizon differential game to our setting with finite time horizon. To this end, we will proceed as follow:

- First, we introduce a discrete time differential game with value function V_h and a reconstruction procedure based on the knowledge of V_h.

- Then, we introduce some regularity results of V_h and we show how to characterize it by means of a discrete dynamic programming principle.
- Next, we use the characterization of V_h to prove the optimality of the first player discrete strategy and the second player discrete control generated by our proposed reconstruction procedure for the discrete time differential game.
- After that, we prove that V_h is an approximation of $v^\sharp$. In particular, V_h converges uniformly, over compact subsets, to $v^\sharp$ when the time step h goes to zero.
- Finally, we construct an optimal strategy of the first player and an optimal control of the second player for the continuous time problem (2.21).

2.6.1 Approximation by discrete time games

Consider a uniform partition of $[0,T]$, with a time step $h = \frac{T}{N}$ (for $N \geq 1$): $s_k = kh$, $k = 0, ..., N$. Starting at time $t \in [s_k, s_{k+1}[$, for some $0 \leq k \leq N-1$ and from an initial state $x \in \mathbb{R}^d$, we define the following Euler forward scheme:

$$\begin{cases} \mathsf{y}_k = x, \\ \mathsf{y}_{k+1} = x + (s_{k+1} - t) f(t, x, \mathsf{a}_k, \mathsf{b}_k) \\ \mathsf{y}_{i+1} = \mathsf{y}_i + h f(s_i, \mathsf{y}_i, \mathsf{a}_i, \mathsf{b}_i), \quad i \geq k+1. \end{cases} \tag{2.25}$$

corresponding to two finite sequences of controls $(\mathsf{a}_i)_i \in A^{N-k}$ and $(\mathsf{b}_i)_i \in B^{N-k}$ of the first and the second players respectively. The solution of (2.25), representing the discrete trajectory of the system, will be denoted by $(\mathsf{y}_i)_{i=k}^N$.

Consider also an approximation of the cost functional J, denoted by J_h and given by:

$$J_h(t, x, \mathsf{a}, \mathsf{b}) := (s_{k+1} - t)\ell(t, x, \mathsf{a}_k, \mathsf{b}_k) + h \sum_{i=k+1}^{N-1} \ell(s_i, \mathsf{y}_i, \mathsf{a}_i, \mathsf{b}_i) + \Psi(\mathsf{y}_N),$$

for any $(\mathsf{a}_i)_i \in A^{N-k}$ and $(\mathsf{b}_i)_i \in B^{N-k}$.

Moreover, we follow the definition of discrete time nonanticpative strategies given in [11, Chapter VIII]. In fact, at each time step s_i, $0 \leq i \leq N-1$, the first player, and before choosing his action $\mathsf{a}_i \in A$, observes the choice of his opponent $\mathsf{b}_i \in B$. In other words, the second player must take his decision before the first player at each time step. The mathematical formulation of a discrete nonanticpative strategy for the first player is given as follows:

Definition 2.6.1. *A discrete nonanticpative strategy $\alpha_h[\cdot]$ of the first player is a mapping from B^N to A^N, such that for any $(\mathsf{b}_i)_i, (\mathsf{b}'_i)_i \in B^N$ and for any $0 \leq j \leq N-1$, if $\mathsf{b}_i = \mathsf{b}'_i$, $\forall\ i \leq j$, then $\alpha_h[\mathsf{b}]_i = \alpha_h[\mathsf{b}']_i$, $\forall i \leq j$.*

The set of discrete nonanticipative strategies of the first player will be denoted by Γ_h.

The discrete value function of the first player is defined, for $t \in [s_k, s_{k+1}[$ with $0 \leq k \leq N-1$ and $x \in \mathbb{R}^d$, by:

$$V_h(t, x) := \inf_{\alpha_h[\cdot] \in \Gamma_h} \sup_{(\mathsf{b}_i)_i \in B^{N-k}} J_h(t, x, \alpha_h[\mathsf{b}], \mathsf{b}). \tag{2.26}$$

Now, we will state the result concerning the convergence of V_h to $v^\sharp$.

Theorem 2.6.2. *The approximated value function V_h converges uniformly to $v^\sharp$ over compact subsets of $[0,T] \times \mathbb{R}^d$.*

In order to prove this Theorem, we will need the following Lemma from [11, Chapter VIII].

Lemma 2.6.3. *Let $u_\epsilon : E \to \mathbb{R}$, for a given set E, be an u.s.c and locally uniformly bounded function (respectively, l.s.c) and $\overline{u} := \limsup_{\epsilon\to 0} u_\epsilon$ (respectively, $\underline{u} := \liminf_{\epsilon\to 0} u_\epsilon$). Let ϕ be a function of class C^1 on E and $B_0 := \overline{B}(x^*, r) \bigcap E$ where $x^* \in B_0$ is a strict maximum (respectively, minimum) for $\overline{u} - \phi$ (respectively, $\underline{u} - \phi$) on B_0. Then there exist a sequence $(x_n)_n$ in B_0 and $(\epsilon_n)_n$ such that x_n is a maximum (respectively, minimum) for $u_{\epsilon_n} - \phi$ in B_0, $\epsilon_n > 0$ and*

$$\lim_{n\to\infty} \epsilon_n = 0, \quad \lim_{n\to\infty} x_n = x^*, \quad \lim_{n\to\infty} u_{\epsilon_n}(x_n) = \overline{u}(x^*) \quad \textit{(respectively, } \underline{u}(x^*)).$$

Proof of Theorem 2.6.2. First, let's define the function $\overline{v}$ by:

$$\overline{v}(t,x) := \limsup_{(s,y)\to(t,x), h\to 0^+} V_h(s,y),$$

and let's prove that $\overline{v}$ is a sub-solution of the HJ equation (2.11) whose unique viscosity solution is $v^\sharp$. Indeed, let ϕ be a function of class C^1 and (t,x) a strict maximum of $\overline{v} - \phi$ in $\overline{B} := \overline{B}((t,x), r)$, with a radius $r > 0$. By Lemma 2.6.3, there exist $(h_n)_n$ and $((t_n, x_n))_n \in \overline{B}$ such that (t_n, x_n) is the maximum of $V_{h_n} - \phi$ over $\overline{B}$ and

$$h_n \to 0, \quad (t_n, x_n) \to (t,x), \quad V_{h_n}(t_n, x_n) \to \overline{v}(t,x), \quad \text{when} \quad n \to \infty.$$

Now consider a uniform partition of $[0,T]$ with a time step h_n: $s_0^n = 0$,...,$s_k^n = kh_n$,... There exists $k \geq 0$ such that $t_n \in [s_k^n, s_{k+1}^n[$. We set $\tau_n := s_{k+1}^n - t_n > 0$.
Applying the discrete dynamic programming principle for V_{h_n} between t_n and s_{k+1}^n (relation (2.27) of Proposition 2.6.4) gives:

$$V_{h_n}(t_n, x_n) = \max_{b\in B} \min_{a\in A} \Big\{ \tau_n \ell(t_n, x_n, a, b) + V_{h_n}(s_{k+1}^n, x_n + \tau_n f(t, x_n, a, b)) \Big\}.$$

There exists $b_n \in B$ s.t. for any $a \in A$:

$$V_{h_n}(t_n, x_n) \leq \tau_n \ell(t_n, x_n, a, b_n) + V_{h_n}(s_{k+1}^n, x_n + \tau_n f(t, x_n, a, b_n)).$$

We set $y_n := x_n + \tau_n f(t, x, a, b_n)$. For h_n small enough, (s_{k+1}^n, y_n) will still in $\overline{B}$. Since (t_n, x_n) is the maximum of $V_{h_n} - \phi$ over $\overline{B}$ we have:

$$(V_{h_n} - \phi)(t_n, x_n) \geq (V_{h_n} - \phi)(s_{k+1}^n, y_n).$$

From the two above inequalities, we deduce for any $a \in A$:

$$\phi(t_n, x_n) - \phi(s_{k+1}^n, y_n) - \tau_n \ell(t_n, x_n, a, b_n) \leq 0.$$

The Taylor expansion of ϕ gives:

$$\phi(s_{k+1}^n, y_n) = \phi(t_n, x_n) + \tau_n \frac{\partial \phi(t_n, x_n)}{\partial t} + \langle D_x \phi(t_n, x_n), (y_n - x_n) \rangle + \sqrt{(s_{k+1}^n - t_n)^2 + \|y_n - x_n\|^2} \times \epsilon\Big((s_{k+1}^n, y_n) - (t_n, x_n)\Big),$$

where $\epsilon\Big((s_{k+1}^n, y_n) - (t_n, x_n)\Big) \to 0$ when $n \to +\infty$. From the last inequality and the Taylor expansion, and after dividing by τ_n, we get, for any $a \in A$:

$$-\frac{\partial \phi(t_n, x_n)}{\partial t} - \langle D_x \phi(t_n, x_n), f(t_n, x_n, a, b_n) \rangle - \ell(t_n, x_n, a, b_n) + \sqrt{1 + \|f(t_n, x_n, a, b_n)\|^2} \times \epsilon\Big((s_{k+1}^n, y_n) - (t_n, x_n)\Big) \leq 0.$$

Now, we can extract a sequence from $(b_n)_n$ which converges to $\bar{b} \in B$ and we let $n \to \infty$, we get for any $a \in A$:

$$-\frac{\partial \phi(t,x)}{\partial t} - \langle D_x \phi(t,x), f(t,x,a,\bar{b})\rangle - \ell(t,x,a,\bar{b}) \leq 0.$$

It follows that:

$$-\frac{\partial \phi(t,x)}{\partial t} + \max_{a\in A}\Big\{ - \langle D_x \phi(t,x), f(t,x,a,\bar{b})\rangle - \ell(t,x,a,\bar{b})\Big\} \leq 0,$$

therefore

$$-\frac{\partial \phi(t,x)}{\partial t} + \min_{b\in B}\max_{a\in A}\Big\{ - \langle D_x \phi(t,x), f(t,x,a,b)\rangle - \ell(t,x,a,b)\Big\} \leq 0,$$

which means that

$$-\frac{\partial \phi(t,x)}{\partial t} + H^+(t,x,D_x\phi(t,x)) \leq 0.$$

As a conclusion, $\overline{v}$ is a sub-solution of (2.11).
Now, consider the function $\underline{v}$ defined by:

$$\underline{v}(t,x) := \liminf_{(s,y)\to(t,x),h\to 0^+} V_h(s,y),$$

and the aim is to prove that $\underline{v}$ is a super-solution of (2.11). Let ϕ be a function of class C^1 and (t,x) a strict minimum of $\underline{v} - \phi$ in $\overline{B} := \overline{B}((t,x),r)$, with $r > 0$. Again by Lemma 2.6.3, there exist $(h_n)_n$ and $((t_n,x_n))_n \in \overline{B}$ such that (t_n,x_n) is the minimum of $V_{h_n} - \phi$ over $\overline{B}$ and when n goes to $+\infty$, we have

$$h_n \to 0, \quad (t_n,x_n) \to (t,x), \quad V_{h_n}(t_n,x_n) \to \underline{v}(t,x).$$

Moreover, there exists $k \geq 0$ such that $t_n \in [s_k^n, s_{k+1}^n[$. We set $\tau_n := s_{k+1}^n - t_n > 0$. From the discrete dynamic programming principle (2.27), we deduce that for any $b \in B$:

$$V_{h_n}(t_n,x_n) - \min_{a\in A}\Big\{\tau_n \ell(t_n,x_n,a,b) + V_h(s_{k+1}^n, x_n + \tau_n f(t_n,x_n,a,b))\Big\} \geq 0.$$

We fix $b \in B$, there exists $a_n := a(b) \in A$ (depending on b) such that:

$$V_{h_n}(t_n,x_n) - \Big(\tau_n \ell(t_n,x_n,a_n,b) + V_{h_n}(s_{k+1}^n, x_n + \tau_n f(t_n,x_n,a_n,b))\Big) \geq 0.$$

We set now $y_n := x_n + \tau_n f(t_n,x,a_n,b)$. For h_n small enough, (s_{k+1}^n, y_n) will still in $\overline{B}$. Since (t_n,x_n) is the minimum of $V_{h_n} - \phi$ over $\overline{B}$, we obtain

$$(V_{h_n} - \phi)(t_n,x_n) \leq (V_{h_n} - \phi)(s_{k+1}^n, y_n).$$

From the two above inequalities, we deduce that:

$$\phi(t_n,x_n) - \phi(s_{k+1}^n, y_n) - \tau_n \ell(t_n,x_n,a_n,b) \geq 0.$$

The Taylor expansion of ϕ gives:

$$\phi(s_{k+1}^n, y_n) = \phi(t_n,x_n) + \tau_n \frac{\partial \phi(t_n,x_n)}{\partial t} + \langle D_x\phi(t_n,x_n), (y_n - x_n)\rangle + \sqrt{(s_{k+1}^n - t_n)^2 + \|y_n - x_n\|^2} \times \epsilon\Big((s_{k+1}^n, y_n) - (t_n,x_n)\Big),$$

where $\epsilon\Big((s_{k+1}^n, y_n) - (t_n, x_n)\Big) \to 0$ when $n \to +\infty$. From the last inequality and the Taylor expansion and after dividing by τ_n, we obtain:

$$-\frac{\partial\phi(t_n,x_n)}{\partial t} - \langle D_x\phi(t_n,x_n), f(t_n,x_n,a_n,b)\rangle - \ell(t_n,x_n,a_n,b) \\ + \sqrt{1+\|f(t_n,x_n,a_n,b)\|^2} \times \epsilon\Big((s_{k+1}^n, y_n) - (t_n,x_n)\Big) \geq 0.$$

Now, we can extract a sequence from $(a_n)_n$ which converges to $\overline{a} \in A$ and we let $n \to \infty$, we get:

$$-\frac{\partial\phi(t,x)}{\partial t} - \langle D_x\phi(t,x), f(t,x,\overline{a},b)\rangle - \ell(t,x,\overline{a},b) \geq 0,$$

henceforth

$$-\frac{\partial\phi(t,x)}{\partial t} + \max_{a\in A}\Big\{ - \langle D_x\phi(t,x), f(t,x,a,b)\rangle - \ell(t,x,a,b)\Big\} \geq 0,$$

therefore

$$-\frac{\partial\phi(t,x)}{\partial t} + \min_{b\in B}\max_{a\in A}\Big\{ - \langle D_x\phi(t,x), f(t,x,a,b)\rangle - \ell(t,x,a,b)\Big\} \geq 0,$$

which means that

$$-\frac{\partial\phi(t,x)}{\partial t} + H^+(t,x,D_x\phi(t,x)) \geq 0.$$

As a conclusion, $\underline{v}$ is a super-solution of (2.11).
To conclude, we have shown that $\overline{v}$ and $\underline{v}$ are respectively a sub and a super-solution of (2.11) which has $v^\sharp$ as a unique solution in the viscosity sense. Therefore, by applying the comparison principle Theorem 2.3.10 , we obtain $\overline{v} \leq v^\sharp \leq \underline{v}$. Since we have already $\underline{v} \leq \overline{v}$, we get $\underline{v} = v^\sharp = \overline{v}$. Finally, by exploiting the properties of weak limits, see for instance [11, Chapter V], we deduce the uniform convergence of V_h to $v^\sharp$ over compact subsets of $[0,T]\times\mathbb{R}^d$.

□

2.6.2 Trajectory reconstruction

Now, we present how to synthesize an optimal strategy for the first player and an optimal control for the second player in feedback form using the value function V_h, see Algorithm 2.1. Recall that an optimal choice of the second player corresponds to the worst case that can occur for the first player. In order to prove the optimality of the first player discrete strategy and the second player discrete control generated by Algorithm 2.1, V_h should verify a discrete dynamic programming principle. This is the purpose of the following Proposition which starts with a regularity result on V_h needed to prove the discrete dynamic programming principle.

Proposition 2.6.4. *Assume* **(H2.1)**, **(H2.2)** *and* **(H2.3)**, *then:*

(i) V_h *is Lipschitz continuous w.r.t. the space variable uniformly in the time variable.*

(ii) *For* $t \in [s_k, s_{k+1}[$, *with* $0 \leq k \leq N-1$, *and* $x \in \mathbb{R}^d$, *we have:*

$$V_h(t,x) = \max_{b\in B}\min_{a\in A}\Big\{(s_{k+1}-t)\ell(t,x,a,b) + V_h(s_{k+1}, x + (s_{k+1}-t)f(t,x,a,b))\Big\}. \tag{2.27}$$

Proof. (i) We fix $t \in [0,T[$ and let $0 \leq k \leq N-1$ such that $t \in [s_k, s_{k+1}[$ and x, $y \in \mathbb{R}^d$. For $\epsilon > 0$, there exists $\alpha_h^\epsilon[\cdot] \in \Gamma_h$ such that:

$$V_h(t,y) \geq \sup_{(\mathsf{b}_i)_i \in B^{N-k}} J_h(t,y,\alpha_h^\epsilon[\mathsf{b}],\mathsf{b}) - \frac{\epsilon}{2}.$$

Algorithm 2.1: Worst case

1: Initialize $\mathsf{y}_k^* = x$ with $0 \leq k \leq N-1$.

2: **for** $i = k$ to $N-1$ **do**

3: The optimal choice of the second player $\mathsf{b}_i^* \in B$ is defined by:

$$\mathsf{b}_i^* \in \begin{cases} \underset{b\in B}{\operatorname{argmax}} \underset{a\in A}{\min} \Big\{\tau\ell(t,x,a,b) + V_h(s_{k+1}, x+\tau f(t,x,a,b))\Big\}, & \text{if} \quad i=k, \\ \underset{b\in B}{\operatorname{argmax}} \underset{a\in A}{\min} \Big\{h\ell(s_i,\mathsf{y}_i^*,a,b) + V_h(s_{i+1}, \mathsf{y}_i^* + hf(s_i,\mathsf{y}_i^*,a,b))\Big\}, & \text{else}, \end{cases}$$

where $\tau := (s_{k+1} - t)$.

4: The optimal reaction of the first player $\alpha_h^*[\mathsf{b}^*]_i := \mathsf{a}_i^* \in A$ is given by:

$$\mathsf{a}_i^* \in \begin{cases} \underset{a\in A}{\operatorname{argmin}} \Big\{\tau\ell(t,x,a,\mathsf{b}_k^*) + V_h(s_{k+1}, x+\tau f(t,x,a,\mathsf{b}_k^*))\Big\}, & \text{if} \quad i=k, \\ \underset{a\in A}{\operatorname{argmin}} \Big\{h\ell(s_i,\mathsf{y}_i^*,a,\mathsf{b}_i^*) + V_h(s_{i+1}, \mathsf{y}_i^* + hf(s_i,\mathsf{y}_i^*,a,\mathsf{b}_i^*))\Big\}, & \text{else}. \end{cases}$$

5: The new state position y_{i+1}^* is given by:

$$\mathsf{y}_{i+1}^* = \begin{cases} x + \tau f(t,x,\mathsf{a}_k^*,\mathsf{b}_k^*), & \text{if} \quad i=k, \\ \mathsf{y}_i^* + hf(s_i,\mathsf{y}_i^*,\mathsf{a}_i^*,\mathsf{b}_i^*), & \text{else}. \end{cases}$$

6: **end for**

7: **return** A discrete optimal strategy $\alpha_h^*[\cdot]$ of the first player, a discrete optimal control $(\mathsf{b}_i^*)_i$ of the second player and a discrete optimal trajectory $(\mathsf{y}_i^*)_i$.

On the other hand, there exists $(\mathsf{b}_i^\epsilon)_i \in B^{N-k}$ such that:

$$V_h(t,x) \leq \sup_{(\mathsf{b}_i)_i \in B^{N-k}} J_h(t,x,\alpha_h^\epsilon[\mathsf{b}],\mathsf{b}) \leq J_h(t,x,\alpha_h^\epsilon[\mathsf{b}^\epsilon],\mathsf{b}^\epsilon) + \frac{\epsilon}{2}.$$

Denote by $(\mathsf{a}_i^\epsilon)_i := \alpha_h^\epsilon[\mathsf{b}^\epsilon] \in A^{N-k}$. From the first inequality involving $V_h(t,y)$, we deduce that:

$$V_h(t,y) \geq J_h(t,y,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon) - \frac{\epsilon}{2}.$$

Now from the two above inequalities, we get:

$$V_h(t,x) - V_h(t,y) \leq J_h(t,x,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon) - J_h(t,y,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon) + \epsilon.$$

Denote by $(\mathsf{y}_i^x)_i$ and $(\mathsf{y}_i^y)_i$ the solutions of (2.25) corresponding to $((\mathsf{a}_i^\epsilon)_i, (\mathsf{b}_i^\epsilon)_i) \in A^{N-k} \times B^{N-k}$ and starting respectively from x and y. We have the following estimation:

$$\begin{aligned} |J_h(t,x,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon) - J_h(t,y,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon)| \leq (s_{k+1}-t)|\ell(t,x,\mathsf{a}_k^\epsilon,\mathsf{b}_k^\epsilon) - \ell(t,y,\mathsf{a}_k^\epsilon,\mathsf{b}_k^\epsilon)| \\ + h\sum_{i=k+1}^{N-1} |\ell(s_i,\mathsf{y}_i^x,\mathsf{a}_i^\epsilon,\mathsf{b}_i^\epsilon) - \ell(s_i,\mathsf{y}_i^y,\mathsf{a}_i^\epsilon,\mathsf{b}_i^\epsilon)| + |\Psi(\mathsf{y}_N^x) - \Psi(\mathsf{y}_N^y)|. \end{aligned}$$

Now we use the Lipschitz continuity of ℓ and Ψ to get:

$$|J_h(t,x,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon) - J_h(t,y,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon)| \leq hL_2\|x-y\| + hL_2\sum_{i=k+1}^{N-1} \|\mathsf{y}_i^x - \mathsf{y}_i^y\| + L_3\|\mathsf{y}_N^x - \mathsf{y}_N^y\|,$$

where L_2 and L_3 are the Lipschitz constants of ℓ and Ψ respectively.
By the Lipschitz continuity of f, with Lipschitz constant L_1, we get the following estimation:

$$\|\mathsf{y}_i^x - \mathsf{y}_i^y\| \leq (1 + hL_1)^{i-k}\|x - y\|, \quad i \geq k+1,$$

which implies that:

$$\sum_{i=k+1}^{N-1} \|\mathsf{y}_i^x - \mathsf{y}_i^y\| \leq \frac{e^{TL_1}}{hL_1}\|x - y\| \quad \text{and} \quad \|\mathsf{y}_N^x - \mathsf{y}_N^y\| \leq e^{TL_1}\|x - y\|.$$

Combining the above estimations implies the existence of some constant $C > 0$ such that:

$$|V_h(t,x) - V_h(t,y)| \leq C\|x - y\| + \epsilon.$$

The fact that ϵ is chosen arbitrarily ends the proof (i).

(ii) Now we prove the discrete dynamic programming principle. We set, for $t \in [s_k, s_{k+1}[$ and $x \in \mathbb{R}^d$,

$$u(t,x) := \min_{b \in B} \max_{a \in A} \Big\{(s_{k+1} - t)\ell(t,x,a,b) + V_h(s_{k+1}, x + (s_{k+1} - t)f(t,x,a,b))\Big\}.$$

We start by showing that $V_h(t,x) \leq u(t,x)$. For any $b \in B$, there exists $\overline{a}(b) \in A$ (depending on b) such that:

$$\begin{aligned} u(t,x) &\geq \min_{a \in A} \Big\{(s_{k+1} - t)\ell(t,x,a,b) + V_h(s_{k+1}, x + (s_{k+1} - t)f(t,x,a,b))\Big\} \\ &= (s_{k+1} - t)\ell(t,x,\overline{a}(b),b) + V_h(s_{k+1}, x + (s_{k+1} - t)f(t,x,\overline{a}(b),b)). \end{aligned}$$

Let $(\mathsf{b}_i)_i \in B^{N-k}$ be a sequence of actions of the second player and let's define:

$$x' := x + (s_{k+1} - t)f(t,x,\overline{a}(\mathsf{b}_k), \mathsf{b}_k).$$

For $\epsilon > 0$, let $\alpha_h^\epsilon[\cdot] \in \Gamma_h$ verifying:

$$\epsilon + V_h(s_{k+1}, x') \geq \sup_{(\mathsf{b}_i')_i \in B^{N-k-1}} J_h(s_{k+1}, x', \alpha_h^\epsilon[\mathsf{b}'], \mathsf{b}').$$

Now let's define $\delta_h[\cdot] \in \Gamma_h$ such that, for any $(\mathsf{b}_i)_i \in B^{N-k}$,

$$\delta_h[\mathsf{b}]_i := \begin{cases} \overline{a}(\mathsf{b}_k), & \text{if} \quad i = k \\ \alpha_h^\epsilon[\mathsf{b}]_i, & \text{if} \quad i \geq k+1. \end{cases}$$

For any $(\mathsf{b}_i)_i \in B^{N-k}$, we have

$$\begin{aligned} J_h(t,x,\delta_h[\mathsf{b}],\mathsf{b}) &= (s_{k+1} - t)\ell(t,x,\overline{a}(\mathsf{b}_k),\mathsf{b}_k) + h\sum_{i=k+1}^{N-1} \ell(s_i, \mathsf{y}_i, \delta_h[\mathsf{b}]_i, \mathsf{b}_i) + \Psi(\mathsf{y}_N) \\ &= (s_{k+1} - t)\ell(t,x,\overline{a}(\mathsf{b}_k),\mathsf{b}_k) + J_h(s_{k+1}, x', \alpha_h^\epsilon[\mathsf{b}'], \mathsf{b}') \end{aligned}$$

where $(\mathsf{b}_i')_i \in B^{N-k-1}$ is the restriction of $(\mathsf{b}_i)_i$ for $i \geq k+1$. Therefore we get

$$\begin{aligned} V_h(t,x) &\leq \sup_{(\mathsf{b}_i)_i \in B^{N-k}} J_h(t,x,\delta_h[\mathsf{b}],\mathsf{b}) \\ &\leq \sup_{(\mathsf{b}_i)_i \in B^{N-k}} \Big\{(s_{k+1} - t)\ell(t,x,\overline{a}(\mathsf{b}_k),\mathsf{b}_k) + J_h(s_{k+1}, x', \alpha_h^\epsilon[\mathsf{b}'], \mathsf{b}')\Big\} \\ &\leq \sup_{\mathsf{b}_k \in B} \Big\{(s_{k+1} - t)\ell(t,x,\overline{a}(\mathsf{b}_k),\mathsf{b}_k) + \sup_{(\mathsf{b}_i')_i \in B^{N-k-1}} J_h(s_{k+1}, x', \alpha_h^\epsilon[\mathsf{b}'], \mathsf{b}')\Big\} \\ &\leq \sup_{\mathsf{b}_k \in B} \{\epsilon + (s_{k+1} - t)\ell(t,x,\overline{a}(\mathsf{b}_k),\mathsf{b}_k) + V_h(s_{k+1}, x')\}. \end{aligned}$$

On the other hand, we have

$$u(t,x) = \max_{b\in B}\min_{a\in A}\Big\{(s_{k+1}-t)\ell(t,x,a,b) + V_h(s_{k+1}, x+(s_{k+1}-t)f(t,x,a,b))\Big\},$$

and

$$\max_{b\in B}\min_{a\in A}\Big\{(s_{k+1}-t)\ell(t,x,a,b) + V_h(s_{k+1}, x+(s_{k+1}-t)f(t,x,a,b))\Big\}$$
$$= \max_{b\in B}\Big\{(s_{k+1}-t)\ell(t,x,\overline{a}(b),b) + V_h(s_{k+1},x')\Big\}.$$

Combining the above estimations gives $V_h(t,x) \le \epsilon + u(t,x)$, for any $\epsilon > 0$, which implies that $V_h(t,x) \le u(t,x)$.

Now, we will prove the converse inequality. Let $\overline{b} \in B$, such that

$$u(t,x) = \min_{a\in A}\Big\{(s_{k+1}-t)\ell(t,x,a,\overline{b}) + V_h(s_{k+1}, x+(s_{k+1}-t)f(t,x,a,\overline{b}))\Big\}.$$

For any discrete control of the second player, $(\mathsf{b}_i)_i \in B^{N-k}$, we define the following control sequence $(\hat{\mathsf{b}}_i)_i \in B^{N-k}$:

$$\hat{\mathsf{b}}_i := \begin{cases} \overline{b}, & \text{if} \quad i = k, \\ \mathsf{b}_i, & \text{else.} \end{cases}$$

For $\epsilon > 0$, consider an ϵ-optimal discrete nonanticpative strategy $\alpha_h^\epsilon[\cdot] \in \Gamma_h$ for $V_h(t,x)$ i.e.

$$\epsilon + V_h(t,x) \ge \sup_{(\mathsf{b}_i)_i\in B^{N-k}} J_h(t,x,\alpha_h^\epsilon[\mathsf{b}],\mathsf{b}). \tag{2.28}$$

Since $\alpha_h^\epsilon[\cdot]$ is a nonanticipative strategy, we deduce from Definition 2.6.1 that $\alpha_h^\epsilon[\hat{\mathsf{b}}]_k$ depends only on $\overline{b}$. Now let $\overline{a} := \alpha_h^\epsilon[\hat{\mathsf{b}}]_k$ and define the system state x' at time step s_{k+1}, corresponding to $\overline{a}$ and $\overline{b}$, by $x' := x + (s_{k+1}-t)f(t,x,\overline{a},\overline{b})$. Moreover, we define the discrete nonanticipative strategy $\delta_h[\cdot] \in \Gamma_h$ starting from s_{k+1}, for $(\mathsf{b}_i)_i \in B^{N-k-1}$, by

$$\delta_h[\mathsf{b}]_i := \alpha_h^\epsilon[\hat{\mathsf{b}}]_i, \quad \text{for} \quad i \ge k+1.$$

From the definition of $\delta_h[\cdot]$, we get

$$\begin{aligned} V_h(s_{k+1},x') &= \inf_{\alpha_h[\cdot]\in\Gamma_h}\sup_{(\mathsf{b}_i)_i\in B^{N-k-1}} J_h(s_{k+1},x',\alpha_h[\mathsf{b}],\mathsf{b}) \\ &\le \sup_{(\mathsf{b}_i)_i\in B^{N-k-1}} J_h(s_{k+1},x',\delta_h[\mathsf{b}],\mathsf{b}) \\ &\le J_h(s_{k+1},x',\delta_h[\mathsf{b}^\epsilon],\mathsf{b}^\epsilon) + \epsilon \end{aligned} \tag{2.29}$$

where $(\mathsf{b}_i^\epsilon)_i \in B^{N-k-1}$ is an ϵ-optimal for the term $\sup_{(\mathsf{b}_i)_i\in B^{N-k-1}} J_h(s_{k+1},x',\delta_h[\mathsf{b}],\mathsf{b})$.

Claim that $u(t,x) \le J_h(t,x,\alpha_h^\epsilon[\hat{\mathsf{b}}^\epsilon],\hat{\mathsf{b}}^\epsilon) + \epsilon$, where $(\hat{\mathsf{b}}_i^\epsilon)_i \in B^{N-k}$ is defined as follows:

$$\hat{\mathsf{b}}_i^\epsilon := \begin{cases} \overline{b}, & \text{if} \quad i = k, \\ \mathsf{b}_i^\epsilon, & \text{if} \quad i \ge k+1. \end{cases}$$

From the above claim, we deduce that

$$u(t,x) \le \sup_{(\mathsf{b}_i)_i\in B^{N-k}} J_h(t,x,\alpha_h^\epsilon[\mathsf{b}],\mathsf{b}) + \epsilon.$$

Together with (2.28), we get $u(t,x) \leq V_h(t,x) + 2\epsilon$, for any $\epsilon > 0$ which ends the proof. Now let's prove the above claim.

$$\begin{aligned} u(t,x) &= \max_{b\in B} \min_{a\in A} \Big\{(s_{k+1}-t)\ell(t,x,a,b) + V_h(s_{k+1}, x + (s_{k+1}-t)f(t,x,a,b))\Big\} \\ &= \min_{a\in A} \Big\{(s_{k+1}-t)\ell(t,x,a,\bar{b}) + V_h(s_{k+1}, x + (s_{k+1}-t)f(t,x,a,\bar{b}))\Big\} \\ &\leq (s_{k+1}-t)\ell(t,x,\bar{a},\bar{b}) + V_h(s_{k+1}, x') \\ &\leq (s_{k+1}-t)\ell(t,x,\bar{a},\bar{b}) + J_h(s_{k+1}, x', \delta_h[\mathsf{b}^\epsilon], \mathsf{b}^\epsilon) + \epsilon, \end{aligned}$$

where the last inequality holds from (2.29). Since. $\bar{a} = \alpha_h^\epsilon[\hat{\mathsf{b}}^\epsilon]_k$, $\hat{\mathsf{b}}_k^\epsilon = \bar{b}$ and $\delta_h[\mathsf{b}^\epsilon]_i = \alpha_h^\epsilon[\hat{\mathsf{b}}^\epsilon]_i$, for any $i \geq k+1$, we obtain the following equality which ends the proof of the claim:

$$(s_{k+1}-t)\ell(t,x,\bar{a},\bar{b}) + J_h(s_{k+1}, x', \delta_h[\mathsf{b}^\epsilon], \mathsf{b}^\epsilon) = J_h(t,x,\alpha_h^\epsilon[\hat{\mathsf{b}}^\epsilon], \hat{\mathsf{b}}^\epsilon).$$

□

Thanks to the results given in Proposition 2.6.4, we deduce an additional result on the regularity of V_h.

Corollary 2.6.5. *V_h is locally Lipschitz continuous w.r.t. the time variable uniformly in the space variable.*

Proof. Let $x \in \mathbb{R}^d$ and $t \in [0,T[$ be fixed and take $s \in [0,T]$ close enough to t. We can distinguish two cases:

- Suppose that $t \in]s_k, s_{k+1}[$ for some $k \in \{0,...,N-1\}$. In this case, $s \in]s_k, s_{k+1}[$ and from (2.27) we can write:

$$\begin{aligned} |V_h(s,x) - V_h(t,x)| \leq \max_{b\in B} \min_{a\in A} \Big\{ &|(s_{k+1}-s)\ell(s,x,a,b) - (s_{k+1}-t)\ell(t,x,a,b)| \\ &+ |V_h(s_{k+1}, x^s) - V_h(s_{k+1}, x^t)|\Big\}, \end{aligned}$$

  where $x^s := x + (s_{k+1}-s)f(s,x,a,b)$ and $x^t := x + (s_{k+1}-t)f(t,x,a,b)$.
  Since ℓ is continuous w.r.t. to all its arguments and $[0,T]$, A and B are compact sets, $(\tau,a,b) \mapsto \ell(\tau,x,a,b)$ is bounded by some constant M. Using also the Lipschitz continuity of ℓ, we get:

$$|t\ell(t,x,a,b) - s\ell(s,x,a,b)| \leq t|\ell(t,x,a,b) - \ell(s,x,a,b)| + |t-s||\ell(s,x,a,b)| \leq (tL_2 + M)|t-s|,$$

  where L_2 is the Lipschitz constant of ℓ. Henceforth, we obtain:

$$|(s_{k+1}-s)\ell(s,x,a,b) - (s_{k+1}-t)\ell(t,x,a,b)| \leq s_{k+1}L_2|t-s| + (tL_2+M)|t-s| \leq (2TL_2+M)|t-s|.$$

  Furthermore, since f is continuous w.r.t. to all its arguments and $[0,T]$, A and B are compact sets, $(\tau,a,b) \mapsto f(\tau,x,a,b)$ is bounded. By the Lipschitz continuity of f and similar computations as we did above for ℓ, we can deduce the existence of some real constant $c_1 > 0$ such that the following estimation holds:

$$\|x^s - x^t\| \leq c_1|t-s|.$$

  From Proposition 2.6.4, V_h is Lipschitz w.r.t. the space variable together with the last inequality, we deduce the existence of a real constant $c_2 > 0$ such that:

$$|V_h(s_{k+1}, x^s) - V_h(s_{k+1}, x^t)| \leq c_2|t-s|.$$

  Combining the above inequalities implies the existence of some real constant $c > 0$ such that:

$$|V_h(s,x) - V_h(t,x)| \leq \max_{b\in B} \min_{a\in A} c|t-s| = c|t-s|.$$

- Now suppose that $t = s_k$ for some $k \in \{0, ..., N-1\}$. First consider the case where $s \in [s_k, s_k + \theta[$ with $\theta > 0$. Again, we can write:

$$|V_h(s,x) - V_h(t,x)| = |V_h(s,x) - V_h(s_k,x)| \leq \max_{b\in B} \max_{a\in A} \Big\{ |(s_{k+1} - s)\ell(s,x,a,b) \\ -(s_{k+1} - s_k)\ell(s_k,x,a,b)| \\ +|V_h(s_{k+1}, x^s) - V_h(s_{k+1}, x_{k+1})| \Big\},$$

where x^s is defined as above and $x_{k+1} := x + hf(t,x,a,b)$. In a similar way to the first case, we obtain the desired estimation i.e. $|V_h(s,x) - V_h(s_k,x)| \leq c|t-s|$.

Now, consider the other case where $s \in]s_k - \theta, s_k]$. By the discrete dynamic programming principle (2.27), $V_h(s,x)$ can be written as follows:

$$V_h(s,x) = \max_{b\in B} \min_{a\in A} \Big\{ (s_k - s)\ell(s,x,a,b) + V_h(s_k, x + (s_k - s)f(s,x,a,b)) \Big\}.$$

Therefore, we can write:

$$|V_h(s,x) - V_h(s_k,x)| = \max_{b\in B} \min_{a\in A} \Big\{ (s_k - s)|\ell(s,x,a,b)| + |V_h(s_k, x + (s_k - s)f(s,x,a,b)) - V_h(s_k,x) \Big\}.$$

From Proposition 2.6.4, V_h is Lipschitz w.r.t. the space variable. Hence there exists some real constant $c >$ such that:

$$|V_h(s_k, x + (s_k - s)f(s,x,a,b)) - V_h(s_k,x)| \leq c\|x + (s_k - s)f(s,x,a,b) - x\| \\ \leq c|s_k - s| \times \|f(s,x,a,b)\| \leq cM'|s_k - s|,$$

since $(\tau, a, b) \mapsto f(\tau, x, a, b)$ is bounded by some real constant $M' > 0$.

Moreover, $(\tau, a, b) \mapsto \ell(\tau, x, a, b)$ is bounded by another real constant $M > 0$. Therefore, $|(s_k - s)\ell(s,x,a,b)| \leq M|s_k - s|$. As a conclusion, we get:

$$|V_h(s,x) - V_h(s_k,x)| \leq \max_{b\in B} \min_{a\in A} (cM' + M)|s_k - s| = (cM' + M)|s_k - s|.$$

□

Corollary 2.6.6. *V_h is the unique solution of the discrete dynamic programming equation (2.27) with a terminal condition of type $V_h(T,x) = \Psi(x)$, for any $x \in \mathbb{R}^d$.*

Proof. Consider two continuous functions u and w that satisfy the discrete dynamic programming equation (2.27) and $u(T,x) = w(T,x)$, for any $x \in \mathbb{R}^d$.

We claim that $u \leq w$. Indeed, let $(t,x) \in [s_k, s_{k+1}[\times\mathbb{R}^d$, for $k \in \{0, ..., N-1\}$. By continuity of u and f and by compactness of B, there exists $b_0 \in B$ such that for any $a \in A$:

$$u(t,x) \leq (s_{k+1} - t)\ell(t,x,a,b_0) + u(s_{k+1}, x + (s_{k+1} - t)f(t,x,a,b_0)).$$

Moreover, by continuity of w and f and by compactness of A, there exists $a_0 \in A$ such that:

$$w(t,x) \geq \min_{a\in A} \Big\{ (s_{k+1} - t)\ell(t,x,a,b_0) + w(s_{k+1}, x + (s_{k+1} - t)f(t,x,a,b_0)) \Big\} \\ \geq (s_{k+1} - t)\ell(t,x,a_0,b_0) + w(s_{k+1}, x + (s_{k+1} - t)f(t,x,a_0,b_0))$$

From the two above inequalities, we deduce that:

$$u(t,x) - w(t,x) \leq u(s_{k+1}, x_{k+1}) - w(s_{k+1}, x_{k+1}), \quad \text{with} \quad x_{k+1} := x + (s_{k+1} - t)f(t,x,a_0,b_0).$$

By the same arguments, we get

$$u(s_{k+1},x_{k+1})-w(s_{k+1},x_{k+1})\leq u(s_{k+2},x_{k+2})-w(s_{k+2},x_{k+2})\leq ...\leq u(s_N,x_N)-w(s_N,x_N),$$

for some x_{k+2},...,$x_N\in\mathbb{R}^d$. The fact that $u(s_N,x_N)-w(s_N,x_N)=u(T,x_N)-w(T,x_N)=0$ ends the proof of the claim and in a similar way, we get $u\geq w$ which gives the desired result.

□

The following Proposition concerns the optimality of $(\alpha_h^*[\cdot],(\mathsf{b}_i^*)_i)\in\Gamma_h\times B^N$, generated by Algorithm 2.1 for the discrete time differential game (2.26).

Proposition 2.6.7. *Let $(t,x)\in[s_k,s_{k+1}[\times\mathbb{R}^d$, for $k\in\{0,...,N-1\}$. For any $(\mathsf{b}_i)_i\in B^{N-k}$, we have:*

$$J_h(t,x,\alpha_h^*[\mathsf{b}],\mathsf{b})\leq V_h(t,x),$$

and the equality holds when $(\mathsf{b}_i)_i=(\mathsf{b}_i^*)_i$.

Proof. From Proposition 2.6.4, we have:

$$V_h(t,x)=\max_{b\in B}\min_{a\in A}\Big\{(s_{k+1}-t)\ell(t,x,a,b)+V_h(s_{k+1},x+(s_{k+1}-t)f(t,x,a,b))\Big\}.$$

Through the instructions of Algorithm 2.1, we deduce:

$$\begin{aligned}V_h(t,x)&=\min_{a\in A}\Big\{(s_{k+1}-t)\ell(t,x,a,\mathsf{b}_k^*)+V_h(s_{k+1},x+(s_{k+1}-t)f(t,x,a,\mathsf{b}_k^*))\Big\}\\&=(s_{k+1}-t)\ell(t,x,\mathsf{a}_k^*,\mathsf{b}_k^*)+V_h(s_{k+1},\mathsf{y}_{k+1}^*).\end{aligned}$$

By the same way we get

$$V_h(s_{k+1},\mathsf{y}_{k+1}^*)=h\ell(s_{k+1},\mathsf{y}_{k+1}^*,\mathsf{a}_{k+1}^*,\mathsf{b}_{k+1}^*)+V_h(s_{k+2},\mathsf{y}_{k+2}^*),$$

and in general, we have for any $i\in\{k+1,...,N-1\}$:

$$V_h(s_i,\mathsf{y}_i^*)=h\ell(s_i,\mathsf{y}_i^*,\mathsf{a}_i^*,\mathsf{b}_i^*)+V_h(s_{i+1},\mathsf{y}_{i+1}^*).$$

At the last time step $s_N=T$, we have $V_h(s_N,\mathsf{y}_N^*)=\Psi(\mathsf{y}_N^*)$. Therefore we conclude:

$$\begin{aligned}V_h(t,x)&=(s_{k+1}-t)\ell(t,x,\mathsf{a}_k^*,\mathsf{b}_k^*)+h\sum_{i=k+1}^{N-1}\ell(s_i,\mathsf{y}_i^*,\mathsf{a}_i^*,\mathsf{b}_i^*)+\Psi(\mathsf{y}_N^*)\\&=J_h(t,x,\alpha_h^*[\mathsf{b}^*],\mathsf{b}^*).\end{aligned}$$

In a similar way, one can prove that for any $(\mathsf{b}_i)_i\in B^{N-k}$:

$$J_h(t,x,\alpha_h^*[\mathsf{b}],\mathsf{b})\leq V_h(t,x).$$

□

Remark 2.6.8. *For $(t,x)\in[0,T]\times\mathbb{R}^d$, the couple $(\alpha_h^*[\cdot],(\mathsf{b}_i^*)_i)$, generated by Algorithm 2.1, constitutes a Nash equilibrium for the discrete game (2.26) in the terminology of the theory of noncooperative games. This means that every player cannot improve his guaranteed outcome, given by $V_h(t,x)$, by any unilateral deviation from his optimal choice, $\alpha_h^*[\cdot]$ for the first player and $(\mathsf{b}_i^*)_i$ for the second player.*

In order to conclude this part, we present the following result whose aim is to define a strategy for the first player and a control for the second player, such that when the time step h goes to zero, the corresponding value of J converges to the first player value function $v^\sharp$.

Theorem 2.6.9. *Let $(t,x) \in [0,T[\times\mathbb{R}^d$, there exist $(\tilde{\alpha}_h^*[\cdot], b_h^*(\cdot)) \in \Gamma \times \mathcal{B}$ verifying*

$$\lim_{h\to 0^+} J(t,x,\tilde{\alpha}_h^*[b_h^*], b_h^*) = v^\sharp(t,x).$$

Proof. We introduce a new subset of $\mathcal{B}$ denoted by $\mathcal{B}_h$ and defined by

$$\mathcal{B}_h := \Big\{ b(\cdot) \in \mathcal{B} \;\mid\; b(s) = b(s_i), \quad \forall s \in [s_i, s_{i+1}[, \quad \text{for} \quad i = 0, ..., N-1 \Big\}.$$

$\mathcal{A}_h$ is the subset of $\mathcal{A}$ defined in a similar way to $\mathcal{B}_h$.
Let $(t,x) \in [0,T[\times\mathbb{R}^d$ and $k \in \{0, ..., N-1\Big\}$ such that $t \in [s_k, s_{k+1}[$ and consider $(\alpha_h^*[\cdot], (\mathsf{b}_i^*)_i) \in \Gamma_h \times B^{N-k}$, generated by Algorithm 2.1 and $(\tilde{\alpha}_h^*[\cdot], b_h^*(\cdot)) \in \Gamma \times \mathcal{B}$ defined by:

$$b_h^*(s) := \mathsf{b}^*_{\lfloor s/h \rfloor}, \quad \text{and} \quad \tilde{\alpha}_h^*[b](s) := \alpha_h^*[\hat{\mathsf{b}}]_{\lfloor s/h \rfloor}$$

where for any $b(\cdot) \in \mathcal{B}$, we define $(\hat{\mathsf{b}}_i)_i \in B^{N-k}$ by $\hat{\mathsf{b}}_i := b(ih)$ for $i = k, ..., N-1$. Notice that $b_h^*(\cdot) \in \mathcal{B}_h$ and, for any $b(\cdot) \in \mathcal{B}$, $\tilde{\alpha}_h^*[b](\cdot) \in \mathcal{A}_h$.
First, recall that the cost functional J, for $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$, is given by the following expression:

$$J(t,x,a,b) = \int_t^T \ell(s, y_{t,x}^{a,b}(s), a(s), b(s))ds + \Psi(y_{t,x}^{a,b}(T)),$$

and J_h, for $((\mathsf{a}_i)_i, (\mathsf{b}_i)_i) \in A^{N-k} \times B^{N-k}$, is given by:

$$J_h(t,x,\mathsf{a},\mathsf{b}) = (s_{k+1} - t)\ell(t,x,\mathsf{a}_k,\mathsf{b}_k) + h\sum_{i=k+1}^{N-1} \ell(s_i, \mathsf{y}_i, \mathsf{a}_i, \mathsf{b}_i) + \Psi(\mathsf{y}_N).$$

For $(a(\cdot), b(\cdot)) \in \mathcal{A}_h \times \mathcal{B}_h$, let's define $((\hat{\mathsf{a}}_i)_i, (\hat{\mathsf{b}}_i)_i) \in A^{N-k} \times B^{N-k}$ such that:

$$\hat{\mathsf{a}}_i = a(ih) \quad \text{and} \quad \hat{\mathsf{b}}_i = b(ih) \quad \text{for} \quad i = k, ..., N-1.$$

For any $(a(\cdot), b(\cdot)) \in \mathcal{A}_h \times \mathcal{B}_h$, we claim that:

$$|J(t,x,a,b) - J_h(t,x,\hat{\mathsf{a}},\hat{\mathsf{b}})| \leq \mathcal{O}(h), \tag{2.30}$$

which implies

$$|J(t,x,\tilde{\alpha}_h^*[b_h^*], b_h^*) - J_h(t,x,\alpha_h^*[\mathsf{b}^*], \mathsf{b}^*)| \leq \mathcal{O}(h). \tag{2.31}$$

On the other hand, from Proposition 2.6.7 and Theorem 2.6.2, we have

$$J_h(t,x,\alpha_h^*[\mathsf{b}^*], \mathsf{b}^*) = V_h(t,x) \quad \text{and} \quad \lim_{h\to 0^+} V_h(t,x) = v^\sharp(t,x).$$

Combining the two above equalities with estimation (2.31) ends the proof.
Now, we will justify the claim (2.30). Let $(a(\cdot), b(\cdot)) \in \mathcal{A}_h \times \mathcal{B}_h$ and $((\hat{\mathsf{a}}_i)_i, (\hat{\mathsf{b}}_i)_i) \in A^{N-k} \times B^{N-k}$ be defined as above and denote by $(\hat{\mathsf{y}}_i)_i$ the solution of (2.25) corresponding to $((\hat{\mathsf{a}}_i)_i, (\hat{\mathsf{b}}_i)_i)$.
One can prove the following estimations:

- For any $s \in [t, s_{k+1}[$:
$$\|y_{t,x}^{a,b}(s) - \hat{\mathsf{y}}_k\| \leq \mathcal{O}(h).$$
- By induction, for any $i \in \{k+1, ..., N-1\}$ and for any $s \in [s_i, s_{i+1}[$:
$$\|y_{t,x}^{a,b}(s) - \hat{\mathsf{y}}_i\| \leq \mathcal{O}(h).$$

- And finally:

$$\|y_{t,x}^{a,b}(T) - \hat{\mathsf{y}}_N\| \leq \mathcal{O}(h).$$

Using the definitions of J and J_h above, we obtain:

$$\begin{aligned}|J(t,x,a,b) - J_h(t,x,\hat{\mathsf{a}},\hat{\mathsf{b}})| \leq & \int_t^{s_{k+1}} |\ell(s, y_{t,x}^{a,b}(s), a(s), b(s)) - \ell(t,x,\hat{\mathsf{a}}_k,\hat{\mathsf{b}}_k)|ds \\ & + \sum_{i=k+1}^{N-1} \int_{s_i}^{s_{i+1}} |\ell(s, y_{t,x}^{a,b}(s), a(s), b(s)) - \ell(s_i,\hat{\mathsf{y}}_i,\hat{\mathsf{a}}_i,\hat{\mathsf{b}}_i)|ds \\ & + |\Psi(y_{t,x}^{a,b}(T)) - \Psi(\hat{\mathsf{y}}_N)|.\end{aligned}$$

Moreover, from the definition of $a(\cdot)$, $b(\cdot)$, $(\hat{\mathsf{a}}_i)_i$ and $(\hat{\mathsf{b}}_i)_i$, we deduce:

- For any $s \in [t, s_{k+1}[$, $a(s) = a(t) = \hat{\mathsf{a}}_k$ and $b(s) = b(t) = \hat{\mathsf{b}}_k$.
- For any $i \in \{k+1, ..., N-1\}$ and any $s \in [s_i, s_{i+1}[$, $a(s) = a(s_i) = \hat{\mathsf{a}}_i$ and $b(s) = b(s_i) = \hat{\mathsf{b}}_i$.

Therefore, using the Lipschitz continuity of ℓ and Ψ, the last inequality becomes:

$$\begin{aligned}|J(t,x,a,b) - J_h(t,x,\hat{\mathsf{a}},\hat{\mathsf{b}})| \leq & \int_t^{s_{k+1}} L_2(|s-t| + \|y_{t,x}^{a,b}(s) - \hat{\mathsf{y}}_k\|)ds \\ & + \sum_{i=k+1}^{N-1} \int_{s_i}^{s_{i+1}} L_2(|s_{i+1} - s_i| + \|y_{t,x}^{a,b}(s) - \hat{\mathsf{y}}_i\|)ds + L_3\|(y_{t,x}^{a,b}(T) - \hat{\mathsf{y}}_N)\|,\end{aligned}$$

where L_2 and L_3 are the Lipschitz constants of ℓ and Ψ respectively.
By the above estimations between the trajectories $y_{t,x}^{a,b}(\cdot)$ and $(\hat{\mathsf{y}}_i)_i$, we obtain the following estimations:

$$\int_t^{s_{k+1}} L_2(|s-t| + \|y_{t,x}^{a,b}(s) - \hat{\mathsf{y}}_k\|)ds \leq \int_t^{s_{k+1}} L_2(h + \mathcal{O}(h))ds \leq h\mathcal{O}(h) = \mathcal{O}(h^2),$$

and

$$\sum_{i=k+1}^{N-1} \int_{s_i}^{s_{i+1}} L_2(|s_{i+1} - s_i| + \|y_{t,x}^{a,b}(s) - \hat{\mathsf{y}}_i\|)ds \leq \sum_{i=k+1}^{N-1} \int_{s_i}^{s_{i+1}} L_2(h + \mathcal{O}(h))ds \leq Nh\mathcal{O}(h) = T\mathcal{O}(h) = \mathcal{O}(h).$$

As a conclusion, we get:

$$|J(t,x,a,b) - J_h(t,x,\hat{\mathsf{a}},\hat{\mathsf{b}})| \leq \mathcal{O}(h).$$

□

2.7 A game example

Consider a zero-sum differential game with a finite time horizon $T > 0$ and state dimension $d = 1$. The dynamics of the system is given by:

$$\dot{y}(s) = f(s, y(s), a(s), b(s)) = |a(s) - b(s)|, \quad s \in [0, T],$$

where the first and the second player controls $a(\cdot)$ and $b(\cdot)$ take values respectively in the control sets A and B with $A = B = [-1, 1]$.
The distributed and the final cost functions are given by $\ell(s, x, a, b) = e^x$ and $\Psi(\cdot) = 0$. The cost functional J, starting from $(t, x) \in [0, T] \times \mathbb{R}$ and corresponding to the controls $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$, is given by:

$$J(t,x,a,b) = \int_t^T \ell(s, y_{t,x}^{a,b}(s), a(s), b(s))ds.$$

2.7.1 Problem of the first player

This game corresponds to the case when the first player, by using nonanticipative strategies, tries to minimize J. The value function corresponding to this problem is of the form:

$$v(t,x) = \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \int_t^T e^{y_{t,x}^{\alpha[b],b}(s)} ds.$$

Since $\dot{y} = f \geq 0$ and the distributed cost ℓ is strictly increasing w.r.t. x, the optimal strategy of the the first player is to keep the system in its initial position. In other words, he tries to keep the dynamics f equal to zero. Henceforth, the first player optimal strategy is of the form:

$$\alpha^*[b](s) = b(s), \quad \forall s \in [0,T].$$

for any choice $b(\cdot) \in \mathcal{B}$ of the second player. Finally, we get the explicit expression of the value function v:

$$v(t,x) = (T-t)e^x, \quad \text{for any} \quad t \in [0,T] \quad \text{and} \quad x \in \mathbb{R}.$$

2.7.2 Problem of the second player

Now, we investigate the case when the second player uses nonanticipatives strategies in order to maximise J. The value function corresponding to this problem is:

$$u(t,x) := \sup_{\beta[\cdot]\in\Delta} \inf_{a(\cdot)\in\mathcal{A}} \int_t^T e^{y_{t,x}^{a,\beta[a]}(s)} ds.$$

Again, as the distributed cost ℓ is strictly increasing, the optimal choice for the second player is to maximize the dynamics f. Thus,

$$\beta^*[a](s) = \begin{cases} 1 & \text{if} \quad a(s) < 0, \\ -1 & \text{else.} \end{cases}$$

Knowing this information, the best choice of the first player is the one that minimizes the dynamics, hence $a^*(\cdot) \equiv 0$.
As a conclusion, the optimal trajectory associated to those choices is given by:

$$y_{t,x}^*(s) = x + s - t, \quad s \in [t,T],$$

and the value function $u(t,x) = e^x(e^{T-t} - 1)$ for any $t \in [0,T]$ and $x \in \mathbb{R}$.
On the other hand, one can check that $u \geq v$, which is in accordance with the result of Corollary 2.4.5.

Chapter 3

Hamilton-Jacobi Approach For Differential Game Problems With State Constraints

Publication from this chapter

N.Gammoudi and H.Zidani, *A differential game control problem with state constraints*, Mathematical Control and Related Fields, 2020 (submitted).

3.1 Introduction

In this chapter, we study the Hamilton-Jacobi (HJ) approach for a two-person zero-sum differential game with state constraints and where the controls of the two players are coupled within the dynamics, the cost functions and the state constraints. We characterize the value function of such a problem through an auxiliary differential game free of state constraints. Furthermore, we establish a link between the optimal strategies of the constrained problem and those of the auxiliary problem and we propose a general approach allowing to construct approximated optimal feebacks of the constrained differential game for both players.

Two-person zero-sum differential games provide a convenient framework for analyzing real conflict situations between two players where the gain of one player corresponds certainly to a loss of the other player. The most classical example is the *Target Problem* where the dynamics is controlled by both players, one player wants the dynamical system to reach, in finite time, a given set called the target while his opponent tries to avoid this target forever (see [41, 40, 42]). Another classical example is the *Pursuit-Evasion* game for which each player controls only half of the system's coordinates and the cost is the *capture time* which is the first time instant when the first player's coordinates become close enough to those of his opponent (see [46, 12, 62]).

Differential games can be studied in different contexts depending on the information advantage accorded to the two players. The most popular class of information pattern is nonanticipative strategies where one of the two players knows, at each time instant, the past and present choices of his opponent without having any idea about his future actions, see [59, 60, 119, 129, 56, 57].

We consider a two-person zero-sum differential game subject to state constraints where the first player is allowed to use nonanticipative strategies which are mappings from the set of controls of the second player, $\mathcal{B}$, to the actions set of the first player, $\mathcal{A}$. For a given finite time horizon $T > 0$, consider the following dynamical system:

$$\begin{cases} \dot{y}(s) = f(s, y(s), \alpha[b](s), b(s)), \ \ a.e. \ \ s \in [t, T], \\ y(t) = x, \end{cases} \tag{3.1}$$

where $\alpha[\cdot] \in \Gamma$ is a nonanticipative strategy of the first player, $b(\cdot) \in \mathcal{B}$ is an action of the second player, and f is a continuous function (more precise definitions and assumptions are given in section 3.2). The absolutely continuous solution of (3.1) is denoted by $y_{t,x}^{\alpha[b],b}(\cdot)$ and will be referred as the system trajectory corresponding to $(\alpha[b](\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$. This trajectory is said to be admissible if for any $s \in [t,T]$, $y_{t,x}^{\alpha[b],b}(s) \in \mathcal{K}$ where $\mathcal{K}$ is a closed non-empty subset of $\mathbb{R}^d$ representing the constraints set.

We are interested in the following differential game with maximum running cost:

$$v(t,x) := \inf_{\alpha[\cdot]\in\Gamma} \pi(t,x;\alpha) \tag{3.2}$$

with the convention that $\inf \varnothing = +\infty$ and where π is defined by:

$$\pi(t,x;\alpha) := \begin{cases} \sup\limits_{b(\cdot)\in\mathcal{B}} \left\{ \left(\max\limits_{s\in[t,T]} \phi(y_{t,x}^{\alpha[b],b}(s)) \right) \bigvee \psi(y_{t,x}^{\alpha[b],b}(T)) \right\}, \text{ if } y_{t,x}^{\alpha[b],b}(\cdot) \text{ is admissible, } \forall b(\cdot) \in \mathcal{B}, \\ +\infty, \text{ else.} \end{cases}$$

The cost functions $\phi : \mathbb{R}^d \to \mathbb{R}$ and $\psi : \mathbb{R}^d \to \mathbb{R}$ are continuous. This problem formulation can model the situation where the first player, representing the controller, uses his advantage of information to counteract to unknown disturbances, representing the second player of the game, that can affect the system and the cost functions, see section 3.5. For such situations, the traditional approach is to represent the disturbances via a statistical model and to optimize the expected value of the cost. However, this approach may not be effective against some catastrophic cases and it is not always possible to have a good statistical model. Therefore this situation can be modeled by a two-person zero-sum differential game where the first player minimizes the cost in the case of the worst behavior of his opponent. We can also imagine another game example where the second player objective is to maximize the cost π or to violate the state constraints.

Without state constraints, $\mathcal{K} \equiv \mathbb{R}^d$, this differential game was considered in [121, 15] with a maximum bounded cost function and in [116] with a Lipschitz continuous infimum cost. It was also studied in the case of a single controller in [111] through characterizing the value function epigraph by use of a *viability kernel*.

In the presence of state constraints, $\mathcal{K} \neq \mathbb{R}^d$, some difficulties appear. Indeed, the value function v may become discontinuous and its characterization as the unique viscosity solution of an HJ equation requires some additional assumptions involving the dynamics f and $\mathcal{K}$. In the case of one controller problems, the most popular assumption which is the Inward Pointing Condition, imposes the existence of a control value, at each point of the boundary of $\mathcal{K}$, that lets the dynamics point into the interior of $\mathcal{K}$. We refer to [34, 104, 124] for this assumption and to [103, 116, 121] for weaker inward pointing assumptions. Equivalent assumptions in the case of a two-person game can be found in [46, 23, 24]. Such assumptions cannot be always satisfied, which complicates the characterization of the value function as solution of an HJ equation. In this general setting, $\mathcal{K} \neq \mathbb{R}^d$, this problem has been studied in [6] in the case of a single controller and without assuming any controllability assumption by following the level set approach introduced in [5].

The first contribution of this chapter is that we do not assume any controllability assumption on the dynamics or on the set of state constraints. In addition to that, controls of the two players are allowed to be coupled within the dynamics, the cost functions and the state constraints. Moreover, we consider weak assumptions on f, ϕ and ψ which are supposed to be unbounded and locally Lipschitz continuous.

Here, we characterize v through a locally Lipschitz continuous value function of an unconstrained auxiliary differential game which is the unique viscosity solution of an HJ equation with an obstacle term. First, the set of state constraints $\mathcal{K}$ can be characterized via the signed distance $d_{\mathcal{K}}(\cdot)$ as follows:

$$\forall y \in \mathbb{R}^d, \quad d_{\mathcal{K}}(y) \leq 0 \Leftrightarrow y \in \mathcal{K}.$$

The value function w of the auxiliary problem is defined, for $t \in [0,T]$ and $(x,z) \in \mathbb{R}^d \times \mathbb{R}$, by:

$$w(t,x,z) := \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \left\{ \left(\max_{s\in[t,T]} \hat{\phi}(y_{t,x}^{\alpha[b],b}(s),z) \right) \bigvee \hat{\psi}(y_{t,x}^{\alpha[b],b}(T),z) \right\} \tag{3.3}$$

where for $(y,z) \in \mathbb{R}^d \times \mathbb{R}$, the functions $\hat{\phi}$ and $\hat{\psi}$ are given by:

$$\hat{\phi}(y,z) := (\phi(y) - z) \bigvee d_{\mathcal{K}}(y) \quad \text{and} \quad \hat{\psi}(y,z) := \psi(y) - z.$$

Then, we prove that w is the unique viscosity solution of the following HJ equation:

$$\begin{cases} \min\left(-\partial_t w(t,x,z) + H(t,x,D_x w(t,x,z)), w(t,x,z) - \hat{\phi}(x,z) \right) = 0, & \text{on } [0,T[\times\mathbb{R}^d \times \mathbb{R}, \\ w(T,x,z) = \hat{\phi}(x,z) \bigvee \hat{\psi}(x,z), & \text{on } \mathbb{R}^d \times \mathbb{R}, \end{cases} \tag{3.4}$$

where the hamiltonian H is given by:

$$H(t,x,p) := \min_{b\in B} \max_{a\in A} -\langle f(t,x,a,b), p\rangle, \quad \text{for} \quad (t,x,p) \in [0,T] \times \mathbb{R}^d \times \mathbb{R}^d.$$

The level sets of the auxiliary value function w can be used, without any additional assumptions, to estimate the constrained differential game value function v as follows:

$$\inf\{z \in \mathbb{R} \mid w(t,x,z) \leq 0\} \leq v(t,x) \leq \inf\{z \in \mathbb{R} \mid w(t,x,z) < 0\},$$

and when some convexity assumption is verified by f, we extend the main result of [5] to the case of a two-player games and we determinate v by the following relation:

$$v(t,x) = \inf\{z \in \mathbb{R} \mid w(t,x,z) \leq 0\}.$$

Another contribution of this chapter is to present a general approach allowing to construct approximated optimal feebacks of the constrained differential game for both players by use of the auxiliary differential game value function w. Indeed, we prove that an optimal strategy of the auxiliary problem (3.3) associated to a particular initial condition remains also optimal for the constrained problem (3.2). In addition to that, we propose algorithms to reconstruct approximated optimal strategies and controls for the auxiliary differential game.

Finally, as an illustrative example, we study an aircraft landing problem in the presence of windshear. Indeed, the best strategy to avoid a failed landing, that can occurs because of quick changes of the wind velocity, is to steer the aircraft to the maximal altitude that can be reached, during an interval of time, in order to prevent a crash on the ground. In [103, 104], a Chebyshev-type optimal control problem was proposed and an approximate solution is provided. The Hamilton-Jacobi-Bellman approach was applied in [6] to solve this problem after supposing the knowledge of the wind velocity fields. In [29], the aircraft landing problem was formulated as a nonlinear differential game with state constraints and a semi-Lagrangian scheme was applied to compute an approximation of the value function.

In chapter 3, we propose a 5D differential game model with maximum running cost, where wind disturbances are considered as a second player and the first player tries, by use of nonanticipative strategies, to counteract to some dangerous scenarios that can occur because of wind disturbances.

We organize this chapter as follows. Section 3.2 introduces the constrained differential game with maximum running cost and formulates its associated auxiliary problem. Section 3.3 shows how the auxiliary problem can be used to overcome the difficulties coming from the state constraints for the original problem. Section 3.4 presents some results concerning reconstruction of optimal trajectories for the auxiliary problem. A numerical example is given in section 3.5 which concerns an aircraft landing problem in presence of wind disturbances.

3.2 Problem formulation

3.2.1 Settings of the constrained differential game

Consider a two-person zero-sum differential game with finite time horizon $T > 0$. Actions of the first and the second players are measurable functions that take values respectively in A, a compact set of $\mathbb{R}^p$ ($p \geq 1$), and B, a compact set of $\mathbb{R}^q$ ($q \geq 1$). The set of admissible controls of the first and the second players, $\mathcal{A}$ and $\mathcal{B}$, are defined respectively as follows:

$$\mathcal{A} := \Big\{a(\cdot) : [t,T] \to A, \ \text{ measurable}\Big\} \quad \text{and} \quad \mathcal{B} := \Big\{b(\cdot) : [t,T] \to B, \ \text{ measurable}\Big\}.$$

In this chapter, the first player is allowed to use nonanticipative strategies.

Definition 3.2.1. *Following the formulation of Elliott and Kalton [56], a nonanticipative strategy of the first player is a map $\alpha[\cdot] : \mathcal{B} \to \mathcal{A}$, such that for any $\tau \leq T$ and any $b(\cdot), b'(\cdot) \in \mathcal{B}$, if $b(s) = b'(s)$ for almost every $s \leq \tau$, then $\alpha[b](\cdot) = \alpha[b'](\cdot)$ almost everywhere in $[0,\tau]$.*
We denote by Γ the set of nonanticipative strategies of the first player.

In other words, the first player takes his control decision at each time instant with the knowledge of the past and current choices of his opponent and without any idea about his future decisions.

For a choice $(\alpha[\cdot], b(\cdot)) \in \Gamma \times \mathcal{B}$ of the two players, consider the following dynamical system

$$\begin{cases} \dot{y}(s) = f(s, y(s), \alpha[b](s), b(s)), \ \ a.e. \ \ s \in [t,T], \\ y(t) = x \in \mathbb{R}^d. \end{cases} \tag{3.5}$$

The corresponding absolutely continuous solution of (3.5) is denoted by $y_{t,x}^{\alpha[b],b}(\cdot)$ and represents the system trajectory.

Furthermore, the following hypothesis will be considered throughout this chapter:

(H3.1) *The dynamics $f : [0,T] \times \mathbb{R}^d \times \mathbb{R}^p \times \mathbb{R}^q \mapsto \mathbb{R}^d$ is continuous and for any $R > 0$, there exists $L_f(R) > 0$, such that for any $(a,b) \in A \times B$, $s \in [0,T]$ and $y_1, y_2 \in \mathbb{R}^d$ verifying $\|y_1\|, \|y_2\| \leq R$:*

$$\|f(s, y_1, a, b) - f(s, y_2, a, b)\| \leq L_f(R)\|y_1 - y_2\|.$$

Moreover, there exists $c_f > 0$, s.t. $\forall\, y \in \mathbb{R}^d$, $\max\Big\{\|f(s,y,a,b)\|, s \in [0,T], (a,b) \in A \times B\Big\} \leq c_f(1 + \|y\|)$.

(H3.2) *The running cost function $\phi : \mathbb{R}^d \mapsto \mathbb{R}$ is locally Lipschitz continuous, i.e. for any $R > 0$ there exists $L_\phi(R) > 0$ s.t. for any $y_1, y_2 \in \mathbb{R}^d$ verifying $\|y_1\|, \|y_2\| \leq R$:*

$$|\phi(y_1) - \phi(y_2)| \leq L_\phi(R)\|y_1 - y_2\|.$$

Moreover, there exists $c_\phi > 0$ such that $\forall y \in \mathbb{R}^d$, $|\phi(y)| \leq c_\phi(1 + \|y\|)$.

(H3.3) *The final cost function $\psi : \mathbb{R}^d \mapsto \mathbb{R}$ is locally Lipschitz continuous, i.e. for any $R > 0$ there exists $L_\psi(R) > 0$ s.t. for any $y_1, y_2 \in \mathbb{R}^d$ verifying $\|y_1\|, \|y_2\| \leq R$:*

$$|\psi(y_1) - \psi(y_2)| \leq L_\psi(R)\|y_1 - y_2\|.$$

There exists also $c_\psi > 0$ such that $\forall y \in \mathbb{R}^d$, $|\psi(y)| \leq c_\psi(1 + \|y\|)$.

Let $\mathcal{K}$ be a non-empty closed subset of $\mathbb{R}^d$ representing the set of state constraints.

Definition 3.2.2. *A trajectory $y_{t,x}^{a,b}(\cdot)$, associated to a couple of actions of the two players $(a(\cdot), b(\cdot)) \in \mathcal{A}\times\mathcal{B}$, is said to be admissible if it remains in $\mathcal{K}$ at any time instant $s \in [t,T]$.*

We are interested in the following state-constrained differential game with maximum running cost:

$$v(t,x) := \inf_{\alpha[\cdot]\in\Gamma} \pi(t,x;\alpha) \tag{3.6}$$

with the convention that $\inf \varnothing = +\infty$ and where π is defined by:

$$\pi(t,x;\alpha) := \begin{cases} \sup\limits_{b(\cdot)\in\mathcal{B}} \left\{ \left(\max\limits_{s\in[t,T]} \phi(y_{t,x}^{\alpha[b],b}(s)) \right) \bigvee \psi(y_{t,x}^{\alpha[b],b}(T)) \right\}, \text{ if } y_{t,x}^{\alpha[b],b}(\cdot) \text{ is admissible, } \forall b(\cdot) \in \mathcal{B}, \\ +\infty, \quad \text{else.} \end{cases}$$

Problem (3.6) describes the situation where the first player is exploiting his information advantage and trying to find nonanticipative strategies that guarantee the admissibility of trajectories against any choice of the second player and minimize the cost functional π. This formulation can model the case where a controller tries to counteract to unknown disturbances which can affect the system and the cost functions. One can imagine another game example where the second player objective is to maximize the cost π or to violate the state constraints.

In general, for such state-constrained optimal control problems ($\mathcal{K} \neq \mathbb{R}^d$), the value function v is not essentially continuous and may require further controllability assumptions to characterize it as the unique viscosity solution of an appropriate HJ equation. An idea about such assumptions in the case of a two-person differential game can be found in [46, 23, 24].

As we said in section 3.1, we do not impose any controllability assumptions in this work. Following [5], we introduce an auxiliary control problem free of state constraints with a more regular value function allowing us to characterize v.

3.2.2 Associated auxiliary problem

First consider the augmented dynamics $\hat{f}$, defined for $s \in [0,T]$, $\hat{x} := (x,z) \in \mathbb{R}^d \times \mathbb{R}$ and $(a,b) \in A \times B$, by:

$$\hat{f}(s,\hat{x},a,b) := \begin{pmatrix} f(s,x,a,b) \\ 0 \end{pmatrix} \tag{3.7}$$

Denote by $\hat{y}_{t,x,z}^{\alpha[b],b}(\cdot)$, for $(\alpha[\cdot], b(\cdot)) \in \Gamma \times \mathcal{B}$, the unique solution of the following augmented differential system:

$$\begin{cases} \dot{\hat{y}}(s) = \hat{f}(s, \hat{y}(s), \alpha[b](s), b(s)), \quad \text{a.e. } s \in [t,T], \\ \hat{y}(t) = (x,z) \in \mathbb{R}^d \times \mathbb{R}. \end{cases} \tag{3.8}$$

Since the last component of the augmented dynamics $\hat{f}$ is equal to zero, $\hat{y}_{t,x,z}^{\alpha[b],b}(\cdot)$ can be expressed also as $\hat{y}_{t,x,z}^{\alpha[b],b}(\cdot) := \left(y_{t,x}^{\alpha[b],b}(\cdot), z \right)$, where $y_{t,x}^{\alpha[b],b}(\cdot)$ is the solution of (3.5).

Moreover, the set of constraints $\mathcal{K}$ is closed. Henceforth, it can be characterized as follows:

$$\forall y \in \mathbb{R}^d, \quad d_{\mathcal{K}}(y) \leq 0 \Leftrightarrow y \in \mathcal{K}, \tag{3.9}$$

where $d_{\mathcal{K}}(\cdot)$ is the signed distance to $\mathcal{K}$, defined by:

$$d_{\mathcal{K}}(x) := \begin{cases} -d(x, \partial\mathcal{K}) \text{ if } x \in \mathcal{K} \\ d(x, \partial\mathcal{K}) \text{ else,} \end{cases}$$

which is a Lipschitz continuous function. Therefore an admissible trajectory $y_{t,x}^{\alpha[b],b}(\cdot)$, corresponding to a couple of controls $(\alpha[b](\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$ can be characterized by means of the signed distance:

$$y_{t,x}^{\alpha[b],b}(s) \in \mathcal{K}, \quad \forall s \in [t,T] \Leftrightarrow \max_{s\in[t,T]} d_{\mathcal{K}}(y_{t,x}^{\alpha[b],b}(s)) \leq 0.$$

The value function w of the auxiliary problem can be defined, for $t \in [0,T]$ and $(x,z) \in \mathbb{R}^d \times \mathbb{R}$, by :

$$w(t,x,z) := \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \left\{ \left(\max_{s\in[t,T]} \hat{\phi}(y_{t,x}^{\alpha[b],b}(s), z) \right) \bigvee \hat{\psi}(y_{t,x}^{\alpha[b],b}(T), z) \right\} \tag{3.10}$$

where for $(x,z) \in \mathbb{R}^d \times \mathbb{R}$ and $s \in [0,T]$, the functions $\hat{\phi}$ and $\hat{\psi}$ are given by:

$$\hat{\phi}(x,z) := (\phi(x) - z) \bigvee d_{\mathcal{K}}(x) \quad \text{and} \quad \hat{\psi}(x,z) := \psi(x) - z.$$

Remark 3.2.3. *When the constrained problem (3.6) is of type Bolza, the auxiliary problem can be formulated by modifying the augmented dynamics $\hat{f}$ and the functions $\hat{\phi}$ and $\hat{\psi}$. In this case, the objective function J is given by:*

$$J(t,x,a,b) := \int_t^T \ell(s, y_{t,x}^{a,b}(s), a(s), b(s)) + \psi(y_{t,x}^{a,b}(T))$$

for $(t,x) \in [0,T] \times \mathbb{R}^d$ and $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$ and where ℓ and ψ are respectively the distributed and the final cost functions. For $\hat{x} = (x,z) \in \mathbb{R}^d \times \mathbb{R}$, consider the augmented dynamics $\hat{f}$:

$$\hat{f}(s,\hat{x},a,b) := \begin{pmatrix} f(s,x,a,b) \\ -\ell(s,x,a,b) \end{pmatrix},$$

and the cost functions $\hat{\phi}$ and $\hat{\psi}$:

$$\hat{\phi}(\hat{x}) := d_{\mathcal{K}}(x) \quad \textit{and} \quad \hat{\psi}(\hat{x}) = \psi(x) - z.$$

Let $\hat{y}_{t,\hat{x}}^{\alpha[b],b}(\cdot)$ be the unique continuous solution of the following differential system, associated to $(\alpha[b](\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$:

$$\begin{cases} \dot{\hat{y}}(s) = \hat{f}(s, \hat{y}(s), \alpha[b](s), b(s)), \quad a.e. \ \textit{in} \ [t,T], \\ \hat{y}(t) = \hat{x} := (x,z) \in \mathbb{R}^d \times \mathbb{R}. \end{cases}$$

Therefore, the corresponding auxiliary problem is given, for $t \in [0,T]$ and $\hat{x} \in \mathbb{R}^d \times \mathbb{R}$, by:

$$w(t,\hat{x}) := \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \left\{ \left(\max_{s\in[t,T]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha[b],b}(s)) \right) \bigvee \hat{\psi}(\hat{y}_{t,\hat{x}}^{\alpha[b],b}(T)) \right\}.$$

The above formulation still valid even for a problem of type Mayer ($\ell \equiv 0$). Furthermore, all the results that will be seen in the following sections still true also for a state-constrained problem of type Bolza or Mayer.

3.3 Properties of the value functions v and w

This section is devoted to some properties of the auxiliary value function w and to show how it can be used to characterize v, the value function of the constrained problem (3.6).

The following Proposition gives some results concerning the regularity of w and its characterization through Hamilton Jacobi equations.

Proposition 3.3.1. *Assume that hypothesis* **(H3.1)**, **(H3.2)** *and* **(H3.3)** *hold, then:*

(i) w *verifies a dynamic programming principle. For any* $h \in [0, T-t]$,

$$w(t,x,z) = \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ w(t+h, y_{t,x}^{\alpha[b],b}(t+h), z) \bigvee \Big(\max_{s\in[t,t+h]} \hat{\phi}(y_{t,x}^{\alpha[b],b}(s), z) \Big) \Big\}. \tag{3.11}$$

(ii) *The value function* w *is locally Lipschitz continuous on* $[0,T] \times \mathbb{R}^d \times \mathbb{R}$.

(iii) w *is the unique viscosity solution of the following HJ equation:*

$$\begin{cases} \min\Big(-\partial_t w(t,x,z) + H(t,x,D_x w(t,x,z)), w(t,x,z) - \hat{\phi}(x,z)\Big) = 0, & \text{in } [0,T[\times\mathbb{R}^d \times \mathbb{R}, \\ w(T,x,z) = \hat{\phi}(x,z) \bigvee \hat{\psi}(x,z), & \text{in } \mathbb{R}^d \times \mathbb{R}, \end{cases} \tag{3.12}$$

where the hamiltonian H *is given by:*

$$H(t,x,p) := \min_{b\in B} \max_{a\in A} -\langle f(t,x,a,b), p\rangle, \quad \text{for} \quad (t,x,p) \in [0,T] \times \mathbb{R}^d \times \mathbb{R}^d. \tag{3.13}$$

Proof. Here, we give only the proof of statement (iii). The proofs of (i) and (ii) can be found in Appendix 3.6 of this chapter.

We start by showing that w is a super-solution of (3.12). Let ξ be a function of class C^1 and (t,x,z) be a local minimum for $w-\xi$ such that $w(t,x,z) = \xi(t,x,z)$. Assume that there exists $\delta > 0$ such that

$$-\partial_t \xi(t,x,z) + H(t,x,D_x\xi(t,x,z)) = -\delta < 0.$$

In this case, there exists $b_0 \in B$ such that, for any $a \in A$,

$$-\partial_t \xi(t,x,z) - \langle f(t,x,a,b_0), D_x\xi(t,x,z)\rangle \leq -\delta.$$

Now let's fix $h > 0$ small enough and $\alpha[\cdot] \in \Gamma$. We have $\alpha[b_0](s) \in A$ for any time instant $s \in [t,T]$. Henceforth, from the last inequality, we get:

$$-\partial_t \xi(t,x,z) - \langle f(t,x,\alpha[b_0](s),b_0), D_x\xi(t,x,z)\rangle \leq -\delta, \text{ for any } s \in [t,t+h].$$

By continuity of ξ, f and $D_x\xi$, we obtain for any $s \in [t,t+h]$:

$$-\partial_t \xi(s, y_{t,x}^{\alpha[b_0],b_0}(s), z) - \langle f(s, y_{t,x}^{\alpha[b_0],b_0}(s), \alpha[b_0](s), b_0), D_x\xi(s, y_{t,x}^{\alpha[b_0],b_0}(s), z)\rangle \leq -\frac{\delta}{2},$$

henceforth

$$\int_t^{t+h} \Big(-\partial_t \xi(s, y_{t,x}^{\alpha[b_0],b_0}(s), z) - \langle f(s, y_{t,x}^{\alpha[b_0],b_0}(s), \alpha[b_0](s), b_0), D_x\xi(s, y_{t,x}^{\alpha[b_0],b_0}(s), z)\rangle \Big) ds \leq -\frac{\delta h}{2},$$

which implies that

$$\xi(t,x,z) - \xi(t+h, y_{t,x}^{\alpha[b_0],b_0}(t+h), z) \leq -\frac{\delta h}{2}.$$

On the other hand, since $w-\xi$ has a local minimum at (t,x,z) we obtain:

$$(w-\xi)(t+h, y_{t,x}^{\alpha[b_0],b_0}(t+h), z) \geq (w-\xi)(t,x,z).$$

From the two above inequalities, we deduce that

$$\frac{\delta h}{2} + w(t,x,z) \leq w(t+h, y_{t,x}^{\alpha[b_0],b_0}(t+h), z) \leq w(t+h, y_{t,x}^{\alpha[b_0],b_0}(t+h), z) \bigvee \Big(\max_{s\in[t,t+h]} \hat{\phi}(y_{t,x}^{\alpha[b_0],b_0}(s), z) \Big).$$

Therefore we get

$$\frac{\delta h}{2} + w(t,x,z) \leq \sup_{b(\cdot)\in\mathcal{B}} \Big\{ w(t+h, y_{t,x}^{\alpha[b],b}(t+h), z) \bigvee \Big(\max_{s\in[t,t+h]} \hat{\phi}(y_{t,x}^{\alpha[b],b}(s), z) \Big) \Big\}.$$

Since the last inequality holds for any $\alpha[\cdot] \in \Gamma$, we deduce that

$$\frac{\delta h}{2} + w(t,x,z) \leq \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ w(t+h, y_{t,x}^{\alpha[b],b}(t+h), z) \bigvee \Big(\max_{s\in[t,t+h]} \hat{\phi}(y_{t,x}^{\alpha[b],b}(s), z) \Big) \Big\} = w(t,x,z),$$

which is impossible. We conclude that

$$-\partial_t \xi(t,x,z) + H(t,x,D_x\xi(t,x,z)) \geq 0.$$

On the other hand, $\hat{\phi}(x,z) \leq \max_{s\in[t,T]} \hat{\phi}(y_{t,x}^{\alpha[b],b}(s), z) \leq w(t,x,z) = \xi(t,x,z)$. As a conclusion, $\min\Big(-\partial_t \xi(t,x,z) + H(t,x,D_x\xi(t,x,z)), \xi(t,x,z) - \hat{\phi}(x,z) \Big) \geq 0$, which means that w is a super-solution of (3.12).

Now, we will show that w is a sub-solution of (3.12). Let ξ be a function of class C^1 such that $w - \xi$ has a maximum at (t,x,z) and $w(t,x,z) = \xi(t,x,z)$. If $\xi(t,x,z) \leq \hat{\phi}(x,z)$, then ξ satisfies:

$$\min\Big(-\partial_t \xi(t,x,z) + H(t,x,D_x\xi(t,x,z)), \xi(t,x,z) - \hat{\phi}(x,z) \Big) \leq 0,$$

which means that w is a sub-solution of (3.12).

If not, we have $w(t,x,z) > \hat{\phi}(x,z)$. Henceforth, there exists $\tau > 0$, such that for any admissible trajectory $y_{t,x}^{\alpha[b],b}(\cdot)$, we have $w(s, y_{t,x}^{\alpha[b],b}(s), z) > \hat{\phi}(y_{t,x}^{\alpha[b],b}(s), z)$, for any $s \in [t, t+\tau]$. In this case and by using the dynamic programming principle (3.11) verified by w between t and $t+h$, for any $0 < h \leq \tau$, we get

$$w(t,x,z) = \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} w(t+h, y_{t,x}^{\alpha[b],b}(t+h), z). \tag{3.14}$$

Now, suppose that there exists $\delta > 0$, such that $-\partial_t \xi(t,x,z) + H(t,x,D_x\xi(t,x,z)) = \delta$. From Lemma 2.4.4 of chapter 2, there exists $\alpha^*[\cdot] \in \Gamma$ such that for any $b(\cdot) \in \mathcal{B}$ and $h > 0$ small enough the following inequality holds:

$$\zeta(t+h, y_{t,x}^{\alpha^*[b],b}(t+h), z) - \zeta(t,x,z) \leq -\frac{\delta h}{4}.$$

On the other hand, from the definition of ξ, we get for any $b(\cdot) \in \mathcal{B}$

$$w(t+h, y_{t,x}^{\alpha^*[b],b}(t+h), z) \leq \zeta(t+h, y_{t,x}^{\alpha^*[b],b}(t+h), z),$$

which implies that

$$\sup_{b(\cdot)\in\mathcal{B}} w(t+h, y_{t,x}^{\alpha^*[b],b}(t+h), z) - w(t,x,z) \leq -\frac{\delta h}{4},$$

henceforth, we conclude that

$$\inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} w(t+h, y_{t,x}^{\alpha[b],b}(t+h), z) - w(t,x,z) \leq -\frac{\delta h}{4} < 0.$$

From the equality (3.14), the last inequality is not possible. Hence, we deduce that

$$-\partial_t \xi(t,x,z) + H(t,x,D_x\xi(t,x,z)) \leq 0,$$

which concludes the proof. Uniqueness of w as the viscosity solution of (3.12) comes from the comparison result given in [5, Appendix].

□

After its characterization, w can be exploited to get an estimation of the value function v defined in (3.6) and this is the aim of the following Theorem.

Theorem 3.3.2. *Assume that hypothesis* **(H3.1)**, **(H3.2)** *and* **(H3.3)** *hold. The value function v can be estimated by means of w through the following relation :*

$$\inf\{z \in \mathbb{R} \mid w(t,x,z) \leq 0\} \leq v(t,x) \leq \inf\{z \in \mathbb{R} \mid w(t,x,z) < 0\}, \tag{3.15}$$

for any $(t,x) \in [0,T] \times \mathcal{K}$.

Proof. We start by proving that if $v(t,x) \leq z$, then $w(t,x,z) \leq 0$. Let $z \in \mathbb{R}$ such that $v(t,x) < z$. By definition of v, there exists a nonanticipative strategy $\alpha_0[\cdot] \in \Gamma$ such that for any $b(\cdot) \in \mathcal{B}$, the trajectory $y_{t,x}^{\alpha_0[b],b}(\cdot)$ remains in $\mathcal{K}$ and $\pi(t,x;\alpha_0) \leq z$. Therefore for any $b(\cdot) \in \mathcal{B}$ we get

$$\max_{s\in[t,T]} d_{\mathcal{K}}(y_{t,x}^{\alpha_0[b],b}(s)) \leq 0 \quad \text{and} \quad \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \Big(\max_{s\in[t,T]} \phi(y_{t,x}^{\alpha_0[b],b}(s)) \Big) \bigvee \psi(y_{t,x}^{\alpha_0[b],b}(T)) \Big\} \leq z.$$

From the two above inequalities, we deduce that

$$w(t,x,z) \leq \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \Big(\max_{s\in[t,T]} \hat{\phi}(y_{t,x}^{\alpha_0[b],b}(s),z) \Big) \bigvee \hat{\psi}(y_{t,x}^{\alpha_0[b],b}(T),z) \Big\} \leq 0.$$

We conclude that $w(t,x,z) \leq 0$ whenever $z > v(t,x)$. By continuity of w w.r.t. z, we deduce that for any $z \geq v(t,x)$ we have $w(t,x,z) \leq 0$. Henceforth

$$\inf\{z \in \mathbb{R} \mid w(t,x,z) \leq 0\} \leq v(t,x).$$

Now let $z \in \mathbb{R}$ such that $w(t,x,z) < 0$ and $\delta_0 := -w(t,x,z) > 0$. From the definition of w, there exists $\alpha_0[\cdot] \in \Gamma$ verifying:

$$\sup_{b(\cdot)\in\mathcal{B}} \Big\{ \Big(\max_{s\in[t,T]} \hat{\phi}(y_{t,x}^{\alpha_0[b],b}(s),z) \Big) \bigvee \hat{\psi}(y_{t,x}^{\alpha_0[b],b}(T),z) \Big\} \leq w(t,x,z) + \delta_0 = 0.$$

Therefore for any $b(\cdot) \in \mathcal{B}$ the trajectory $y_{t,x}^{\alpha_0[b],b}(\cdot)$ is admissible and

$$\sup_{b(\cdot)\in\mathcal{B}} \Big\{ \Big(\max_{s\in[t,T]} \phi(y_{t,x}^{\alpha_0[b],b}(s)) \Big) \bigvee \psi(y_{t,x}^{\alpha_0[b],b}(T)) \Big\} \leq z,$$

which means that $\pi(t,x;\alpha_0) \leq z$. By definition of v, we deduce that $v(t,x) \leq z$. This proves that

$$v(t,x) \leq \inf\{z \in \mathbb{R} \mid w(t,x,z) < 0\}.$$

□

Remark 3.3.3. *For any $(t,x) \in [0,T] \times \mathcal{K}$, if $\inf\{z \in \mathbb{R} \mid w(t,x,z) \leq 0\} = +\infty$, thereby $v(t,x) = +\infty$ which means that there is no strategies of the first player that guarantee the admissibility of the trajectories for any action $b(\cdot) \in \mathcal{B}$ of the second player. However if*

$$\inf\{z \in \mathbb{R} \mid w(t,x,z) \leq 0\} \in \mathbb{R},$$

then the infimum is reached by some $z \in \mathbb{R}$. Furthermore, if $\inf\{z \in \mathbb{R} \mid w(t,x,z) < 0\} \in \mathbb{R}$ then one can prove that:

$$\inf\{z \in \mathbb{R} \mid w(t,x,z) \leq 0\} = v(t,x) = \inf\{z \in \mathbb{R} \mid w(t,x,z) < 0\}.$$

On the other hand it may occur that

$$\inf\{z \in \mathbb{R} \mid w(t,x,z) \leq 0\} \in \mathbb{R} \quad \textit{and} \quad \inf\{z \in \mathbb{R} \mid w(t,x,z) < 0\} = +\infty$$

and in this case we have no information on $v(t,x)$.

Remark 3.3.4. *In [5], it was shown that when some convexity assumption is verified by f, a precise connection is established between v and w. Theorem 3.3.2 gives a more general result on the link between those two value functions.*

When some convexity assumption is verified by f and A, we can prove a more precise connection. For this, assume that:

(H3.4) *A is a convex subset of $\mathbb{R}^p$ and the dynamics f is affine in the first control variable a i.e. f is of the form*

$$f(t,x,a,b) := f_0(t,x,b) + f_1(t,x,b)a.$$

The aim of the following Theorem is to characterize the value of the constrained problem v through the auxiliary value function w. In addition to that, it establishes a link between optimal strategies of the auxiliary and the constrained problems.

Theorem 3.3.5. *Assume that assumptions* **(H3.1)**, **(H3.2)**, **(H3.3)** *and* **(H3.4)** *hold and let $(t,x) \in [0,T]\times\mathcal{K}$.*

(i) Suppose that $w(t,x,z) \leq 0$ for some $z \in \mathbb{R}$, then there exists $\alpha^*[\cdot] \in \Gamma$ such that for any $b(\cdot) \in \mathcal{B}$, the trajectory $y_{t,x}^{\alpha^*[b],b}(\cdot)$ is admissible and

$$\Big(\max_{s\in[t,T]} \phi(y_{t,x}^{\alpha^*[b],b}(s))\Big) \bigvee \psi(y_{t,x}^{\alpha^*[b],b}(T)) \leq z.$$

(ii) The exact value of v can be determined through the following relation:

$$v(t,x) = \inf\{z \in \mathbb{R} \;\mid\; w(t,x,z) \leq 0\}. \tag{3.16}$$

(iii) If $\overline{z} := v(t,x) < \infty$, then any optimal strategy of the auxiliary problem (3.10) on $[t,T]$ starting from the initial state $(x,\overline{z})$ is optimal for the constrained problem (3.6) on $[t,T]$ associated with the initial position x.

Proof. (i) The proof of this assertion uses some basic ideas from [40, 41, 42, 69]. First, let $\Lambda : \mathcal{B} \to \mathcal{A}$ be a set-valued map defined, for any $b(\cdot) \in \mathcal{B}$, by:

$$\Lambda(b) := \left\{ a(\cdot) \in \mathcal{A} \;\mid\; \Big(\max_{s\in[t,T]} \hat{\phi}(y_{t,x}^{a,b}(s),z)\Big) \bigvee \hat{\psi}(y_{t,x}^{a,b}(T),z) \leq 0 \right\}.$$

$\Lambda(\cdot)$ is said to be nonanticipative if for any $\tau \in [0, T-t]$, for any $b_1(\cdot), b_2(\cdot) \in \mathcal{B}$ which coincide almost everywhere on $[t, t+\tau]$ and for any $a_1(\cdot) \in \Lambda(b_1)$, one can find $a_2(\cdot) \in \Lambda(b_2)$ which coincides with $a_1(\cdot)$ almost everywhere on $[t, t+\tau]$ (see [40, 41, 42]).

Now, let $(t,x,z) \in [0,T]\times\mathbb{R}^d\times\mathbb{R}$ such that $w(t,x,z) \leq 0$. By the definition of w, for any $n \in \mathbb{N}^*$, there exists $\alpha_n[\cdot] \in \Gamma$ such that:

$$\sup_{b(\cdot)\in\mathcal{B}} \left\{ \Big(\max_{s\in[t,T]} \hat{\phi}(y_{t,x}^{\alpha_n[b],b}(s),z)\Big) \bigvee \hat{\psi}(y_{t,x}^{\alpha_n[b],b}(T),z) \right\} \leq \frac{1}{n}.$$

We start by showing that the set-valued map $\Lambda(\cdot)$ has nonempty values. Indeed, let fix $b(\cdot) \in \mathcal{B}$ and let $y_n(\cdot)$ be the solution of:

$$\begin{cases} \dot{y}_n(s) = f(s, y_n(s), \alpha_n[b](s), b(s)), \quad a.e.\ s \in [t,T], \\ y_n(t) = x. \end{cases}$$

Therefore, for any $n \in \mathbb{N}^*$, $y_n(\cdot)$ verifies:

$$\Big(\max_{s\in[t,T]} \hat{\phi}(y_n(s), z)\Big) \bigvee \hat{\psi}(y_n(T), z) \leq \frac{1}{n}.$$

Now, denote by $S_{[t,T]}(x)$ the set of absolutely continuous solutions of:

$$\begin{cases} \dot{y}(s) \in f(s, y(s), A, b(s)), \ \ a.e. \ \ s \in [t,T], \\ y(t) = x. \end{cases}$$

Under hypothesis **(H3.1)** and **(H3.4)**, $S_{[t,T]}(x)$ is a compact subset of $W^{1,1}([0,T])$ for the topology of $C([0,T];\mathbb{R}^d)$ (see [7, 8]). Therefore when $n \to \infty$, $y_n(\cdot)$ converges to $y^*(\cdot) \in S_{[t,T]}(x)$ solution of:

$$\begin{cases} \dot{y}^*(s) \in f(s, y^*(s), A, b(s)), \ \ a.e. \ \ s \in [t,T], \\ y^*(t) = x. \end{cases}$$

By the measurable selection theorem from [69], there exists a control of the first player, $a_b(\cdot) \in \mathcal{A}$, depending on $b(\cdot) \in \mathcal{B}$ which is already fixed, verifying $y^*(\cdot) = y_{t,x}^{a_b,b}(\cdot)$ almost everywhere on $[t,T]$. By continuity of $\hat{\phi}$ and $\hat{\psi}$, we conclude that:

$$\Big(\max_{s\in[t,T]} \hat{\phi}(y_{t,x}^{a_b,b}(s), z)\Big) \bigvee \hat{\psi}(y_{t,x}^{a_b,b}(T), z) \leq 0,$$

which means that $a_b(\cdot) \in \Lambda(b)$ and therefore $\Lambda(\cdot)$ has nonempty values.

Then, we claim that the set-valued map $\Lambda(\cdot)$ is nonanticipative. To prove this claim, let $\tau \in [0, T-t]$, $b_1(\cdot), b_2(\cdot) \in \mathcal{B}$ coinciding almost everywhere on $[t, t+\tau]$ and consider $a_1(\cdot) \in \Lambda(b_1)$. From the definition of Λ, the trajectory $y_{t,x}^{a_1,b_2}(\cdot)$ is admissible (will remain in $\mathcal{K}$ on $[t, t+\tau]$) and verifies:

$$\Big(\max_{s\in[t,t+\tau]} \phi(y_{t,x}^{a_1,b_2}(s))\Big) \bigvee \psi(y_{t,x}^{a_1,b_2}(t+\tau)) \leq z.$$

Now, consider $a_2(\cdot) \in \mathcal{A}$ such that $a_2(\cdot) := a_1(\cdot)$ on $[t, t+\tau]$. Starting at time $t+\tau$ from x':=$y_{t,x}^{a_1,b_2}(t+\tau)$ and by exploiting the same arguments already used to show that $\Lambda(\cdot)$ has nonempty values, there exists $a_{b_2}(\cdot) \in \mathcal{A}$ s.t. $y_{t+\tau,x'}^{a_{b_2},b_2}(\cdot)$ is admissible $i.e.$ will remain in $\mathcal{K}$ on $[t+\tau, T]$ and verifies:

$$\Big(\max_{s\in[t+\tau,T]} \phi(y_{t+\tau,x'}^{a_{b_2},b_2}(s))\Big) \bigvee \psi(y_{t+\tau,x'}^{a_{b_2},b_2}(T)) \leq z.$$

Define the control $a_2(\cdot) \in \mathcal{A}$ by:

$$a_2(\cdot) = \begin{cases} a_1(\cdot) & \text{on } \ [t, t+\tau], \\ a_{b_2}(\cdot) & \text{on } \ [t+\tau, T] \end{cases}$$

which belongs to $\Lambda(b_2)$ and coincides almost everywhere with $a_1(\cdot)$ on $[t, t+\tau]$. Henceforth we conclude that $\Lambda(\cdot)$ is nonanticipative.

Finally, $\Lambda(\cdot)$ has closed values for the weak topology of $L^2([0,T], A)$. Indeed, let $(a_n(\cdot))_n$ be a sequence of $\Lambda(b)$, for a fixed control of the second player $b(\cdot) \in \mathcal{B}$, that converges, for the weak topology of $L^2([0,T], A)$, to some control $a(\cdot) \in \mathcal{A}$.

Since for any $n \in \mathbb{N}$, $a_n(\cdot) \in \Lambda(b)$, the trajectory $y_{t,x}^{a_n,b}(\cdot)$ verifies:

$$\Big(\max_{s\in[t,T]} \hat{\phi}(y_{t,x}^{a_n,b}(s), z)\Big) \bigvee \hat{\psi}(y_{t,x}^{a_n,b}(T), z) \leq 0.$$

Under hypothesis **(H3.1)** and **(H3.4)**, the sequence $(y_{t,x}^{a_n,b}(\cdot))_n$ will converge, for the compact convergence, to $y_{t,x}^{a,b}(\cdot)$. By continuity of $\hat{\phi}$ and $\hat{\psi}$, we deduce that:

$$\Big(\max_{s\in[t,T]}\hat{\phi}(y_{t,x}^{a,b}(s),z)\Big)\bigvee\hat{\psi}(y_{t,x}^{a,b}(T),z)\leq 0.$$

Therefore, $a(\cdot)$ belongs to $\Lambda(b)$. We conclude that the set-valued map $\Lambda(\cdot)$ has closed values for the weak topology of the Hilbert space $L^2([0,T],A)$.

To end this proof, it is enough to use [41, Lemma 4.1] that guarantees the existence of a nonanticipative selection $\alpha^*[\cdot]$ such that for any $b(\cdot)\in\mathcal{B}$, $\alpha^*[b](\cdot)\in\Lambda(b)$. We conclude that there exists $\alpha^*[\cdot]\in\Gamma$ s.t. for any $b(\cdot)\in\mathcal{B}$, the trajectory $y_{t,x}^{\alpha^*[b],b}(\cdot)$ is admissible and

$$\Big(\max_{s\in[t,T]}\phi(y_{t,x}^{\alpha^*[b],b}(s))\Big)\bigvee\psi(y_{t,x}^{\alpha^*[b],b}(T))\leq z.$$

(ii) From Theorem 3.3.2, we already know that

$$\inf\{z\in\mathbb{R}\ \mid\ w(t,x,z)\leq 0\}\leq v(t,x).$$

On the other hand, let $z\in\mathbb{R}$ such that $w(t,x,z)\leq 0$. From assertion (i), there exists $\alpha^*[\cdot]\in\Gamma$ s.t. for any $b(\cdot)\in\mathcal{B}$, the trajectory $y_{t,x}^{\alpha^*[b],b}(\cdot)$ is admissible and

$$\Big(\max_{s\in[t,T]}\phi(y_{t,x}^{\alpha^*[b],b}(s))\Big)\bigvee\psi(y_{t,x}^{\alpha^*[b],b}(T))\leq z,$$

which means that $\pi(t,x;\alpha^*)\leq z$. By definition of v, we conclude that $v(t,x)\leq z$, for any $z\in\mathbb{R}$ verifying $w(t,x,z)\leq 0$. Henceforth,

$$v(t,x)\leq\inf\{z\in\mathbb{R}\ \mid\ w(t,x,z)\leq 0\}.$$

(iii) Let $\bar{z}:=v(t,x)<\infty$ and $\alpha^*[\cdot]\in\Gamma$ be an optimal strategy of the auxiliary problem (3.10) on $[t,T]$ associated with the initial point $(x,\bar{z})$ which means that

$$w(t,x,\bar{z})=\sup_{b(\cdot)\in\mathcal{B}}\Big\{\Big(\max_{s\in[t,T]}\hat{\phi}(y_{t,x}^{\alpha^*[b],b}(s),\bar{z})\Big)\bigvee\hat{\psi}(y_{t,x}^{\alpha^*[b],b}(T),\bar{z})\Big\}.$$

Since $\bar{z}=v(t,x)$, we get $w(t,x,\bar{z})\leq 0$ and hence

$$\sup_{b(\cdot)\in\mathcal{B}}\Big\{\Big(\max_{s\in[t,T]}\hat{\phi}(y_{t,x}^{\alpha^*[b],b}(s),\bar{z})\Big)\bigvee\hat{\psi}(y_{t,x}^{\alpha^*[b],b}(T),\bar{z})\Big\}\leq 0.$$

Therefore, for any $b(\cdot)\in\mathcal{B}$, the trajectory $y_{t,x}^{\alpha^*[b],b}(\cdot)$ is admissible and

$$\sup_{b(\cdot)\in\mathcal{B}}\Big\{\Big(\max_{s\in[t,T]}\phi(y_{t,x}^{\alpha^*[b],b}(s))\Big)\bigvee\psi(y_{t,x}^{\alpha^*[b],b}(T))\Big\}\leq\bar{z}=v(t,x).$$

Henceforth, $\pi(t,x;\alpha^*)\leq v(t,x)=\inf_{\alpha[\cdot]\in\Gamma}\pi(t,x;\alpha)$. We conclude that $\alpha^*[\cdot]$ is an optimal strategy for the constrained problem (3.6) associated with the initial state x. □

Comments: Reduction of the computational domain Since problem (3.10) is free of state constraints, the auxiliary value function w is defined on $[0,T]\times\mathbb{R}^d\times\mathbb{R}$. Nevertheless, for computational issues, we should restrict the domain of interest of w to a neighbourhood of $\mathcal{K}\times\mathbb{R}$. To this end, we will follow a technique developed in [6] where the auxiliary value function w will keep a constant value outside the neighbourhood of $\mathcal{K}\times\mathbb{R}$. Let $\mu>0$ be a fixed parameter and $\mathcal{K}_\mu$ be a neighbourhood of $\mathcal{K}$ defined by

$\mathcal{K}_\mu := \mathcal{K} + \mu\mathbb{B}_{\mathbb{R}^d}$.

The idea consists in introducing a truncation of $d_{\mathcal{K}}$, $\hat{\phi}$ and $\hat{\psi}$ to obtain a new control problem free of state constraints with value function w_μ characterized by a constant value outside $\mathcal{K}_\mu$.

First consider the Lipschitz continuous function $d^\mu_{\mathcal{K}} := d_{\mathcal{K}} \bigwedge \mu$ which verifies for any $y \in \mathbb{R}^d$:

$$d^\mu_{\mathcal{K}}(y) \leq 0 \Leftrightarrow y \in \mathcal{K}, \quad d^\mu_{\mathcal{K}}(y) \leq \mu, \quad \text{and} \quad d^\mu_{\mathcal{K}}(y) = \mu \Leftrightarrow y \notin \mathcal{K}_\mu.$$

Furthermore, we consider a truncation of $\hat{\phi}$ and $\hat{\psi}$ as follows:

$$\hat{\phi}_\mu := \hat{\phi} \bigwedge \mu \quad \text{and} \quad \hat{\psi}_\mu := \hat{\psi} \bigwedge \mu.$$

Finally, we define the specific auxiliary value function w_μ, for $(t,x,z) \in [0,T] \times \mathbb{R}^d \times \mathbb{R}$ as:

$$w_\mu(t,x,z) := \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \Big(\max_{s\in[t,T]} \hat{\phi}_\mu(y^{\alpha[b],b}_{t,x}(s), z) \Big) \bigvee \hat{\psi}_\mu(y^{\alpha[b],b}_{t,x}(T), z) \Big\},$$

which verifies the following relation:

$$w_\mu(t,x,z) = w(t,x,z) \bigwedge \mu.$$

Since, we are interested in the region $\{z \mid w(t,x,z) \leq 0\}$, for $(t,x) \in [0,T] \times \mathcal{K}$, which coincides with $\{z \mid w_\mu(t,x,z) \leq 0\}$ for any $\mu > 0$, it does not matter which auxiliary value function we use (w or w_μ). Therefore, for the sequel we will confound w and w_μ, for any $\mu > 0$, which will be denoted simply by w.

The question now is how to characterize w_μ and this is the aim of the following Proposition.

Proposition 3.3.6. *w_μ is the unique viscosity solution of the following HJ equation:*

$$\begin{cases} \min\Big(-\partial_t w_\mu(t,x,z) + H(t,x,D_x w_\mu(t,x,z)), w_\mu(t,x,z) - \hat{\phi}_\mu(x,z)\Big) = 0, & \text{on } [0,T[\times \overset{\circ}{\mathcal{K}}_\mu \times \mathbb{R}, \\ w_\mu(T,x,z) = \hat{\phi}_\mu(x,z) \bigvee \hat{\psi}_\mu(x,z), & \text{on } \overset{\circ}{\mathcal{K}}_\mu \times \mathbb{R}, \\ w_\mu(t,x,z) = \mu, & \text{for } t \in [0,T],\ x \notin \overset{\circ}{\mathcal{K}}_\mu \text{ and } z \in \mathbb{R}. \end{cases}$$

Proposition 3.3.6 can be proven in a similar way to assertion (iii) of Proposition 3.3.1.

The aim of Proposition 3.3.6 is to characterize the specific auxiliary value function, w_μ, having the same region of interest as w and for which the computational domain is restricted to a neighbourhood of $[0,T] \times \mathcal{K} \times \mathbb{R}$ in contrary to w which is defined on $[0,T] \times \mathbb{R}^d \times \mathbb{R}$.

Remark 3.3.7. *Suppose that the cost functions ϕ and ψ are bounded (they take values in some interval $[m,M]$). Thus, to establish estimations of the value function v or to find its exact value, as in (3.15) or (3.16), it is enough to consider the auxiliary variable z in the interval $[m,M]$.*

3.4 Trajectory reconstruction based on the value function and approximation by discrete time games

This section is devoted to present our proposed reconstruction procedure that will be applied to approximate optimal strategies and controls of the auxiliary problem (3.10). To this end, we will consider a differential game free of state constraints having the following general form:

$$u(t,\chi) := \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \Big(\max_{s\in[t,T]} \Phi(\zeta^{\alpha[b],b}_{t,\chi}(s)) \Big) \bigvee \Psi(\zeta^{\alpha[b],b}_{t,\chi}(T)) \Big\}, \tag{3.17}$$

where $T > 0$, Φ and Ψ are the maximum running and the final cost functions respectively and $\zeta_{t,\chi}^{\alpha[b],b}(\cdot)$ is the unique continuous solution of the following dynamical system:

$$\begin{cases} \dot{\zeta}(s) = F(s, \zeta(s), \alpha[b](s), b(s)), \ \ a.e. \ \text{ in } \ [t, T], \\ \zeta(t) = \chi, \end{cases} \tag{3.18}$$

where $\chi \in \mathbb{R}^m$, $m \geq 1$, $(\alpha[b](\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$ are the actions of the first and the second players respectively and F is a nonlinear dynamics.

Furthermore, throughout this section we suppose that F, Φ and Ψ satisfy respectively hypothesis **(H3.1)**, **(H3.2)** and **(H3.3)**. Henceforth, from section 3.2 we already know that u is the unique viscosity solution of the following HJ equation:

$$\begin{cases} \min\left(-\partial_t u(t,\chi) + H(t,\chi, D_\chi u(t,\chi)), u(t,\chi) - \Phi(\chi)\right) = 0, \quad \text{for} \quad (t,\chi) \in [0,T] \times \mathbb{R}^m, \\ u(T,\chi) = \Phi(\chi) \bigvee \Psi(\chi), \quad \text{for} \quad \chi \in \mathbb{R}^m, \end{cases} \tag{3.19}$$

where the Hamiltonian H is given by:

$$H(t,\chi,p) := \min_{b \in B} \ \max_{a \in A} - \langle F(t,\chi,a,b), p \rangle, \quad \text{for} \quad (t,\chi,p) \in [0,T] \times \mathbb{R}^m \times \mathbb{R}^m.$$

We denote by J the cost functional in (3.17):

$$J(t,\chi,a,b) := \left(\max_{s \in [t,T]} \Phi(\zeta_{t,\chi}^{a,b}(s)) \right) \bigvee \Psi(\zeta_{t,\chi}^{a,b}(T)), \tag{3.20}$$

for $(t,\chi) \in [0,T] \times \mathbb{R}^m$ and $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$.

Recall that the aim of this section is to approximate optimal feedbacks of the differential game (3.17). To this end, we will discretize in time and synthesize from the discrete time differential game an approximation of the optimal strategy of the first player and the optimal control of the second player. Consider a uniform time partition of $[0,T]$ with time step $h := \frac{T}{N}$, $N \in \mathbb{N}^* : s_0 = 0$, $s_1 = h$,..., $s_k = kh$,..., $s_N = T$.

The dynamical system (3.18) can be approximated by the following Euler forward scheme:

$$\begin{cases} \zeta_k = \chi \\ \zeta_{k+1} = \chi + (s_{k+1} - t) F(t, \zeta_k, \mathsf{a}_k, \mathsf{b}_k) \\ \zeta_{i+1} = \zeta_i + h F(s_i, \zeta_i, \mathsf{a}_i, \mathsf{b}_i), \quad i = k+1, ..., N-1. \end{cases} \tag{3.21}$$

for $t \in [s_k, s_{k+1}[$ with $0 \leq k \leq N-1$, $\chi \in \mathbb{R}^m$, $(\mathsf{a}_i)_i \in A^{N-k}$ and $(\mathsf{b}_i)_i \in B^{N-k}$. More precise approximations of (3.18) can be considered by using higher order Runge-Kutta schemes.

Finally, even for the discrete time game, we attribute to the first player an advantage of information. This advantage of information can be modeled by discrete nonanticipative strategies. Indeed, at each time step s_i, for $i \in \{0, ..., N-1\}$, and before choosing his action $\mathsf{a}_i \in A$, the first player knows the choice of his opponent $\mathsf{b}_i \in B$ without having any information about his future choices.

Subsection 3.4.1 deals with the case of trajectory reconstruction by use of a general class of approximated functions u_h while in subsection 3.4.2, we show a convergence result when the approximation u_h verifies a specific criterion.

3.4.1 A general reconstruction procedure

In this part, consider a general approximation u_h that may come from the numerical resolution of a discretized form of the Hamilton Jacobi equation (3.19) verified by u. Let E_h denote the uniform error estimate between u and u_h given by $E_h := \|u_h - u\|$.

(H3.5) *Suppose that the error estimate* E_h *satisfies* $E_h = o(h)$.

The reconstruction procedure presented in Algorithm 3.1 corresponds to the case where choices of the second player are not optimal and take arbitrary values in B. The first player will observe his opponent choice and will choose his optimal reaction. This algorithm is given for a particular reconstruction from the initial time instant $t = 0$ and an initial position $\chi \in \mathbb{R}^m$.

Algorithm 3.1: Arbitrary case

1: Initialise $\zeta_0 = \chi$.
2: **for** $i = 0, ..., N-1$ **do**
3: An arbitrary choice of the second player $\mathsf{b}_i \in B$.
4: The optimal reaction of the first player $\mathsf{a}_i^* \in A$ is defined by:

$$\mathsf{a}_i^* \in \underset{a\in A}{\operatorname{argmin}}\Big\{u_h(s_{i+1}, \zeta_i + hF(s_i, \zeta_i, a, \mathsf{b}_i)) \bigvee \Phi(\zeta_i)\Big\}.$$

5: The new state position: $\zeta_{i+1} = \zeta_i + hF(s_i, \zeta_i, \mathsf{a}_i^*, \mathsf{b}_i)$.
6: **end for**

For $\chi \in \mathbb{R}^m$, let $(\mathsf{a}_i^*)_i$, $(\mathsf{b}_i)_i$ and $(\zeta_i)_i$ be the sequences of controls and trajectory generated by Algorithm 3.1 and we define the following piecewise constant controls $(\tilde{\alpha}_h^*[b_h](\cdot), b_h(\cdot)) \in \mathcal{A}\times\mathcal{B}$, such that $\tilde{\alpha}_h^*[b_h](s) := \mathsf{a}_k^*$ and $b_h(s) := \mathsf{b}_k$, for $s \in [s_k, s_{k+1}[$ with $k \in \{0, ..., N-1\}$, and an approximate trajectory $\zeta_h(\cdot)$ solution of:

$$\begin{cases} \dot{\zeta}_h(s) = F(s, \zeta_h(s), \tilde{\alpha}_h^*[b_h](s), b_h(s)), \quad a.e. \quad \text{in } [0,T], \\ \zeta_h(0) = \chi. \end{cases} \tag{3.22}$$

Theorem 3.4.1. *Assume that hypothesis* **(H3.5)** *holds and that assumption* **(H3.4)** *is verified by the set* A *and the dynamics* F. *For* $\chi \in \mathbb{R}^m$, *the trajectory* $\zeta_h(\cdot)$, *defined in* (3.22), *verifies:*

$$\limsup_{h\to 0^+}\Big\{\Big(\max_{s\in[0,T]} \Phi(\zeta_h(s))\Big) \bigvee \Psi(\zeta_h(T))\Big\} \le u(0,\chi). \tag{3.23}$$

Proof of Theorem 3.4.1. Let $\chi \in \mathbb{R}^m$ and let $(\zeta_i)_i$, $(\mathsf{a}_i^*)_i$ and $(\mathsf{b}_i)_i$ be the sequences of trajectory and players' actions generated by Algorithm 3.1.

Since F is locally Lipschitz continuous, there exists $R > 0$ such that for any time step $h > 0$ and any $0 \le k \le N$, we have $\|\zeta_k\| \le R$. We can choose the constant R large enough such that any trajectory starting from any initial position ζ_k will remain in $\mathbb{B}(0,R)$, the ball of $\mathbb{R}^d$ centred at 0 and with radius R. Let $M_R > 0$ be a constant verifying:

$$\|F(s,\zeta,a,b)\| \le M_R, \quad \text{for any } \ s \in [0,T], \ \ \zeta \in \mathbb{B}(0,R) \ \text{ and } \ (a,b) \in A\times B.$$

Step 1. We claim that there exists $\epsilon_h > 0$, s.t. $\lim_{h\to 0} \epsilon_h = 0$, and

$$u_h(s_0,\zeta_0) \ge u_h(s_1,\zeta_1) \bigvee \Phi(\chi) - h\epsilon_h - 2E_h \quad \text{with} \quad s_0 = 0 \ \text{ and } \ \zeta_0 = \chi. \tag{3.24}$$

By the dynamic programming principle verified by u, between s_0 and $s_1 = h$, we get

$$u(s_0,\chi) = \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{u(s_1, \zeta_{s_0,\chi}^{\alpha[b],b}(s_1)) \bigvee \Big(\max_{s\in[s_0,s_1]} \Phi(\zeta_{s_0,\chi}^{\alpha[b],b}(s))\Big)\Big\},$$

which implies that

$$u(s_0,\chi) \ge \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{u(s_1, \zeta_{s_0,\chi}^{\alpha[b],b}(s_1)) \bigvee \Phi(\chi)\Big\}.$$

For $\epsilon > 0$, we pick an ϵ-optimal strategy, $\alpha^\epsilon[\cdot] \in \Gamma$, and the above inequality becomes:

$$u(s_0, \chi) \geq -\epsilon + \sup_{b(\cdot)\in\mathcal{B}} \Big\{ u(s_1, \zeta_{s_0,\chi}^{\alpha^\epsilon[b],b}(s_1)) \bigvee \Phi(\chi) \Big\}.$$

Let $b_0(\cdot) \in \mathcal{B}$ be a constant control such that $b_0(\cdot) = \mathsf{b}_0 \in B$ where b_0 is generated by Algorithm 3.1. The above inequality becomes:

$$u(s_0, \chi) \geq -\epsilon + u(s_1, \zeta_{s_0,\chi}^{\alpha^\epsilon[b_0],b_0}(s_1)) \bigvee \Phi(\chi). \tag{3.25}$$

We set $a^\epsilon(\cdot) := \alpha^\epsilon[b_0](\cdot) \in \mathcal{A}$. On the other hand, from assumption **(H3.4)** the set $F(s_0, \chi, A, \mathsf{b}_0)$ is convex. Hence, there exists $a_0 \in A$ such that:

$$\int_{s_0}^{s_1} F(s_0, \chi, a^\epsilon(s), \mathsf{b}_0) ds = hF(s_0, \chi, a_0, \mathsf{b}_0).$$

Moreover, from the Lipschitz continuity of F there exists $\delta(h) \geq 0$, the modulus of continuity of F, defined as:

$$\delta(h) := \max \Big\{ \|F(s, \zeta, a, b) - F(s', \zeta, a, b)\|, \text{ for } \zeta \in \mathbb{B}(0, R),\ (a, b) \in A \times B \text{ and } s, s' \in [0, T] \text{ with } |s - s'| \leq h \Big\}.$$

The trajectory $\zeta_{s_0,\chi}^{a^\epsilon,b_0}(\cdot)$ verifies $\|\zeta_{s_0,\chi}^{a^\epsilon,b_0}(s) - \chi\| \leq M_R h$, for any $s \in [s_0, s_1]$, and

$$\begin{aligned}
\|\zeta_{s_0,\chi}^{a^\epsilon,b_0}(s_1) - \chi - hF(s_0, \chi, a_0, \mathsf{b}_0)\| &\leq \int_{s_0}^{s_1} \|F(s, \zeta_{s_0,\chi}^{a^\epsilon,b_0}(s), a^\epsilon(s), \mathsf{b}_0) - F(s_0, \chi, a^\epsilon(s), \mathsf{b}_0)\| ds \\
&\leq \int_{s_0}^{s_1} \|F(s, \zeta_{s_0,\chi}^{a^\epsilon,b_0}(s), a^\epsilon(s), \mathsf{b}_0) - F(s, \chi, a^\epsilon(s), \mathsf{b}_0)\| ds \\
&+ \int_{s_0}^{s_1} \|F(s, \chi, a^\epsilon(s), \mathsf{b}_0) - F(s_0, \chi, a^\epsilon(s), \mathsf{b}_0)\| ds \\
&\leq \int_{s_0}^{s_1} L_F(R) \|\zeta_{s_0,\chi}^{a^\epsilon,b_0}(s) - \chi\| ds + h\delta(h) \\
&\leq L_F(R) M_R h^2 + h\delta(h).
\end{aligned}$$

From the last inequality and the Lipschitz continuity of u, we get:

$$u(s_1, \zeta') - hL_u(R) \times (\delta(h) + L_F(R) M_R h) \leq u(s_1, \zeta_{s_0,\chi}^{a^\epsilon,b_0}(s_1)), \tag{3.26}$$

where $\zeta' := \chi + hF(s_0, \chi, a_0, \mathsf{b}_0)$ and $L_u(R)$ denotes the Lipschitz constant of u. We set

$$\epsilon_h := L_u(R) \times (\delta(h) + L_F(R) M_R h).$$

Therefore from (3.25) and (3.26), we deduce:

$$u(s_0, \chi) \geq -\epsilon - h\epsilon_h + u(s_1, \zeta') \bigvee \Phi(\chi),$$

where the last inequality holds since $(a_1 - a_3) \bigvee a_2 \geq a_1 \bigvee a_2 - a_3$, for any $a_1, a_2, a_3 \in \mathbb{R}$ s.t. $a_3 \geq 0$. Now, by using the fact that $\|u_h - u\| = E_h$, we deduce from the last inequality:

$$u_h(s_0, \chi) \geq -\epsilon - h\epsilon_h - 2E_h + u_h(s_1, \zeta') \bigvee \Phi(\chi).$$

From Algorithm 3.1 and since a_0^* is a minimizer of $a \mapsto u_h(s_1, \chi + hF(s_0, \chi, a, \mathsf{b}_0)) \bigvee \Phi(\chi)$, we obtain:

$$u_h(s_1, \zeta_1) \bigvee \Phi(\chi) \leq u_h(s_1, \zeta') \bigvee \Phi(\chi),$$

where ζ_1 is defined in Algorithm 3.1. Therefore by exploiting the two above inequalities, we get:

$$u_h(s_0, \chi) \geq -\epsilon - h\epsilon_h - 2E_h + u_h(s_1, \zeta_1) \bigvee \Phi(\chi),$$

which concludes (3.24) because ϵ is chosen arbitrarily.

Finally, the inequality (3.24) can be generalized by the same arguments to obtain for any $k = 0, ..., N-1$:

$$u_h(s_k, \zeta_k) \geq u_h(s_{k+1}, \zeta_{k+1}) \bigvee \Phi(\zeta_k) - h\epsilon_h - 2E_h, \tag{3.27}$$

where ζ_k is the system state at time step s_k, generated by Algorithm 3.1.

Step 2. From (3.27) and by using the fact that $(a_1 - a_3) \bigvee a_2 \geq a_1 \bigvee a_2 - a_3$, for any $a_1, a_2, a_3 \in \mathbb{R}$ s.t. $a_3 \geq 0$, we get by induction:

$$u_h(s_0, \chi) \geq \Big(u_h(s_N, \zeta_N) \bigvee \Phi(\zeta_{N-1}) \bigvee \cdots \bigvee \Phi(\chi)\Big) - Nh\epsilon_h - 2NE_h. \tag{3.28}$$

Recall that E_h is the uniform error between u and u_h. Therefore, we deduce from (3.28):

$$u_h(s_0, \chi) \geq (u(s_N, \zeta_N) - E_h) \bigvee \Big(\max_{0 \leq k \leq N-1} \Phi(\zeta_k)\Big) - Nh\epsilon_h - 2NE_h.$$

Since $s_N = Nh = T$ and $u(T, \zeta_N) = \Phi(T, \zeta_N) \bigvee \Psi(\zeta_N)$, the last inequality becomes:

$$u_h(s_0, \chi) \geq \Big(\max_{0 \leq k \leq N} \Phi(\zeta_k)\Big) \bigvee \Psi(\zeta_N) - T\epsilon_h - (\frac{2T}{h} + 1)E_h.$$

By hypothesis **(H3.5)**, we conclude that:

$$\limsup_{h \to 0^+} \Big(\max_{0 \leq k \leq N} \Phi(\zeta_k)\Big) \bigvee \Psi(\zeta_N) \leq u(0, \chi). \tag{3.29}$$

Step 3. In this step, we will establish an estimation between $(\zeta_k)_k$, the trajectory generated by Algorithm 3.1, and $\zeta_h(\cdot)$, the solution of (3.22).

We claim that for any $k = 0, ..., N-1$, we have:

$$\max_{s \in [s_k, s_{k+1}]} \|\zeta_h(s) - \zeta_k\| \leq \mathcal{O}(h) \quad \text{and} \quad \|\zeta_h(T) - \zeta_N\| \leq \mathcal{O}(h). \tag{3.30}$$

We start by proving the claim by induction. For $k = 0$, we have for any $s \in [s_0, s_1]$:

$$\|\zeta_h(s) - \chi\| \leq \int_{s_0}^{s} \|F(\theta, \zeta_h(\theta), \mathsf{a}_0^*, \mathsf{b}_0)\| d\theta \leq M_R(s - s_0) \leq M_R h = \mathcal{O}(h),$$

which gives the result for $k = 0$, *i.e.* $\max_{s \in [s_0, s_1]} \|\zeta_h(s) - \chi\| \leq \mathcal{O}(h)$.

Suppose that (3.30) is verified for $k \leq N-2$, and let's prove it for $k+1$. For any $s \in [s_{k+1}, s_{k+2}]$:

$$\begin{aligned} \|\zeta_h(s) - \zeta_{k+1}\| &= \|\zeta_h(s_{k+1}) + \int_{s_{k+1}}^{s} F(\theta, \zeta_h(\theta), \mathsf{a}_{k+1}^*, \mathsf{b}_{k+1}) d\theta - hF(s_k, \zeta_k, \mathsf{a}_k^*, \mathsf{b}_k) - \zeta_k\| \\ &\leq 2M_R h + \|\zeta_h(s_{k+1}) - \zeta_k\|. \end{aligned}$$

Using the estimation verified by $\|\zeta(s_{k+1}) - \zeta_k\|$ concludes the proof. Now, let's prove that $\|\zeta_h(T) - \zeta_N\| \leq \mathcal{O}(h)$. Indeed, we have:

$$\|\zeta_h(T) - \zeta_N\| \leq \|\zeta_h(T) - \zeta_{N-1}\| + h\|F(s_{N-1}, \zeta_{N-1}, \mathsf{a}^*_{N-1}, \mathsf{b}_{N-1})\|.$$

Moreover, we have proven that $\|\zeta_h(s) - \zeta_k\| \leq \mathcal{O}(h)$, for any $k = 0, ..., N-1$ and $s \in [s_k, s_{k+1}]$. In particular, for $k = N-1$ and $s = T$, we have $\|\zeta_h(T) - \zeta_{N-1}\| \leq \mathcal{O}(h)$. Finally, since $\|F(s_{N-1}, \zeta_{N-1}, \mathsf{a}^*_{N-1}, \mathsf{b}_{N-1})\| \leq M_R$, we conclude:

$$\|\zeta_h(T) - \zeta_N\| \leq \mathcal{O}(h) + M_R h = \mathcal{O}(h).$$

Step 4. Now, let's prove that:

$$|\max_{0\leq k\leq N} \Phi(\zeta_k) - \max_{s\in[0,T]} \Phi(\zeta_h(s))| \leq \mathcal{O}(h).$$

Indeed, we have:

$$\begin{aligned}
|\max_{0\leq k\leq N} \Phi(\zeta_k) - \max_{s\in[0,T]} \Phi(\zeta_h(s))| &= |\max_{0\leq k\leq N-1} \Phi(\zeta_k) \bigvee \Phi(\zeta_N) \\
&\quad - \max_{0\leq k\leq N-1} \Big(\max_{s\in[s_k,s_{k+1}]} \Phi(\zeta_h(s))\Big) \bigvee \Phi(\zeta_h(T))| \\
&\leq |\max_{0\leq k\leq N-1} \Phi(\zeta_k) - \max_{0\leq k\leq N-1} \Big(\max_{s\in[s_k,s_{k+1}]} \Phi(\zeta_h(s))\Big)| \\
&\quad \bigvee |\Phi(\zeta_N) - \Phi(\zeta_h(T))|,
\end{aligned}$$

where the last line holds since $|a \bigvee b - c \bigvee d| \leq |a - c| \bigvee |b - d|$, for any $a, b, c, d \in \mathbb{R}$. First, we have:

$$|\Phi(\zeta_N) - \Phi(\zeta_h(T))| \leq L_\Phi(R)\|\zeta_N - \zeta_h(T)\|,$$

where $L_\Phi(R)$ is the Lipschitz constant of Φ. From **Step 3.**, we get $|\Phi(\zeta_N) - \Phi(\zeta_h(T))| \leq \mathcal{O}(h)$. Moreover,

$$\begin{aligned}
|\max_{0\leq k\leq N-1} \Phi(\zeta_k) - \max_{0\leq k\leq N-1} \Big(\max_{s\in[s_k,s_{k+1}]} \Phi(\zeta_h(s))\Big)| &\leq \max_{0\leq k\leq N-1} |\Phi(\zeta_k) - \max_{s\in[s_k,s_{k+1}]} \Phi(\zeta_h(s))| \\
&\leq \max_{0\leq k\leq N-1} \max_{s\in[s_k,s_{k+1}]} |\Phi(\zeta_k) - \Phi(\zeta_h(s))| \\
&\leq \max_{0\leq k\leq N-1} \max_{s\in[s_k,s_{k+1}]} L_\Phi(R)\|\zeta_k - \zeta_h(s)\|.
\end{aligned}$$

From **Step 3.**, we already know that:

$$\max_{s\in[s_k,s_{k+1}]} \|\zeta_h(s) - \zeta_k\| \leq \mathcal{O}(h),$$

henceforth we deduce:

$$|\max_{0\leq k\leq N} \Phi(\zeta_k) - \max_{s\in[0,T]} \Phi(\zeta_h(s))| \leq \mathcal{O}(h).$$

Finally, by using again the fact that $\|\zeta_h(T) - \zeta_N\| \leq \mathcal{O}(h)$, we get:

$$|\Big(\max_{0\leq k\leq N} \Phi(\zeta_k)\Big) \bigvee \Psi(\zeta_N) - \Big(\max_{s\in[0,T]} \Phi(\zeta_h(s))\Big) \bigvee \Psi(\zeta_h(T))| \leq \mathcal{O}(h). \tag{3.31}$$

By combining the estimates (3.29) and (3.31), we obtain:

$$\limsup_{h\to 0^+} \Big\{ \Big(\max_{s\in[0,T]} \Phi(\zeta_h(s))\Big) \bigvee \Psi(\zeta_h(T)) \Big\} \leq u(0, \chi).$$

□

Remark 3.4.2. *In practice, the approximated value function u_h that will be used later (in the next section 3.5) comes from the numerical resolution of a discretized form of the Hamilton-Jacobi equation (3.19) whose unique viscosity solution is u.*

3.4.2 Reconstruction with a *specific* approximation

In this part, we extend the results about the approximation by discrete time games presented in [11, Chapter VIII] to our case with finite time horizon and maximum running cost functions. We propose a specific approximation u_h of the value function u which verifies a discrete dynamic programming principle. Then, we prove the existence of an optimal strategy of the first player and an optimal control of the second player associated with the discrete time game. Henceforth, we deduce an approximation of optimal feedbacks for problem (3.17).

Following the formulation presented in [11, Chapter VIII], a discrete nonanticpative strategy of the first player can be mathematically formulated as follows:

Definition 3.4.3. *A discrete nonanticpative strategy of the first player $\alpha_h[\cdot]$ is a mapping from B^N to A^N, such that $\forall\, j \in \{0, ..., N-1\}$ and for any $(\mathsf{b}_i)_i, (\mathsf{b}'_i)_i \in B^N$, if $\mathsf{b}_i = \mathsf{b}'_i\ \forall\, i \leq j$, then $\alpha_h[\mathsf{b}]_i = \alpha_h[\mathsf{b}']_i, \forall i \leq j$.*

Let Γ_h denote the set of discrete nonanticipative strategies of the first player.

Let J_h be an approximation of the cost functional J defined, for $t \in [s_k, s_{k+1}[$ with $k \in \{0, ..., N-1\}$, $\chi \in \mathbb{R}^m$, $\mathsf{a} := (\mathsf{a}_i)_i \in A^{N-k}$ and $\mathsf{b} := (\mathsf{b}_i)_i \in B^{N-k}$, by:

$$J_h(t, \chi, \mathsf{a}, \mathsf{b}) := \Phi(\chi) \bigvee \Big(\max_{i=k+1,...,N} \Phi(\zeta_i) \Big) \bigvee \Psi(\zeta_N),$$

where $(\zeta_i)_i$ is the solution of (3.21) associated to the control sequences $((\mathsf{a}_i)_i, (\mathsf{b}_i)_i)$.

Now, we define the following specific approximation u_h of the value function u on $[0, T] \times \mathbb{R}^m$:

$$u_h(t, \chi) := \inf_{\alpha_h[\cdot] \in \Gamma_h} \sup_{(\mathsf{b}_i)_i \in B^{N-k}} J_h(t, \chi, \alpha_h[\mathsf{b}], \mathsf{b}) \tag{3.32}$$

if $t \in [s_k, s_{k+1}[$, for $k \in \{0, ..., N-1\}$, and with the terminal condition $u_h(T, \chi) := \Phi(\chi) \bigvee \Psi(\chi)$.

Finally, we present Algorithm 3.2 corresponding to a reconstruction procedure based on the value function u_h in the worst case where the second player takes optimal decisions which corresponds to the worst situation for the first player. This algorithm is presented in a general form for some initial time instant $t \in [0, T[$ and from an initial position $\chi \in \mathbb{R}^m$.

The following Proposition presents some results verified by the approximated value function u_h, the reconstructed trajectory $(\zeta_i^*)_i$, the discrete strategy $\alpha_h^*[\cdot] \in \Gamma_h$ and the control $(\mathsf{b}_i^*)_i \in B^{N-k}$ generated by Algorithm 3.2.

Proposition 3.4.4. *For $t \in [s_k, s_{k+1}[$ with $k \in \{0, ..., N-1\}$ and $\chi \in \mathbb{R}^m$, we have:*

(i) u_h verifies the following discrete dynamic programming principle:

$$u_h(t, \chi) = \max_{b \in B} \min_{a \in A} \Big\{ u_h(s_{k+1}, \chi + (s_{k+1} - t) F(t, \chi, a, b)) \bigvee \Phi(\chi) \Big\} \tag{3.33}$$

(ii) Furthermore, $J_h(t, \chi, \alpha_h^*[\mathsf{b}], \mathsf{b}) \leq u_h(t, \chi)$ for any $(\mathsf{b}_i)_i \in B^{N-k}$. The equality holds when $(\mathsf{b}_i)_i = (\mathsf{b}_i^*)_i$.

(iii) Finally, u_h converges to u, when $h \to 0$, over compact subsets of $[0, T] \times \mathbb{R}^m$.

Remark 3.4.5. *$(\alpha_h^*[\cdot], (\mathsf{b}_i^*)_i)$ represents a Nash equilibrium for (3.32) in the terminology of the theory of noncooperative games. This means that every player cannot improve his guaranteed outcome, given by $u_h(t, \chi)$, by any unilateral deviation from his optimal choice, $\alpha_h^*[\cdot]$ and $(\mathsf{b}_i^*)_i$ for the first and the second players respectively.*

Proof of Proposition 3.4.4. (i) In order to prove the discrete dynamic programming principle, we will need the continuity of u_h in its space variable. To this end, we start by proving that u_h is locally Lipschitz

Algorithm 3.2: Worst case

Require: $t \in [s_k, s_{k+1}[$, for $0 \leq k \leq N-1$ and $\chi \in \mathbb{R}^m$.

1: Initialise $\zeta_k^* = \chi$.

2: **for** $i = k, ..., N-1$ **do**

3: The optimal choice of the second player $\mathsf{b}_i^* \in B$ is

$$\mathsf{b}_i^* \in \begin{cases} \underset{b\in B}{\text{argmax}} \min\limits_{a\in A}\Big\{u_h(s_{k+1}, \chi + (s_{k+1}-t)F(t,\chi,a,b)) \bigvee \Phi(\chi)\Big\}, \text{ if } i = k, \\ \underset{b\in B}{\text{argmax}} \min\limits_{a\in A}\Big\{u_h(s_{i+1}, \zeta_i^* + hF(s_i,\zeta_i^*,a,b)) \bigvee \Phi(\zeta_i^*)\Big\}, \text{ else.} \end{cases}$$

4: The optimal reaction of the first player $\alpha_h^*[\mathsf{b}^*]_i := \mathsf{a}_i^*$ is

$$\mathsf{a}_i^* \in \begin{cases} \underset{a\in A}{\text{argmin}}\Big\{u_h(s_{k+1}, \chi + (s_{k+1}-t)F(t,\chi,a,\mathsf{b}_k^*)) \bigvee \Phi(\chi)\Big\}, \text{ if } i = k, \\ \underset{a\in A}{\text{argmin}}\Big\{u_h(s_{i+1}, \zeta_i^* + hF(s_i,\zeta_i^*,a,\mathsf{b}_i^*)) \bigvee \Phi(\zeta_i^*)\Big\}, \text{ else.} \end{cases}$$

5: The new state position:

$$\begin{cases} \zeta_{k+1}^* = \chi + (s_{k+1}-t)F(t,\chi,\mathsf{a}_k^*,\mathsf{b}_k^*), \text{ if } i = k, \\ \zeta_{i+1}^* = \zeta_i^* + hF(s_i,\zeta_i^*,\mathsf{a}_i^*,\mathsf{b}_i^*), \text{ else.} \end{cases}$$

6: **end for**

7: **return** A discrete strategy $\alpha_h^*[\cdot]$ of the first player, a discrete control $(\mathsf{b}_i^*)_i$ of the second player and a discrete trajectory $(\zeta_i^*)_i$.

continuous w.r.t. the space variable uniformly in the time variable. We fix $t \in [0, T[$ and let $0 \leq k \leq N-1$ such that $t \in [s_k, s_{k+1}[$ and x and y be two arbitrary vectors of $\mathbb{R}^m$ such that $\|x\|, \|y\| \leq R$ with some $R > 0$.

For $\epsilon > 0$, there exists $\alpha_h^\epsilon[\cdot] \in \Gamma_h$ such that:

$$u_h(t,y) \geq \sup_{(\mathsf{b}_i)_i \in B^{N-k}} J_h(t,y,\alpha_h^\epsilon[\mathsf{b}],\mathsf{b}) - \frac{\epsilon}{2}.$$

On the other hand, there exists $(\mathsf{b}_i^\epsilon)_i \in B^{N-k}$ such that:

$$u_h(t,x) \leq \sup_{(\mathsf{b}_i)_i \in B^{N-k}} J_h(t,x,\alpha_h^\epsilon[\mathsf{b}],\mathsf{b}) \leq J_h(t,x,\alpha_h^\epsilon[\mathsf{b}^\epsilon],\mathsf{b}^\epsilon) + \frac{\epsilon}{2}.$$

Denote by $(\mathsf{a}_i^\epsilon)_i := \alpha_h^\epsilon[\mathsf{b}^\epsilon] \in A^{N-k}$. From the first inequality involving $u_h(t,y)$, we deduce that:

$$u_h(t,y) \geq J_h(t,y,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon) - \frac{\epsilon}{2}.$$

Now from the two above inequalities, we get:

$$u_h(t,x) - u_h(t,y) \leq J_h(t,x,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon) - J_h(t,y,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon) + \epsilon.$$

Denote by $(\zeta_i^x)_i$ and $(\zeta_i^y)_i$ the solutions of (3.21) corresponding to $((\mathsf{a}_i^\epsilon)_i, (\mathsf{b}_i^\epsilon)_i) \in A^{N-k} \times B^{N-k}$ and starting respectively from x and y. We have the following estimation:

$$|J_h(t,x,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon) - J_h(t,y,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon)| \leq |\Phi(x) - \Phi(y)| \bigvee \Big(\max_{i=k+1,...,N} |\Phi(\zeta_i^x) - \Phi(\zeta_i^y)|\Big) \bigvee |\Psi(\zeta_N^x) - \Psi(\zeta_N^y)|.$$

Now we use the Lipschitz continuity of Φ and Ψ to deduce that:

$$|J_h(t,x,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon) - J_h(t,y,\mathsf{a}^\epsilon,\mathsf{b}^\epsilon)| \leq L_\Phi(R)\|x-y\| \bigvee L_\Phi(R)\Big(\max_{i=k+1,\dots,N}\|\zeta_i^x - \zeta_i^y\|\Big) \bigvee L_\Psi(R)\|\zeta_N^x - \zeta_N^y\|,$$

where $L_\Phi(R)$ and $L_\Psi(R)$ are respectively the Lipschitz constants of Φ and Ψ. Moreover, by the Lipschitz continuity of F, with Lipschitz constant $L_F(R)$, one can prove the following estimation on the discrete trajectories $(\zeta_i^x)_i$ and $(\zeta_i^y)_i$:

$$\|\zeta_i^x - \zeta_i^y\| \leq (1+hL_F(R))^{i-k}\|x-y\|, \quad \text{for any } \ i \geq k+1.$$

We conclude that there exists some constant $C(R) > 0$ such that:

$$|u_h(t,x) - u_h(t,y)| \leq C(R)\|x-y\| + \epsilon.$$

The fact that ϵ is chosen arbitrarily gives the desired result.

Now we will prove the discrete dynamic programming equation (3.33). To this end, we define for $t \in [s_k, s_{k+1}[$ and $\chi \in \mathbb{R}^m$

$$\rho(t,\chi) := \max_{b\in B}\min_{a\in A}\Big\{u_h(s_{k+1}, \chi + (s_{k+1}-t)F(t,\chi,a,b)) \bigvee \Phi(\chi)\Big\}.$$

We start by showing that $u_h(t,\chi) \leq \rho(t,\chi)$. Since u_h is continuous w.r.t. the space variable, for any $b \in B$ there exists $a(b) \in A$, depending on b, such that:

$$\begin{aligned}\rho(t,\chi) &\geq \min_{a\in A}\Big\{u_h(s_{k+1}, \chi + (s_{k+1}-t)F(t,\chi,a,b)) \bigvee \Phi(\chi)\Big\} \\ &\geq u_h(s_{k+1}, \chi + (s_{k+1}-t)F(t,\chi,a(b),b)) \bigvee \Phi(\chi).\end{aligned}$$

Let $(\mathsf{b}_i)_i \in B^{N-k}$ be a discrete control of the second player and define χ' the system state at time s_{k+1}, $\chi' := \chi + (s_{k+1}-t)F(t,\chi,a(\mathsf{b}_k),\mathsf{b}_k)$. For any $\epsilon > 0$, consider a discrete strategy $\alpha_h^\epsilon[\cdot] \in \Gamma_h$ verifying:

$$\epsilon + u_h(s_{k+1},\chi\prime) \bigvee \Phi(\chi) \geq \sup_{(\mathsf{b}'_i)_i \in B^{N-k-1}} \Big\{J_h(s_{k+1},\chi',\alpha_h^\epsilon[\mathsf{b}'],\mathsf{b}') \bigvee \Phi(\chi)\Big\}.$$

Let $\delta_h[\cdot] \in \Gamma_h$ be defined, for $(\mathsf{b}_i)_i \in B^{N-k}$, as follows:

$$\delta_h[\mathsf{b}]_i = \begin{cases} a(\mathsf{b}_k), & i = k \\ \alpha_h^\epsilon[\mathsf{b}]_i, & i \geq k+1. \end{cases}$$

For any $(\mathsf{b}_i)_i \in B^{N-k}$ we have:

$$J_h(t,\chi,\delta_h[\mathsf{b}],\mathsf{b}) = J_h(s_{k+1},\chi',\alpha_h^\epsilon[\mathsf{b}'],\mathsf{b}') \bigvee \Phi(\chi),$$

where $(\mathsf{b}'_i)_i \in B^{N-k-1}$ is the restriction of $(\mathsf{b}_i)_i$ to $i \geq k+1$. We deduce that:

$$\begin{aligned}u_h(t,\chi) &\leq \sup_{(\mathsf{b}_i)_i \in B^{N-k}} J_h(t,\chi,\delta_h[\mathsf{b}],\mathsf{b}) \\ &\leq \sup_{\mathsf{b}_k\in B}\sup_{(\mathsf{b}'_i)_i\in B^{N-k-1}}\Big\{J_h(s_{k+1},\chi',\alpha_h^\epsilon[\mathsf{b}'],\mathsf{b}') \bigvee \Phi(\chi)\Big\} \\ &\leq \epsilon + \sup_{\mathsf{b}_k\in B}\Big\{u_h(s_{k+1},\chi') \bigvee \Phi(\chi)\Big\}.\end{aligned}$$

On the other hand, we have

$$\sup_{\mathsf{b}_k\in B}\Big\{u_h(s_{k+1},\chi')\bigvee\Phi(\chi)\Big\}=\max_{\mathsf{b}_k\in B}\Big\{u_h(s_{k+1},\chi+(s_{k+1}-t)F(t,\chi,a(\mathsf{b}_k),\mathsf{b}_k))\bigvee\Phi(\chi)\Big\}$$
$$=\rho(t,\chi)$$

therefore for any $\epsilon>0$, we conclude that $u_h(t,\chi)\leq\epsilon+\rho(t,\chi)$. Hence, we obtain the desired inequality since ϵ is chosen arbitrarily.

Now, we will show that $u_h(t,\chi)\geq\rho(t,\chi)$. Let $\bar{b}\in B$ such that:

$$\rho(t,\chi)=\min_{a\in A}\Big\{u_h(s_{k+1},\chi+(s_{k+1}-t)F(t,\chi,a,\bar{b}))\bigvee\Phi(\chi)\Big\}.$$

For $\epsilon>0$, let $\alpha_h^\epsilon[\cdot]\in\Gamma_h$ be an ϵ-optimal discrete strategy for $u_h(t,\chi)$ i.e.

$$\epsilon+u_h(t,\chi)=\sup_{(\mathsf{b}_i)_i\in B^{N-k}}J_h(t,\chi,\alpha_h^\epsilon[\mathsf{b}],\mathsf{b}).\tag{3.34}$$

For any discrete control $(\mathsf{b}_i)_i\in B^{N-k}$, we define the following control sequence $(\hat{\mathsf{b}}_i)_i\in B^{N-k}$ by:

$$\hat{\mathsf{b}}_i:=\begin{cases}\bar{b},\ \text{if } i=k,\\ \mathsf{b}_i,\ \text{if } i\geq k+1.\end{cases}$$

Since $\alpha_h^\epsilon[\cdot]$ is a discrete nonanticipative strategy, $\alpha_h^\epsilon[\hat{\mathsf{b}}]_k$ depends only on $\bar{b}$. Let $\bar{a}\in A$ and $\chi'\in\mathbb{R}^m$ be given by:

$$\bar{a}:=\alpha_h^\epsilon[\hat{\mathsf{b}}]_k\quad\text{and}\quad\chi':=\chi+(s_{k+1}-t)F(t,\chi,\bar{a},\bar{b}).$$

Finally, we define the strategy $\delta_h[\cdot]\in\Gamma_h$, for $(\mathsf{b}_i)_i\in B^{N-k-1}$, by:

$$\delta_h[\mathsf{b}]_i:=\alpha_\epsilon[\hat{\mathsf{b}}]_i,\ \text{ for } i\geq k+1.$$

We have

$$\begin{aligned}u_h(s_{k+1},\chi')\bigvee\Phi(\chi)&=\inf_{\alpha_h[\cdot]\in\Gamma_h}\sup_{(\mathsf{b}_i)_i\in B^{N-k-1}}\Big\{J_h(s_{k+1},\chi',\alpha_h[\mathsf{b}],\mathsf{b})\bigvee\Phi(\chi)\Big\}\\&\leq\sup_{(\mathsf{b}_i)_i\in B^{N-k-1}}\Big\{J_h(s_{k+1},\chi',\delta_h[\mathsf{b}],\mathsf{b})\bigvee\Phi(\chi)\Big\}\\&\leq J_h(s_{k+1},\chi',\delta_h[\mathsf{b}^\epsilon],\mathsf{b}^\epsilon)\bigvee\Phi(\chi)+\epsilon\end{aligned}$$

where $(\mathsf{b}_i^\epsilon)_i$ is an ϵ-optimal sequence for $\sup_{(\mathsf{b}_i)_i\in B^{N-k-1}}J_h(s_{k+1},\chi',\delta_h[\mathsf{b}],\mathsf{b})\bigvee\Phi(t,\chi)$.

Claim that

$$\rho(t,\chi)\leq J_h(t,\chi,\alpha_h^\epsilon[\hat{\mathsf{b}}^\epsilon],\hat{\mathsf{b}}^\epsilon)+\epsilon,$$

where

$$\hat{\mathsf{b}}_i^\epsilon:=\begin{cases}\bar{b},\ \text{if } i=k,\\ \mathsf{b}_i^\epsilon,\ \text{if } i\geq k+1.\end{cases}$$

From the above inequality, we obtain:

$$\rho(t,\chi)\leq\sup_{(\mathsf{b}_i)_i\in B^{N-k}}J_h(t,\chi,\alpha_h^\epsilon[\mathsf{b}],\mathsf{b})+\epsilon,$$

together with (3.34), we obtain $\rho(t,\chi)\leq u_h(t,\chi)+2\epsilon$, which ends the proof of (i) since ϵ is chosen arbitrarily.

Now let's prove the claim.

$$\begin{aligned}\rho(t,\chi) &= \max_{b\in B}\min_{a\in A}\Big\{u_h(s_{k+1},\chi+(s_{k+1}-t)F(t,\chi,a,b))\bigvee\Phi(\chi)\Big\}\\ &= \min_{a\in A}\Big\{u_h(s_{k+1},\chi+(s_{k+1}-t)F(t,\chi,a,\bar{b}))\bigvee\Phi(\chi)\Big\}\\ &\leq u_h(s_{k+1},\chi')\bigvee\Phi(\chi)\\ &\leq J_h(s_{k+1},\chi',\delta_h[\hat{\mathsf{b}}^\epsilon],\hat{\mathsf{b}}^\epsilon)\bigvee\Phi(\chi)+\epsilon.\end{aligned}$$

Combining the last inequality with the following equality justifies our claim:

$$J_h(s_{k+1},\chi',\delta_h[\hat{\mathsf{b}}^\epsilon],\hat{\mathsf{b}}^\epsilon)\bigvee\Phi(\chi)=J_h(t,\chi,\alpha_h^\epsilon[\hat{\mathsf{b}}^\epsilon],\hat{\mathsf{b}}^\epsilon).$$

(ii) From (i) and Algorithm 3.2, we obtain:

$$\begin{aligned}u_h(t,\chi) &= \max_{b\in B}\min_{a\in A}\Big\{u_h(s_{k+1},\chi+(s_{k+1}-t)F(t,\chi,a,b))\bigvee\Phi(\chi)\Big\}\\ &= \min_{a\in A}\Big\{u_h(s_{k+1},\chi+(s_{k+1}-t)F(s,\chi,a,\mathsf{b}_k^*))\bigvee\Phi(\chi)\Big\}\\ &= u_h(s_{k+1},\zeta_{k+1}^*)\bigvee\Phi(\chi).\end{aligned}$$

By the same argument, we get:

$$u_h(s_{k+1},\zeta_{k+1}^*)=u_h(s_{k+2},\zeta_{k+2}^*)\bigvee\Phi(\zeta_{k+1}^*),$$

which can be generalized for any $i\geq k+1$:

$$u_h(s_i,\zeta_i^*)=u_h(s_{i+1},\zeta_{i+1}^*)\bigvee\Phi(\zeta_i^*).$$

From the above equalities, we conclude:

$$u_h(t,\chi)=\Phi(\chi)\bigvee\Big(\max_{i=k+1,\dots,N}\Phi(\zeta_i^*)\Big)\bigvee\Psi(\zeta_N^*)=J_h(t,\chi,\alpha_h^*[\mathsf{b}^*],\mathsf{b}^*).$$

In a similar way, one can prove that for any $(\mathsf{b}_i)_i\in B^{N-k}$:

$$J_h(t,\chi,\alpha_h^*[\mathsf{b}],\mathsf{b})\leq u_h(t,\chi).$$

(iii) First, consider the function $\overline{u}$ defined by

$$\overline{u}(t,\chi):=\limsup_{(s,y)\to(t,\chi),h\to0^+}u_h(s,y).$$

Let's prove that $\overline{u}$ is a sub-solution of the HJ equation (3.19) whose unique viscosity solution is u. Indeed, let ξ be a function of class C^1 and $(\bar{t},\overline{\chi})$ be a strict maximum of $\overline{u}-\xi$ in $\overline{B}:=\overline{B}((\bar{t},\overline{\chi}),r)$, with $r>0$. From Lemma 2.6.3, there exist two sequences $(h_n)_n$ and $((t_n,\chi_n))_n$ such that for any $n\in\mathbb{N}$, $h_n>0$ and (t_n,χ_n) is a maximum point of $u_{h_n}-\xi$ over $\overline{B}$ and:

$$h_n\to0,\quad (t_n,\chi_n)\to(\bar{t},\overline{\chi}),\quad \text{and}\quad u_{h_n}(t_n,\chi_n)\to\overline{u}(\bar{t},\overline{\chi})\quad\text{when}\quad n\to+\infty.$$

Furthermore, consider a uniform partition of $[0,T]$ with time steps: $s_0^n=0,...,s_i^n=ih_n,...$ There exists $k\geq0$ such that $t_n\in[s_k^n,s_{k+1}^n[$ and let $\tau_n:=s_{k+1}^n-t_n>0$. From the discrete dynamic programming equation (3.33), there exists $b_n\in B$ $s.t.$ for any $a\in A$:

$$u_{h_n}(t_n,\chi_n)\leq u_h(s_{k+1}^n,y_n)\bigvee\Phi(\chi_n),$$

where $y_n := \chi_n + \tau_n F(t_n, \chi_n, a, b_n)$. We denote by $rhs(n)$ the right hand side of the above equation. There exists a subsequence of $((t_n, \chi_n))_n$, denoted by $((t_{\sigma(n)}, \chi_{\sigma(n)}))_n$, such that $rhs(n)$ is equal to either $u_h(s_{k+1}^{\sigma(n)}, y_{\sigma(n)})$ or $\Phi(\chi_{\sigma(n)})$.

If $rhs(n) = \Phi(\chi_{\sigma(n)})$, then $u_{h_{\sigma(n)}}(t_{\sigma(n)}, \chi_{\sigma(n)}) \leq \Phi(\chi_{\sigma(n)})$. When $n \to \infty$, we obtain $\overline{u}(\overline{t}, \overline{\chi}) \leq \Phi(\overline{\chi})$, which gives the result. In the other case, $rhs(n) = u_{h_{\sigma(n)}}(s_{k+1}^{\sigma(n)}, y_{\sigma(n)})$ and hence

$$u_{h_{\sigma(n)}}(t_{\sigma(n)}, \chi_{\sigma(n)}) \leq u_{h_{\sigma(n)}}(s_{k+1}^{\sigma(n)}, y_{\sigma(n)}).$$

On the other hand, we have

$$(u_{h_{\sigma(n)}} - \xi)(t_{\sigma(n)}, \chi_{\sigma(n)}) \geq (u_{h_{\sigma(n)}} - \xi)(s_{k+1}^{\sigma(n)}, y_{\sigma(n)}).$$

From the two last inequalities, we get

$$\xi(t_{\sigma(n)}, \chi_{\sigma(n)}) - \xi(s_{k+1}^{\sigma(n)}, y_{\sigma(n)}) \leq 0.$$

The Taylor expansion of ξ at $(t_{\sigma(n)}, \chi_{\sigma(n)})$ gives

$$\begin{aligned}\xi(s_{k+1}^{\sigma(n)}, y_{\sigma(n)}) = \xi(t_{\sigma(n)}, \chi_{\sigma(n)}) + \tau_{\sigma(n)} \frac{\partial \xi}{\partial t}(t_{\sigma(n)}, \chi_{\sigma(n)}) + \langle y_{\sigma(n)} - \chi_{\sigma(n)}, D_\chi \xi(t_{\sigma(n)}, \chi_{\sigma(n)}) \rangle \\ + \sqrt{\tau_{\sigma(n)}^2 + \|y_{\sigma(n)} - \chi_{\sigma(n)}\|^2} \times \epsilon\Big(\|(s_{k+1}^{\sigma(n)}, y_{\sigma(n)}) - (t_{\sigma(n)}, \chi_{\sigma(n)})\|\Big),\end{aligned}$$

where $\epsilon\Big(\|(s_{k+1}^{\sigma(n)}, y_{\sigma(n)}) - (t_{\sigma(n)}, \chi_{\sigma(n)})\|\Big) \to 0$ when $\|(s_{k+1}^{\sigma(n)}, y_{\sigma(n)}) - (t_{\sigma(n)}, \chi_{\sigma(n)})\| \to 0$. From the last inequality and the Taylor expansion, and after dividing by $\tau_{\sigma(n)}$, we obtain:

$$\begin{aligned}-\frac{\partial \xi}{\partial t}(t_{\sigma(n)}, \chi_{\sigma(n)}) - \langle F(t_{\sigma(n)}, \chi_{\sigma(n)}, a, b_{\sigma(n)}), D_\chi \xi(t_{\sigma(n)}, \chi_{\sigma(n)}) \rangle \\ + \sqrt{1 + \|F(t_{\sigma(n)}, \chi_{\sigma(n)}, a, b_{\sigma(n)})\|^2} \times \epsilon\Big(\|(s_{k+1}^{\sigma(n)}, y_{\sigma(n)}) - (t_{\sigma(n)}, \chi_{\sigma(n)})\|\Big) \leq 0.\end{aligned}$$

Since B is compact, we can extract a sequence from $(b_{\sigma(n)})_n$, denoted also by $(b_{\sigma(n)})_n$ for simplicity, that converges to some $\overline{b} \in B$ and when $n \to \infty$, we get:

$$-\frac{\partial \xi}{\partial t}(\overline{t}, \overline{\chi}) - \langle F(\overline{t}, \overline{\chi}, a, \overline{b}), D_\chi \xi(\overline{t}, \overline{\chi}) \rangle \leq 0.$$

Since the last inequality still true for any $a \in A$, we deduce that:

$$-\frac{\partial \xi}{\partial t}(\overline{t}, \overline{\chi}) + H(\overline{t}, \overline{\chi}, D_\chi \xi(\overline{t}, \overline{\chi})) \leq 0.$$

Finally, since

$$\min\Big(-\frac{\partial \xi}{\partial t}(\overline{t}, \overline{\chi}) + H(\overline{t}, \overline{\chi}, D_\chi \xi(\overline{t}, \overline{\chi})), \xi(\overline{t}, \overline{\chi}) - \Phi(\overline{\chi})\Big) \leq -\frac{\partial \xi}{\partial t}(\overline{t}, \overline{\chi}) + H(\overline{t}, \overline{\chi}, D_\chi \xi(\overline{t}, \overline{\chi}))$$

we conclude that $\overline{u}$ is a sub-solution of (3.19). Now, consider the function $\underline{u}$ defined by:

$$\underline{u}(t, \chi) := \liminf_{(s,y) \to (t,\chi), h \to 0^+} u_h(s, y),$$

and let's prove that $\underline{u}$ is a super-solution of (3.19). Let ξ be a function of class C^1 and $(\underline{t}, \underline{\chi})$ a strict minimum of $\underline{u} - \xi$ in $\overline{B} := \overline{B}((\underline{t}, \underline{\chi}), r)$, with $r > 0$, such that $(\underline{u} - \xi)(\underline{t}, \underline{\chi}) = 0$. From Lemma 2.6.3, there

exist two sequences $(h_n)_n$ and $((t_n, \chi_n))_n$ such that for any $n \in \mathbb{N}$, $h_n > 0$, (t_n, χ_n) is a minimum point of $u_{h_n} - \xi$ over $\overline{B}$ and:

$$h_n \to 0, \quad (t_n, \chi_n) \to (\underline{t}, \underline{\chi}) \quad \text{and} \quad u_{h_n}(t_n, \chi_n) \to \underline{u}(\underline{t}, \underline{\chi}) \quad \text{when} \quad n \to +\infty.$$

Consider again the uniform partition of $[0, T]$ with time steps: $s_0^n = 0, ..., s_i^n = ih_n, ...$ There exists $k \geq 0$ such that $t_n \in [s_k^n, s_{k+1}^n[$ and let $\tau_n := s_{k+1}^n - t_n > 0$.

For any $b \in B$, there exists $a_n \in A$ (depending on b) such that:

$$u_{h_n}(t_n, \chi_n) \geq u_{h_n}(s_{k+1}^n, \chi_n + \tau_n F(t_n, \chi_n, a_n, b)) \bigvee \Phi(\chi_n).$$

First, we have $u_{h_n}(t_n, \chi_n) \geq \Phi(\chi_n)$ and when $n \to \infty$, we obtain

$$\xi(\underline{t}, \underline{\chi}) = \underline{u}(\underline{t}, \underline{\chi}) \geq \Phi(\underline{\chi}). \tag{3.35}$$

Moreover, we have $u_{h_n}(t_n, \chi_n) \geq u_h(s_{k+1}^n, y_n)$ where $y_n := \chi_n + \tau_n F(t_n, \chi_n, a_n, b)$. Since (t_n, χ_n) is the minimum of $u_{h_n} - \xi$ over $\overline{B}$, we get:

$$\frac{\zeta(t_n, \chi_n) - \xi(s_{k+1}^n, y_n)}{\tau_n} \geq 0.$$

Now, we use the Taylor expansion of ξ at (t_n, χ_n), we extract a sequence from $(a_n)_n$ that converges to some $\overline{a} \in A$ and we let $n \to \infty$ to obtain:

$$-\frac{\partial \xi}{\partial t}(\underline{t}, \underline{\chi}) - \langle F(\underline{t}, \underline{\chi}, \overline{a}, b), D_\chi \xi(\underline{t}, \underline{\chi}) \rangle \geq 0,$$

therefore,

$$-\frac{\partial \xi}{\partial t}(\underline{t}, \underline{\chi}) + \max_{a \in A} - \langle F(\underline{t}, \underline{\chi}, a, b), D_\chi \xi(\underline{t}, \underline{\chi}) \rangle \geq 0.$$

Since the last inequality is verified for any $b \in B$, we deduce that:

$$-\frac{\partial \xi}{\partial t}(\underline{t}, \underline{\chi}) + H(\underline{t}, \underline{\chi}, D_\chi \xi(\underline{t}, \underline{\chi})) \geq 0.$$

Finally, from (3.35) and the last inequality we deduce that $\underline{u}$ is a super-solution of (3.19).

As a conclusion, we have shown that $\underline{u}$ and $\overline{u}$ are respectively super and sub-solution of (3.19). Therefore, by applying the comparison principle theorem that holds for (3.19) (see [5, Appendix]), we obtain $\overline{u} \leq u \leq \underline{u}$. Since in general $\underline{u} \leq \overline{u}$, we get $\overline{u} = u = \underline{u}$. Finally, by exploiting the properties of weak limits, see for instance [11, Chapter V], we deduce the uniform convergence of u_h to u over compact subsets of $[0, T] \times \mathbb{R}^m$.

□

Finally, we present the following result that defines a continuous time strategy for the first player and control for the second player, such that when the time step h goes to zero, the corresponding value of J converges to the continuous time value function u.

Theorem 3.4.6. *Let $t \in [0, T[$, there exist $(\tilde{\alpha}_h^*[\cdot], b_h^*(\cdot)) \in \Gamma \times \mathcal{B}$ verifying*

$$\lim_{h \to 0^+} J(t, \chi, \tilde{\alpha}_h^*[b_h^*], b_h^*) = u(t, \chi).$$

Proof. We define $\mathcal{B}_h$, a subset of $\mathcal{B}$, by:

$$\mathcal{B}_h := \Big\{ b(\cdot) \in \mathcal{B},\ \text{s.t. } b(s) = b(kh),\ \forall s \in [kh, (k+1)h[,\ \text{for } k = 0, ..., N-1 \Big\},$$

and let $\mathcal{A}_h$ be the subset of $\mathcal{A}$ defined in a similar way to $\mathcal{B}_h$.
Now, let's define $(\tilde{\alpha}_h^*[\cdot], b_h^*(\cdot)) \in \Gamma \times \mathcal{B}$ by:

$$b_h^*(s) := \mathsf{b}^*_{\lfloor s/h \rfloor} \quad \text{and} \quad \tilde{\alpha}_h^*[b](s) := \alpha_h^*[\hat{\mathsf{b}}]_{\lfloor s/h \rfloor}$$

where $\alpha_h^*[\cdot] \in \Gamma_h$ and $(\mathsf{b}_i^*)_i \in B^{N-k}$ are generated by Algorithm 3.2 and $(\hat{\mathsf{b}}_i)_i \in B^{N-k}$ is defined by $\hat{\mathsf{b}}_i := b(ih)$, for $i = k, ..., N-1$, for any $b(\cdot) \in \mathcal{B}$. Notice that $b_h^*(\cdot) \in \mathcal{B}_h$ and $\tilde{\alpha}_h^*[b](\cdot) \in \mathcal{A}_h$, for any $b(\cdot) \in \mathcal{B}$. Furthermore, for any $(a(\cdot), b(\cdot)) \in \mathcal{A}_h \times \mathcal{B}_h$, we define $((\hat{\mathsf{a}}_i)_i, (\hat{\mathsf{b}}_i)_i) \in A^{N-k} \times B^{N-k}$ by:

$$\hat{\mathsf{a}}_i = a(ih) \quad \text{and} \quad \hat{\mathsf{b}}_i = b(ih) \quad \text{for} \quad i = k, ..., N-1.$$

We claim that:

$$|J(t, \chi, a, b) - J_h(t, \chi, \hat{\mathsf{a}}, \hat{\mathsf{b}})| \leq \mathcal{O}(h), \tag{3.36}$$

where J is given by the following expression:

$$J(t, \chi, a, b) = \Big(\max_{s \in [t,T]} \Phi(\zeta_{t,\chi}^{a,b}(s)) \Big) \bigvee \Psi(\zeta_{t,\chi}^{a,b}(T)),$$

for any $(a(\cdot), b(\cdot)) \in \mathcal{A} \times \mathcal{B}$ and J_h, for any $((\mathsf{a}_i)_i, (\mathsf{b}_i)_i) \in A^{N-k} \times B^{N-k}$, is given by:

$$J_h(t, \chi, \mathsf{a}, \mathsf{b}) = \Phi(\chi) \bigvee \Big(\max_{i=k+1,...,N} \Phi(\zeta_i) \Big) \bigvee \Psi(\zeta_N).$$

First, the estimation (3.36) implies

$$|J(t, \chi, \tilde{\alpha}_h^*[b_h^*], b_h^*) - J_h(t, \chi, \alpha_h^*[\mathsf{b}^*], \mathsf{b}^*)| \leq \mathcal{O}(h). \tag{3.37}$$

Then, from (ii) and (iii) of Proposition 3.4.4 we have:

$$J_h(t, \chi, \alpha_h^*[\mathsf{b}^*], \mathsf{b}^*) = u_h(t, \chi) \quad \text{and} \quad \lim_{h \to 0^+} u_h(t, \chi) = u(t, \chi).$$

Finally, combining the two above equalities with inequality (3.37) gives the desired result. Now, we will justify the claim (3.36). We have:

$$|J(t, \chi, a, b) - J_h(t, \chi, \hat{\mathsf{a}}, \hat{\mathsf{b}})| \leq |\max_{s \in [t, s_{k+1}]} \Phi(\zeta_{t,\chi}^{a,b}(s)) - \Phi(\chi)| \bigvee \max_{i=k+1,...,N-1} |\max_{s \in [s_i, s_{i+1}]} \Phi(\zeta_{t,\chi}^{a,b}(s)) - \Phi(\zeta_i)|$$
$$\bigvee |\Phi(\zeta_{t,\chi}^{a,b}(T)) - \Phi(\zeta_N)| \bigvee |\Psi(\zeta_{t,\chi}^{a,b}(T)) - \Psi(\zeta_N)|.$$

Moreover, we have:

$$|\max_{s \in [t, s_{k+1}]} \Phi(\zeta_{t,\chi}^{a,b}(s)) - \Phi(\chi)| \leq \max_{s \in [t, s_{k+1}]} |\Phi(\zeta_{t,\chi}^{a,b}(s)) - \Phi(\chi)| \leq \max_{s \in [t, s_{k+1}]} L_\Phi(R) \|\zeta_{t,\chi}^{a,b}(s) - \chi\|,$$

and for any $i = k+1, ..., N-1$:

$$|\max_{s \in [s_i, s_{i+1}]} \Phi(\zeta_{t,\chi}^{a,b}(s)) - \Phi(\zeta_i)| \leq \max_{s \in [s_i, s_{i+1}]} |\Phi(\zeta_{t,\chi}^{a,b}(s)) - \Phi(\zeta_i)| \leq \max_{s \in [s_i, s_{i+1}]} L_\Phi(R) \|\zeta_{t,\chi}^{a,b}(s) - \zeta_i\|,$$

and finally:

$$|\Phi(\zeta_{t,\chi}^{a,b}(T)) - \Phi(\zeta_N)| \leq L_\Phi(R) \|\zeta_{t,\chi}^{a,b}(T) - \zeta_N\| \quad \text{and} \quad |\Psi(\zeta_{t,\chi}^{a,b}(T)) - \Psi(\zeta_N)| \leq L_\Psi(R) \|\zeta_{t,\chi}^{a,b}(T) - \zeta_N\|.$$

Then, by following the same arguments used in **Step 3.** and **Step 4.** from the proof of Theorem 3.4.1 and by exploiting the Lipschitz continuity of F, Φ and Ψ, one can prove that:

$$\|\zeta_{t,\chi}^{a,b}(s) - \chi\| \leq \mathcal{O}(h), \quad \text{for any} \quad s \in [t, s_{k+1}],$$

and for any $i = k+1, ..., N-1$:

$$\|\zeta_{t,\chi}^{a,b}(s) - \zeta_i\| \leq \mathcal{O}(h), \quad \text{for any} \quad s \in [s_i, s_{i+1}],$$

and

$$\|\zeta_{t,\chi}^{a,b}(T) - \zeta_N\| \leq \mathcal{O}(h).$$

Combining the above estimations ends the proof of the claim (3.36).

□

Even for a trajectory reconstructed by Algorithm 3.2 (the worst case), we cannot prove the equality in (3.23) (see Theorem 3.4.1) when the approximation u_h is general. In other words, one cannot guarantee the existence of a Nash equilibrium for the discrete time game if u_h does not satisfy the discrete dynamic programming principle (3.33). Nevertheless, we will see in the illustrative example, in the following section, how the performances of the trajectories generated by Algorithm 3.1 (arbitrary case) are better than those obtained by Algorithm 3.2 (worst case).

3.5 Application to an aircraft landing problem

3.5.1 Introduction

Aircraft accidents can occur because of quick changes of the wind velocity at low altitudes which present a real danger. For this reason, it is important to look for the best flying configurations to avoid a failed landing. It consists in steering the aircraft to the maximum altitude that can be reached, during an interval of time, in order to prevent a crash on the ground.

In papers [104, 103], a Chebyshev-type optimal control was proposed and an approximated solution of the problem is computed in order to deduce an approximated feedback control. In paper [6], the Hamilton-Jacobi-Bellman approach was used to characterize and compute the value function of the control problem when the wind behavior is supposed to be known.

A more realistic situation can be found in [29] where a nonlinear differential game with integral payoff functional and state constraints was studied. In particular, the dynamic programming approach was applied to the problem of an aircraft control during take-off in a windshear. In this case, the first player (the minimizer, the pilot) uses feedback strategies, while the second player (the maximizer, the wind) uses nonanticipative strategies (the wind is permitted to measure the current value of the first player's control). To solve this problem, a semi-Lagrangian scheme is applied to compute an approximation of the value function.

Consider the flight of an aircraft in a vertical plane. Different forces are acting on the center of gravity of the aircraft. Among those forces one can cite:

- The thrust force $\mathbf{F_T}$ with a modulus of the form $F_T(u)$ where u is the modulus of the aircraft velocity.
- The lift and drag forces $\mathbf{F_L}$ and $\mathbf{F_D}$ with modulus F_L and F_D depending on u and the angle of attack θ.
- The weight force $\mathbf{F_P}$ with modulus $F_P = mg$, where m is the aircraft mass and g is the gravitational constant.

From the Newton's law, we deduce the following equations of motion (see [6, 33]):

$$\begin{cases} \dot{x}(s) &= u(s)\cos(\gamma(s)) + \omega_x(s) \\ \dot{h}(s) &= u(s)\sin(\gamma(s)) + \omega_h(s) \\ \dot{u}(s) &= \frac{F_T(u(s))}{m}\cos(\theta(s)+\delta) - \frac{F_D(u(s),\theta(s))}{m} - g\sin(\gamma(s)) - \dot{\omega}_x(s)\cos(\gamma(s)) - \dot{\omega}_h(s)\sin(\gamma(s)) \\ \dot{\gamma}(s) &= \frac{F_T(u(s))}{mu(s)}\sin(\theta(s)+\delta) + \frac{F_L(u(s),\theta(s))}{mu(s)} - g\frac{\cos(\gamma(s))}{u(s)} + \dot{\omega}_x(s)\frac{\sin(\gamma(s))}{u(s)} - \dot{\omega}_h(s)\frac{\cos(\gamma(s))}{u(s)} \\ \dot{\theta}(s) &= a(s), \end{cases}$$

where x is the horizontal distance, h denotes the aircraft altitude, u is the velocity, γ is the relative path inclination, θ is the angle of attack, $\delta > 0$ is a parameter of the model, ω_x and ω_h are respectively the horizontal and the vertical components of the wind velocity vector, $\dot{\omega}_x$ and $\dot{\omega}_h$ are their derivatives and a represents the control variable.

Precise expressions of F_T, F_L and F_D and numerical values of different parameters of the model can be found in [6, 33, 34] and in Appendix 4.7.2 of chapter 4.

3.5.2 5D differential game model

In this paper, we propose a differential game model with maximum running cost in which wind disturbances are considered as a second player and our aim is to steer the aircraft to the maximum altitude that can be reached during an interval of time, by means of nonanticipative strategies, in order to prevent a crash on the ground.

Model presentation and differential game

The wind disturbances are represented by $\dot{\omega}_x$ and $\dot{\omega}_h$ which are the derivatives of the horizontal and the vertical components of the wind velocity vector. To simplify the notations, denote $b(\cdot) = (b_1(\cdot), b_2(\cdot)) := (\dot{\omega}_x(\cdot), \dot{\omega}_h(\cdot))$ that takes values in a compact set $B \subset \mathbb{R}^2$ of the form:

$$B := B_1 \times B_2 = [b_{1,\min}, b_{1,\max}] \times [b_{2,\min}, b_{2,\max}].$$

Moreover, we consider new state variables represented by $y(\cdot) : [0,T] \to \mathbb{R}^5$ with $T > 0$ and

$$y(\cdot) := (h(\cdot), u(\cdot), \gamma(\cdot), \omega_h(\cdot), \theta(\cdot))^\top.$$

Henceforth, the above 5-D differential system becomes:

$$\begin{cases} \dot{h}(s) = u(s)\sin(\gamma(s)) + \omega_h(s) \\ \dot{u}(s) = \frac{F_T(u(s))}{m}\cos(\theta(s)+\delta) - \frac{F_D(u(s),\theta(s))}{m} - g\sin(\gamma(s)) - b_1(s)\cos(\gamma(s)) - b_2(s)\sin(\gamma(s)) \\ \dot{\gamma}(s) = \frac{F_T(u(s))}{mu(s)}\sin(\theta(s)+\delta) + \frac{F_L(u(s),\theta(s))}{mu(s)} - \frac{g}{u(s)}\cos(\gamma(s)) + \frac{b_1(s)}{u(s)}\sin(\gamma(s)) - \frac{b_2(s)}{u(s)}\cos(\gamma(s)) \\ \dot{\omega}_h(s) = b_2(s) \\ \dot{\theta}(s) = a(s), \end{cases}$$

where $a(\cdot)$ is the control of the first player that takes values in a compact and convex set given by $A := [a_{\min}, a_{\max}]$ with $a_{\max} = -a_{\min}$. This differential system can be expressed differently:

$$\dot{y}(t) = f(t, y(t), a(t), b(t)) := g_0(y(t)) + b_1(t)g_1(y(t)) + b_2(t)g_2(y(t)) + a(t)e_5$$

where $e_5 = (0,0,0,0,1)^T$ and for $y = (h, u, \gamma, \omega_h, \theta) \in \mathbb{R}^5$

$$g_0(y) = \begin{pmatrix} u\sin(\gamma) \\ \frac{F_T(u)}{m}\cos(\theta+\delta) - \frac{F_D(u,\theta)}{m} - g\sin(\gamma) \\ \frac{1}{u}\left(\frac{F_T(u)}{m}\sin(\theta+\delta) + \frac{F_L(u,\theta)}{m} - g\cos(\gamma)\right) \\ 0 \\ 0 \end{pmatrix}, g_1(y) = \begin{pmatrix} 0 \\ -\cos(\gamma) \\ \frac{1}{u}\sin(\gamma) \\ 0 \\ 0 \end{pmatrix} \text{ and } g_2(y) = \begin{pmatrix} 0 \\ -\sin(\gamma) \\ \frac{1}{u}\cos(\gamma) \\ 1 \\ 0 \end{pmatrix}.$$

In order to transform our problem into a minimization problem, the maximum running cost function ϕ is defined as $\phi(y) := H_r - h$, where h is the aircraft altitude (the first component of the state vector y) and $H_r > 0$ is a given reference altitude. Finally, the admissible set $\mathcal{K}$ has the following form:

$$\mathcal{K} = [h_{\min}, h_{\max}] \times [u_{\min}, u_{\max}] \times [\gamma_{\min}, \gamma_{\max}] \times [\omega_{h,\min}, \omega_{h,\max}] \times [\theta_{\min}, \theta_{\max}].$$

Numerical resolution

In order to determinate the intervals in which the state variables, the control of the first player and the wind disturbances take values, we exploit the wind model presented in [6, Appendix A]. Therefore, we obtain the following constraints on the system state and the players controls, presented respectively in tables 3.1 and 3.2:

State variable	h(ft)	u(ft s^{-1})	γ (deg)	ω_h(ft s^{-1})	θ (deg)
min	450	160	-7.0	-100.0	0.0
max	1000	260	15.0	50.0	17.2

Table 3.1: State constraints: domain $\mathcal{K}$.

Control variables	a (deg s^{-1})	b_1(ft s^{-2})	b_2(ft s^{-2})
min	-3.0	0.0	-2.0
max	3.0	7.7	2.0

Table 3.2: Control constraints: sets A and B.

Remark 3.5.1. *Since ϕ is bounded, the auxiliary variable z will take values in an interval of the form $[z_{\min}, z_{\max}]$. Here, we take $z_{\min} = H_r - h_{\max}$ and $z_{\max} = H_r - h_{\min}$.*

To solve the auxiliary optimal control problem, we extend the computational domain in all directions (see Proposition 3.3.6 of section 3.3) to obtain $\mathcal{K}_\mu := \mathcal{K} + \mu B_\infty$ where $B_\infty := [-1,1]^5$ and the parameter μ is a small fixed positive value (here, we take $\mu = 0.05$). Then, consider the following mesh steps $\Delta := (\delta t, (\delta y_i)_{1\le i\le 5}, \delta z)$. For a given multi-index $i = (i_1, ..., i_5) \in \mathbb{N}^5$, let $y_i := y_{\min,i} + i\delta y_i$, $z_j := z_{\min} + j\delta z$, $j \in \mathbb{N}$ and $t_n := n\delta t$, $n = 0, ..., N$, where N is the integer part of $T/\delta t$. Therefore, we define the following grid on $\mathcal{K}_\mu \times [z_{min}, z_{max}]$ by:

$$\mathcal{G} := \Big\{(y_i, z_j) \in \mathcal{K}_\mu \times [z_{min}, z_{max}], \quad i \in \mathbb{N}^5, j \in \mathbb{N}\Big\}.$$

On the other hand, the Hamiltonian function $H(y,p)$, for $(y,p) \in \mathbb{R}^5 \times \mathbb{R}^5$, can be explicitly calculated:

$$\begin{aligned} H(y,p) &= \min_{b\in B}\max_{a\in A} -\langle f(t,y,a,b), p\rangle \\ &= -\langle g_0(y), p\rangle - \max_{b_1\in B_1} b_1\langle g_1(y), p\rangle - \max_{b_2\in B_2} b_2\langle g_2(y), p\rangle + a_{\max}|p_5|. \end{aligned}$$

The last equality is justified by the fact that $a_{\max} = -a_{\min}$. Therefore, the corresponding numerical Hamiltonian can be given by:

$$\mathcal{H}(y, p^-, p^+) = \sum_{i=1}^{i=4} \max\Big(- f_i(y, b_{opt}), 0\Big) p_i^- + \min\Big(- f_i(y, b_{opt}), 0\Big) p_i^+ + a_{\max} \max\Big(p_5^-, -p_5^+, 0\Big),$$

where $b_{opt} \in \underset{b \in B}{\text{argmax}} \langle (b_1 g_1(y) + b_2 g_2(y)), p \rangle$ with $p = \frac{p^- + p^+}{2}$.

Finally, to approximate the auxiliary value function w, we solve numerically a discretized form of the corresponding HJ equation by using the following explicit scheme (see [6, 25]):

$$\begin{cases} w_{i,j}^N = \hat{\phi}_{i,j} \\ w_{i,j}^n = \max\left(w_{i,j}^{n+1} - \delta t \mathcal{H}(y_i, D^- w_{i,j}^{n+1}, D^+ w_{i,j}^{n+1}), \hat{\phi}_{i,j}\right), \quad n \in \{N-1, ..., 0\}, \ \ (y_i, z_j) \in \mathcal{G}, \end{cases} \tag{3.38}$$

where $\hat{\phi}_{i,j} := \hat{\phi}(y_i, z_j)$, $w_{i,j}^n$ is an approximation of $w(t_n, y_i, z_j)$ and the terms $D^{\pm} w_{i,j}^n$ are approximated through a second order ENO scheme, see [108], given by the following expression:

$$D_k^{\pm} w_{i,j}^n := \pm \frac{w_{i \pm e_k, j}^n - w_{i,j}^n}{\delta y_k} \mp \frac{\delta y_k}{2} \sigma(D_{k,0}^2 w_{i,j}^n, D_{k,\pm 1}^2 w_{i,j}^n)$$

with $D_{k,\epsilon}^2 w_{i,j}^n := (w_{i+(\epsilon-1)e_k, j}^n + 2 w_{i+2\epsilon e_k, j}^n - w_{i+(\epsilon+1)e_k, j}^n)/(\delta y_k)^2$, for $k \in \{1, ..., 5\}$, e_k denotes the $k-$element of the canonical basis of $\mathbb{R}^5$ and σ is defined, for any $a, b \in \mathbb{R}$, as follows:

$$\sigma(a, b) := \begin{cases} a \ \text{ if } \ ab > 0 \ \text{ and } \ |a| \leq |b|, \\ b \ \text{ if } \ ab > 0 \ \text{ and } \ |a| > |b|, \\ 0 \ \text{ if } \ ab \leq 0. \end{cases}$$

Remark 3.5.2. *The numerical Hamiltoanian $\mathcal{H}$ is consistent with H, i.e.*

$$\mathcal{H}(y, p, p) = H(y, p), \quad \textit{for any } \ y, p \in \mathbb{R}^5,$$

monotone, i.e. for any $k = 1, ..., 5$, $\frac{\partial \mathcal{H}}{\partial p_k^-}(y, p^-, p^+) \geq 0$ and $\frac{\partial \mathcal{H}}{\partial p_k^+}(y, p^-, p^+) \leq 0$ and Lipschitz continuous w.r.t. all its arguments. Furthermore, the time step δt is chosen in order to satisfy the Courant-Friedrich-Levy condition:

$$\delta t \Big(\sum_{i=1}^{4} \frac{|f_i(y)|}{\delta y_i} + \frac{a_{\max}}{\delta y_5} \Big) \leq CFL,$$

where $CFL \leq 1$ is a real number in $[0, 1]$ (in our model, we take $CFL = 0.5$). Under those conditions, the approximation $w_{i,j}^n$ converges to w, the viscosity solution of (3.12) (see [25]).

As a conclusion, to solve the constrained problem (3.6), we proceed as follows:

1. First, we compute an approximation of the auxiliary value function w by solving (3.38). Denote this approximation by W^{Δ}.
2. Then, thanks to Theorem 3.3.5, we get an approximation of the value function v at an initial state y_0, denoted by $z^{\Delta}(y_0)$ and defined by:

$$z^{\Delta}(y_0) := \min\{z \in [z_{min}, z_{max}] \ \mid \ W^{\Delta}(y_0, z) \leq 0\}.$$

3. Finally, we apply our reconstruction procedure (Algorithm 3.1 or 3.2) to the approximated auxiliary value function W^{Δ}, starting from $(y_0, z^{\Delta}(y_0))$, to get approximated optimal strategies of the first player for the constrained problem (3.6), starting at time $t = 0$ from the initial state y_0 (see assertion (iii) of Theorem 3.3.5).

Numerical test: Reconstruction of optimal trajectories and controls

The final time horizon is set to $T = 8$ and consider a grid $\mathcal{G}$ containing $40 \times 20^4 \times 5$ points. In this test, we show results of trajectory reconstruction for some initial positions obtained with algorithms 3.1 and 3.2. As initial points, we take:

$$y_1 = (600, 239.7, -2.249, -26.0, 7.373), \quad y_2 = (650, 230, -2.249, -28.0, 7.373)$$

and

$$y_3 = (800, 200.0, -2.249, -30.0, 7.373).$$

First, we shall mention that the reconstruction of admissible trajectories is not theoretically possible for any initial position. Indeed, there are some initial points from which we cannot find any strategy $\alpha[\cdot] \in \Gamma$ of the controller (the first player) that guarantees the admissibility of trajectories, for any perturbation of the wind. Those initial points correspond to a positive value of the auxiliary value function w for any value of the auxiliary variable z and hence to an infinite value of the constrained problem (3.6) (see remark 3.3.3).

Furthermore, one can get an idea about all the initial positions from which there exists at least one strategy of the first player that corresponds to a trajectory satisfying the state constraints for any perturbation of the wind. The set of such initial positions is called the feasible set which can be obtained by the negative level set of the auxiliary value function w.

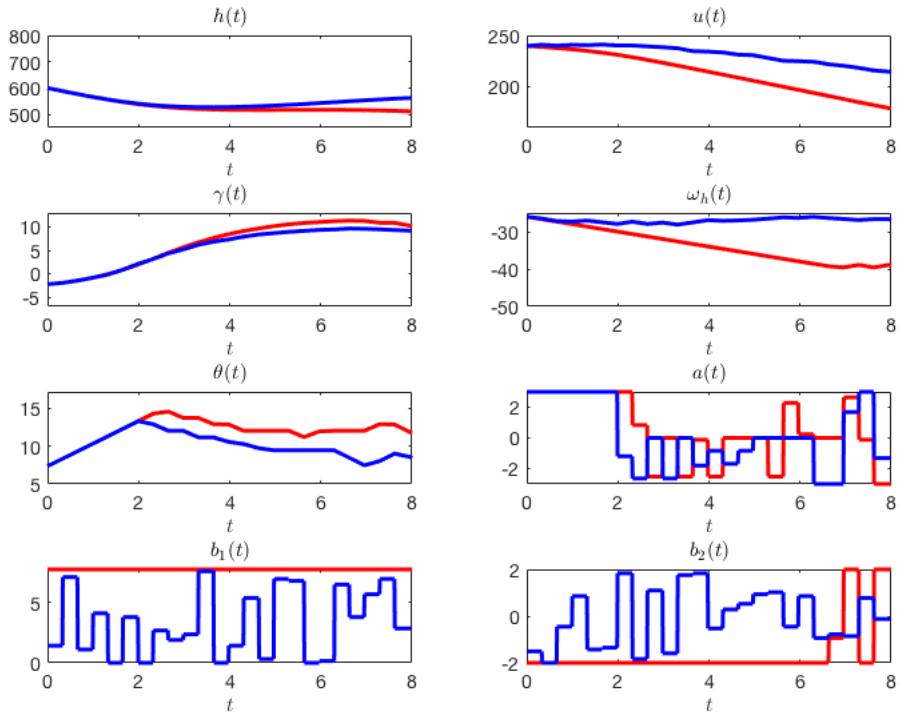

Figure 3.1: Trajectories and controls reconstruction as a response to an arbitrary case (random distribution, Algorithm 3.1, in blue) and to the worst case (Algorithm 3.2, in red) starting from y_1.

In figures 3.1 and 3.2, starting from y_1 and y_2 respectively, as expected, we observe that the altitudes reached in the worst case (Algorithm 3.2) are always below those obtained in the random case (Algorithm 3.1).

In the worst case, perturbations aim to decrease as much as possible the altitude h. Therefore, its objective is to have $\dot{h} \leq 0$. Since $\dot{h}(s) = u(s)\sin(\gamma(s)) + \omega_h(s)$, wind perturbations in this case will take values that decrease ω_h and $u\sin(\gamma)$. For this reason, we observe that the first component of the wind disturbances takes the maximal possible value, $b_1(\cdot) \equiv b_{1,max} > 0$, in order to decrease u along time. However, $b_2(\cdot)$ is equal to $b_{2,min} < 0$ $a.e.$ in order to decrease ω_h since $\dot{\omega}_h(s) = b_2(s)$, for $s \in [0, T]$.

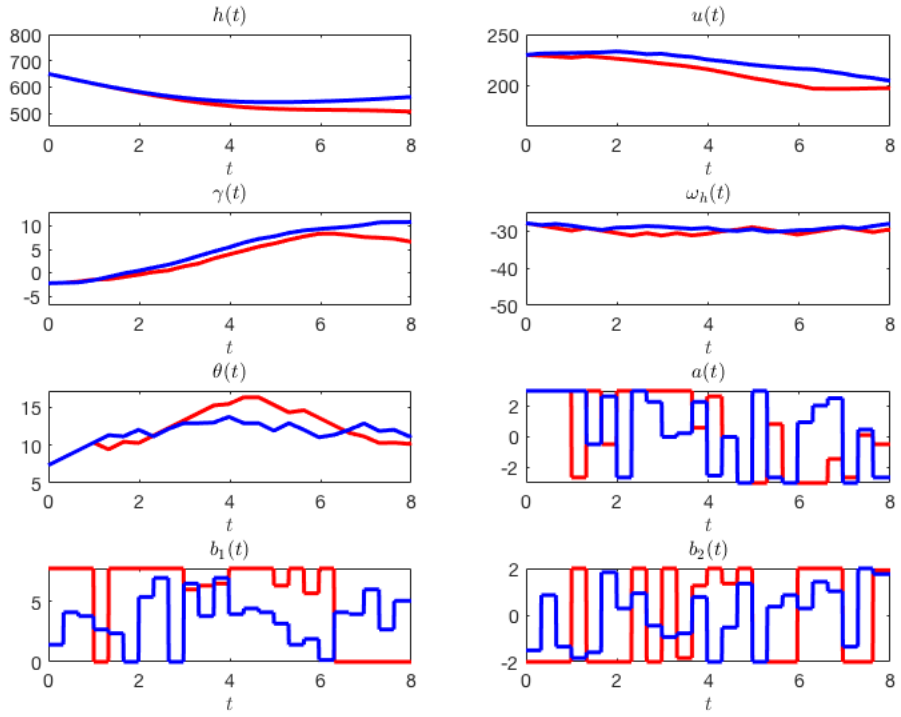

Figure 3.2: Trajectories and controls reconstruction as a response to an arbitrary case (random distribution, Algorithm 3.1, in blue) and to the worst case (Algorithm 3.2, in red) starting from y_2.

In figure 3.3, starting from y_3, we remark that the perturbations values in the worst case do not change compared to the previous initial flight configurations (y_1 and y_2). Nevertheless, we observe that the altitudes reached in the random case (Algorithm 3.1) are similar to those obtained in the worst case and there are even sensibly lower at some time instants. This observation can be justified by the high initial altitude of y_3 compared to y_1 and y_2. In other words, when the initial altitude of the aircraft is higher enough, worst perturbations cannot have the same effect as in the previous situations (initial flight configuration with lower altitude such as the case of y_1 and y_2).

3.6 Appendix: Properties of the auxiliary value function w

Proof of Proposition 3.3.1. (i) We start by proving the dynamic programming principle (3.11). To simplify notations, let's denote $\hat{x} := (x, z) \in \mathbb{R}^d \times \mathbb{R}$ and let u be defined by:

$$u(t, \hat{x}) := \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ w(\tau, \hat{y}_{t,\hat{x}}^{\alpha[b],b}(\tau)) \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha[b],b}(s)) \Big\},$$

with $\tau := t + h \leq T$ for some $h \geq 0$.

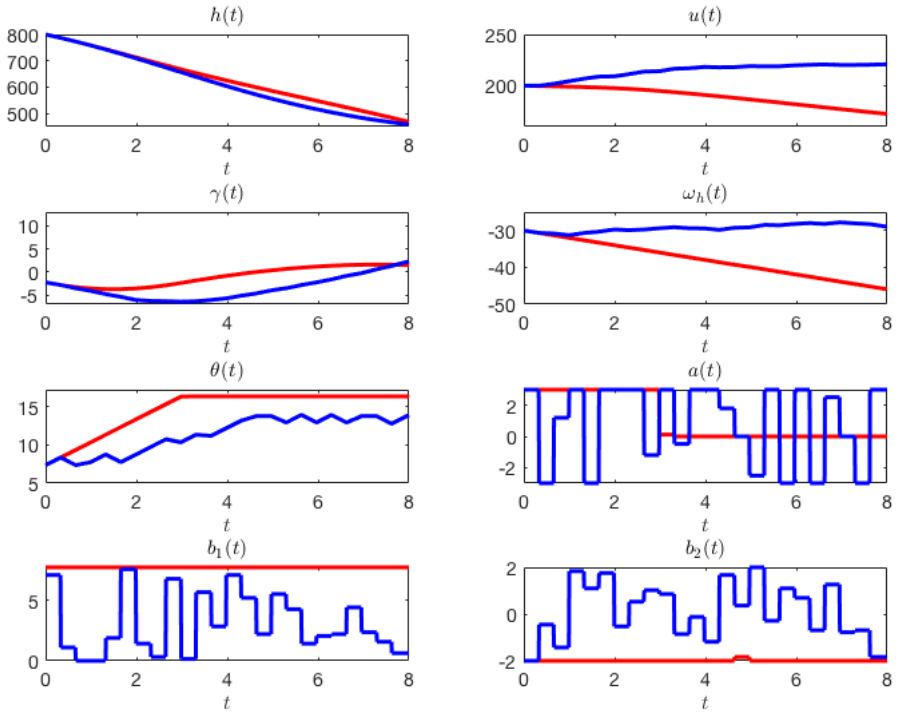

Figure 3.3: Trajectories and controls reconstruction as a response to an arbitrary case (random distribution, Algorithm 3.1, in blue) and to the worst case (Algorithm 3.2, in red) starting from y_3.

We will start by proving that $w(t,\hat{x}) \leq u(t,\hat{x})$. Let $\epsilon > 0$ and let $\alpha_\epsilon[\cdot] \in \Gamma$ be an ϵ-optimal strategy for u, i.e.

$$u(t,\hat{x}) + \epsilon \geq \sup_{b(\cdot)\in\mathcal{B}} \left\{ w(\tau, \hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b],b}(\tau)) \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b],b}(s)) \right\}. \tag{3.39}$$

For a given strategy $\alpha[\cdot] \in \Gamma$, we denote by $\alpha^1[\cdot]$ and $\alpha^2[\cdot]$ its restrictions to $[t,\tau]$ and $[\tau,T]$ respectively. In a similar way, for any $b(\cdot) \in \mathcal{B}$, we will denote by $b^1(\cdot)$ and $b^2(\cdot)$ its restrictions to $[t,\tau]$ and $[\tau,T]$ respectively. Furthermore, for a given $b(\cdot) \in \mathcal{B}$, let $\hat{y}^1(\cdot)$ denote the restriction of $\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b],b}(\cdot)$ to $[t,\tau]$. Finally, we define $\sigma^\tau(t,\hat{x};b^1)$ by

$$\sigma^\tau(t,\hat{x};b^1) := w(\tau,\hat{y}^1(\tau)) \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}^1(s)).$$

We have the following equality:

$$\sigma^\tau(t,\hat{x};b^1) = \inf_{\alpha^2[\cdot]} \sup_{b^2(\cdot)} \left\{ \left(\max_{s\in[\tau,T]} \hat{\phi}(\hat{y}_{\tau,\hat{y}^1(\tau)}^{\alpha^2[b^2],b^2}(s)) \right) \bigvee \hat{\psi}(\hat{y}_{\tau,\hat{y}^1(\tau)}^{\alpha^2[b^2],b^2}(T)) \right\} \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}^1(s)).$$

Since $\left(\max_{s\in[t,\tau]} \hat{\phi}(\hat{y}^1(s)) \right)$ does not depend on $\alpha^2[\cdot]$ in the above equality, we deduce that:

$$\sigma^\tau(t,\hat{x};b^1) = \inf_{\alpha^2[\cdot]} \left\{ \sup_{b^2(\cdot)} \left(\max_{s\in[\tau,T]} \hat{\phi}(\hat{y}_{\tau,\hat{y}^1(\tau)}^{\alpha^2[b^2],b^2}(s)) \bigvee \hat{\psi}(\hat{y}_{\tau,\hat{y}^1(\tau)}^{\alpha^2[b^2],b^2}(T)) \right) \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}^1(s)) \right\}.$$

Now, let $\delta_\epsilon[\cdot]$ be an ϵ-optimal strategy for $\sigma^\tau(t,\hat{x};b^1)$, which means that:

$$\sup_{b^2(\cdot)} \left\{ \max_{s\in[\tau,T]} \hat{\phi}(\hat{y}_{\tau,\hat{y}^1(\tau)}^{\delta_\epsilon[b^2],b^2}(s)) \bigvee \hat{\psi}(\hat{y}_{\tau,\hat{y}^1(\tau)}^{\delta_\epsilon[b^2],b^2}(T)) \right\} \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}^1(s)) \leq \epsilon + \sigma^\tau(t,\hat{x};b^1). \tag{3.40}$$

We define the strategy $\alpha_0[\cdot] \in \Gamma$ as follows

$$\alpha_0[\cdot] = \begin{cases} \alpha_\epsilon[\cdot] \text{ on } [t,\tau], \\ \delta_\epsilon[\cdot] \text{ on } [\tau,T]. \end{cases}$$

From the definition of w, we have:

$$w(t,\hat{x}) \leq \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \Big(\max_{s\in[t,T]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_0[b],b}(s)) \Big) \bigvee \hat{\psi}(\hat{y}_{t,\hat{x}}^{\alpha_0[b],b}(T)) \Big\}.$$

On the other hand, from the definition of $\alpha_0[\cdot]$, we have for any $b(\cdot) \in \mathcal{B}$

$$\max_{s\in[t,T]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_0[b],b}(s)) = \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}^1(s)) \bigvee \max_{s\in[\tau,T]} \hat{\phi}(\hat{y}_{\tau,\hat{y}^1(\tau)}^{\delta_\epsilon[b^2],b^2}(s))$$

and

$$\hat{\psi}(\hat{y}_{t,\hat{x}}^{\alpha_0[b],b}(T)) = \hat{\psi}(\hat{y}_{\tau,\hat{y}^1(\tau)}^{\delta_\epsilon[b^2],b^2}(T)),$$

where $\hat{y}^1(\cdot)$ is the restriction of $\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b],b}(\cdot)$ to $[t,\tau]$. Therefore, the above inequality becomes:

$$w(t,\hat{x}) \leq \sup_{b^1(\cdot)} \left\{ \sup_{b^2(\cdot)} \Big(\max_{s\in[\tau,T]} \hat{\phi}(\hat{y}_{\tau,\hat{y}^1(\tau)}^{\delta_\epsilon[b^2],b^2}(s)) \bigvee \hat{\psi}(\hat{y}_{\tau,\hat{y}^1(\tau)}^{\delta_\epsilon[b^2],b^2}(T)) \Big) \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}^1(s)) \right\}.$$

From (3.40), we deduce that:

$$w(t,\hat{x}) \leq \sup_{b^1(\cdot)} \Big\{ \epsilon + \sigma^\tau(t,\hat{x};b^1) \Big\} = \epsilon + \sup_{b^1(\cdot)} \Big\{ w(\tau,\hat{y}^1(\tau)) \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}^1(s)) \Big\}.$$

Furthermore, we have:

$$\begin{aligned} \sup_{b^1(\cdot)} \Big\{ w(\tau,\hat{y}^1(\tau)) \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}^1(s)) \Big\} &= \sup_{b(\cdot)\in\mathcal{B}} \Big\{ w(\tau,\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b],b}(\tau)) \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}_{t,x}^{\alpha^1[b],b}(s)) \Big\} \\ &\leq \epsilon + u(t,\hat{x}), \end{aligned}$$

where the last line holds by using (3.39). As a conclusion, by combining the two last inequalities we obtain $w(t,\hat{x}) \leq u(t,\hat{x}) + 2\epsilon$ for any $\epsilon > 0$, which gives the desired inequality.

Now, we will show that $w(t,\hat{x}) \geq u(t,\hat{x})$. For any $\epsilon > 0$, there exists an ϵ-optimal strategy for $w(t,\hat{x})$, denoted by $\alpha_\epsilon[\cdot]$, such that:

$$w(t,\hat{x}) + \epsilon \geq \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \max_{s\in[t,T]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b],b}(s)) \bigvee \hat{\psi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b],b}(T)) \Big\}. \tag{3.41}$$

Let $\mathcal{A}(\tau)$ and $\mathcal{B}(\tau)$ be the set of restrictions of the first and the second players controls respectively to the time interval $[\tau,T]$:

$$\mathcal{A}(\tau) := \{a^2(\cdot) : [\tau,T] \to A, \text{ measurable}\} \quad \text{and} \quad \mathcal{B}(\tau) := \{b^2(\cdot) : [\tau,T] \to B, \text{ measurable}\}.$$

Let's fix a control of the second player $\overline{b}(\cdot) \in \mathcal{B}$ and let $\alpha^2[\cdot]$ be a nonanticpative strategy of the first player defined, for any $b^2(\cdot) \in \mathcal{B}(\tau)$, by:

$$\alpha^2[b^2](s) := \alpha_\epsilon[\hat{b}](s) \quad \text{for} \quad s \in [\tau,T],$$

where $\hat{b}(\cdot) \in \mathcal{B}$ is given by:

$$\hat{b}(s) = \begin{cases} \overline{b}(s) & \text{for} \quad s \in [t,\tau] \\ b^2(s) & \text{for} \quad s \in [\tau,T]. \end{cases}$$

We mention here that $\alpha^2[\cdot] \notin \Gamma$ since $\alpha^2[\cdot]$ is a mapping from $\mathcal{B}(\tau)$ to $\mathcal{A}(\tau)$. Finally, we define $\mathcal{B}(t,\tau,\bar{b}) \subset \mathcal{B}$, by:

$$\mathcal{B}(t,\tau,\bar{b}) := \Big\{ b(\cdot) \in \mathcal{B} \;\mid\; b(s) = \bar{b}(s) \quad \text{a.e.} \quad s \in [t,\tau] \Big\}.$$

For a second player control $b(\cdot) \in \mathcal{B}$, we set $\hat{y}^1 := \hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b],b}(\tau) \in \mathbb{R}^{d+1}$. From the definition of $w(\tau,\hat{y}^1)$, we have:

$$w(\tau,\hat{y}^1) \leq \sup_{b^2(\cdot)\in\mathcal{B}(\tau)} \Big\{ \max_{s\in[\tau,T]} \hat{\phi}(\hat{y}_{\tau,\hat{y}^1}^{\alpha^2[b^2],b^2}(s)) \bigvee \hat{\psi}(\hat{y}_{\tau,\hat{y}^1}^{\alpha^2[b^2],b^2}(T)) \Big\}. \tag{3.42}$$

Now, let $b_0(\cdot) \in \mathcal{B}(t,\tau,\bar{b})$ and denote by $b_0^2(\cdot) \in \mathcal{B}(\tau)$ its restriction to $[\tau,T]$. Following the definition of $\alpha^2[\cdot]$ above, we obtain $\alpha^2[b_0^2](\cdot) = \alpha_\epsilon[\hat{b}_0](\cdot)$ on $[\tau,T]$. On the other hand, $\hat{b}_0(\cdot) = b_0(\cdot)$ because $b_0(\cdot) \in \mathcal{B}(t,\tau,\bar{b})$. We deduce that $\alpha^2[b_0^2](\cdot) = \alpha_\epsilon[b_0](\cdot)$ on $[\tau,T]$ and which implies that:

$$\max_{s\in[t,T]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b_0],b_0}(s)) = \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[\bar{b}],\bar{b}}(s)) \bigvee \max_{s\in[\tau,T]} \hat{\phi}(\hat{y}_{\tau,\hat{y}^1}^{\alpha^2[b_0^2],b_0^2}(s))$$

and

$$\hat{\psi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b_0],b_0}(T)) = \hat{\psi}(\hat{y}_{\tau,\hat{y}^1}^{\alpha^2[b_0^2],b_0^2}(T)),$$

where $\hat{y}^1 = \hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b_0],b_0}(\tau) = \hat{y}_{t,\hat{x}}^{\alpha_\epsilon[\bar{b}],\bar{b}}(\tau)$ since $b_0(\cdot) = \bar{b}(\cdot)$ on $[t,\tau]$. From the two above equalities, we deduce:

$$\begin{aligned}
&\sup_{b_0(\cdot)\in\mathcal{B}(t,\tau,\bar{b})} \Big\{ \max_{s\in[t,T]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b_0],b_0}(s)) \bigvee \hat{\psi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b_0],b_0}(T)) \Big\} \\
&\qquad = \sup_{b_0^2(\cdot)\in\mathcal{B}(\tau)} \Big\{ \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[\bar{b}],\bar{b}}(s)) \bigvee \max_{s\in[\tau,T]} \hat{\phi}(\hat{y}_{\tau,\hat{y}^1}^{\alpha^2[b_0^2],b_0^2}(s)) \bigvee \hat{\psi}(\hat{y}_{\tau,\hat{y}^1}^{\alpha^2[b_0^2],b_0^2}(T)) \Big\}.
\end{aligned} \tag{3.43}$$

Now since $\mathcal{B}(t,\tau,\bar{b}) \subset \mathcal{B}$, we deduce from (3.41) that:

$$\sup_{b_0(\cdot)\in\mathcal{B}(t,\tau,\bar{b})} \Big\{ \max_{s\in[t,T]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b_0],b_0}(s)) \bigvee \hat{\psi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[b_0],b_0}(T)) \Big\} \leq w(t,\hat{x}) + \epsilon,$$

and from (3.43), the last inequality becomes, for any $\bar{b}(\cdot) \in \mathcal{B}$:

$$\sup_{b_0^2(\cdot)\in\mathcal{B}(\tau)} \Big\{ \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[\bar{b}],\bar{b}}(s)) \bigvee \max_{s\in[\tau,T]} \hat{\phi}(\hat{y}_{\tau,\hat{y}^1}^{\alpha^2[b_0^2],b_0^2}(s)) \bigvee \hat{\psi}(\hat{y}_{\tau,\hat{y}^1}^{\alpha^2[b_0^2],b_0^2}(T)) \Big\} \leq w(t,\hat{x}) + \epsilon.$$

The inequality (3.42) allows us to conclude that for any $\bar{b}(\cdot) \in \mathcal{B}$, we have:

$$w(\tau,\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[\bar{b}],\bar{b}}(\tau)) \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[\bar{b}],\bar{b}}(s)) \leq w(t,\hat{x}) + \epsilon.$$

Taking the supremum over $\bar{b}(\cdot) \in \mathcal{B}$ in the last inequality gives:

$$u(t,\hat{x}) \leq \sup_{\bar{b}(\cdot)\in\mathcal{B}} \Big\{ w(\tau,\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[\bar{b}],\bar{b}}(\tau)) \bigvee \max_{s\in[t,\tau]} \hat{\phi}(\hat{y}_{t,\hat{x}}^{\alpha_\epsilon[\bar{b}],\bar{b}}(s)) \Big\} \leq w(t,\hat{x}) + \epsilon,$$

which gives the desired inequality since ϵ is chosen arbitrarily.

(ii) Now, we give the proof of the local Lipschitz continuity of w. We start by proving that w is Lipschitz continuous with respect to the (x,z) variables. To simplify the notations, denote by $\hat{x} := (x,z)$, $\hat{x}' := (x',z') \in \mathbb{R}^d \times \mathbb{R}$ and let w_0 be defined by $w_0(\hat{x})$:=$w(T,\hat{x}) = \hat{\phi}(\hat{x}) \bigvee \hat{\psi}(\hat{x})$. We notice that w_0 is locally

Lipschitz continuous with a Lipschitz constant $L_0(R)$, for any $R > 0$, since $\hat{\phi}$ and $\hat{\psi}$ are locally Lipschitz continuous. For $t \in [0, T]$, we have:

$$w(t,\hat{x}) - w(t,\hat{x}') \leq \sup_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \Big(\max_{s\in[t,T]} \hat{\phi}(\hat{y}^{\alpha[b],b}_{t,\hat{x}}(s)) \bigvee \hat{\psi}(\hat{y}^{\alpha[b],b}_{t,\hat{x}}(T)) \Big) - \Big(\max_{s\in[t,T]} \hat{\phi}(\hat{y}^{\alpha[b],b}_{t,\hat{x}'}(s)) \bigvee \hat{\psi}(\hat{y}^{\alpha[b],b}_{t,\hat{x}'}(T)) \Big) \Big\},$$

since $\inf_{x\in X} f(x) - \inf_{x\in X} g(x) \leq \sup_{x\in X}(f-g)(x)$ for any arbitrary set X and any functions f and g from X to $\mathbb{R}$. From the definition of w_0, the last inequality becomes:

$$w(t,\hat{x}) - w(t,\hat{x}') \leq \sup_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \Big(w_0(\hat{y}^{\alpha[b],b}_{t,\hat{x}}(T)) - w_0(\hat{y}^{\alpha[b],b}_{t,\hat{x}'}(T)) \Big) \bigvee \max_{s\in[t,T]} \Big(\hat{\phi}(\hat{y}^{\alpha[b],b}_{t,\hat{x}}(s)) - \hat{\phi}(\hat{y}^{\alpha[b],b}_{t,\hat{x}'}(s)) \Big) \Big\},$$

since $\max(a,b) - \max(c,d) \leq \max(a-c, b-d)$, for any a, b, c, $d \in \mathbb{R}$. By the Lipschitz continuity of w_0 and $\hat{\phi}$, we deduce that:

$$w(t,\hat{x}) - w(t,\hat{x}') \leq \sup_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \Big(L_0(R) \|\hat{y}^{\alpha[b],b}_{t,\hat{x}}(T) - \hat{y}^{\alpha[b],b}_{t,\hat{x}'}(T)\| \Big) \bigvee \Big(L_{\hat{\phi}}(R) \max_{s\in[t,T]} \|\hat{y}^{\alpha[b],b}_{t,\hat{x}}(s) - \hat{y}^{\alpha[b],b}_{t,\hat{x}'}(s)\| \Big) \Big\}.$$

Since $\hat{f}$ is locally Lipschitz continuous, with Lipschitz constant $L_{\hat{f}}(R)$ for any $R > 0$, and from the Gronwall Lemma, we obtain for any $(\alpha[\cdot], b(\cdot)) \in \Gamma \times \mathcal{B}$ and any $s \in [t,T]$:

$$\|\hat{y}^{\alpha[b],b}_{t,\hat{x}}(s) - \hat{y}^{\alpha[b],b}_{t,\hat{x}'}(s)\| \leq e^{L_{\hat{f}}(R)T} \|\hat{x} - \hat{x}'\|.$$

As a conclusion, we obtain:

$$w(t,\hat{x}) - w(t,\hat{x}') \leq \Big(L_0(R) \bigvee L_{\hat{\phi}}(R) \Big) e^{L_{\hat{f}}(R)T} \|\hat{x} - \hat{x}'\|.$$

Now, since $\hat{x}$ and $\hat{x}'$ play symmetric roles, we have also:

$$w(t,\hat{x}') - w(t,\hat{x}) \leq \Big(L_0(R) \bigvee L_{\hat{\phi}}(R) \Big) e^{L_{\hat{f}}(R)T} \|\hat{x} - \hat{x}'\|,$$

from which we conclude that:

$$|w(t,\hat{x}) - w(t,\hat{x}')| \leq \Big(L_0(R) \bigvee L_{\hat{\phi}}(R) \Big) e^{L_{\hat{f}}(R)T} \|\hat{x} - \hat{x}'\|. \tag{3.44}$$

For the sequel of the proof, denote $L_w(R) := \Big(L_0(R) \bigvee L_{\hat{\phi}}(R) \Big)$, for any $R > 0$.

Now, let $\hat{x} = (x,z) \in \mathbb{R}^d \times \mathbb{R}$, $t, t_1 \in [0,T]$ and without loss of generality assume that $t_1 > t$. Notice that $w(t_1,\hat{x}) \geq \hat{\phi}(\hat{x})$, which implies that $w(t_1,\hat{x}) = w(t_1,\hat{x}) \bigvee \hat{\phi}(\hat{x})$. By using the dynamic programming principle (3.11) between t and t_1, we get:

$$\begin{aligned}
|w(t,\hat{x}) - w(t_1,\hat{x})| &= |\inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ \Big(w(t_1, \hat{y}^{\alpha[b],b}_{t,\hat{x}}(t_1) \bigvee \max_{s\in[t,t_1]} \hat{\phi}(\hat{y}^{\alpha[b],b}_{t,\hat{x}}(s)) \Big) - w(t_1,\hat{x}) \bigvee \hat{\phi}(\hat{x}) \Big\}| \\
&\leq \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ |w(t_1, \hat{y}^{\alpha[b],b}_{t,\hat{x}}(t_1)) - w(t_1,\hat{x})| \bigvee |\max_{s\in[t,t_1]} \hat{\phi}(\hat{y}^{\alpha[b],b}_{t,\hat{x}}(s)) - \hat{\phi}(\hat{x})| \Big\} \\
&\leq \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ |w(t_1, \hat{y}^{\alpha[b],b}_{t,\hat{x}}(t_1)) - w(t_1,\hat{x})| \bigvee \max_{s\in[t,t_1]} |\hat{\phi}(\hat{y}^{\alpha[b],b}_{t,\hat{x}}(s)) - \hat{\phi}(\hat{x})| \Big\} \\
&\leq \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \Big\{ L_w(R) \|\hat{y}^{\alpha[b],b}_{t,\hat{x}}(t_1) - \hat{x}\| \bigvee L_{\hat{\phi}}(R) \max_{s\in[t,t_1]} \|\hat{y}^{\alpha[b],b}_{t,\hat{x}}(s) - \hat{x}\| \Big\}.
\end{aligned}$$

We denote by $C_{\hat{f}}$:= $\max_{(s,a,b)\in[0,T]\times A\times B} \|\hat{f}(s,0,a,b)\|$. $C_{\hat{f}}$ is finite since $(s,a,b)\mapsto \hat{f}(s,0,a,b)$ is continuous over the compact set $[0,T]\times A\times B$. Therefore, we have $\|\hat{f}(s,\hat{x},a,b)\| \leq C_{\hat{f}} + L_{\hat{f}}(R)\|\hat{x}\|$ for any $\hat{x}\in\mathbb{R}^d\times\mathbb{R}$. Hence by a Gronwall estimate, we obtain

$$\|\hat{y}_{t,\hat{x}}^{\alpha[b],b}(s) - \hat{x}\| \leq (C_{\hat{f}} + L_{\hat{f}}(R)\|\hat{x}\|)e^{L_{\hat{f}}(R)T}|s-t|, \quad \text{for any } \ s\in[t,t_1].$$

Henceforth, there exits some constant $C>0$ such that:

$$|w(t,\hat{x}) - w(t_1,\hat{x})| \leq C(1+\|\hat{x}\|)|t_1 - t|.$$

Combining (3.44) with the last inequality implies the existence of a real constant $C'>0$ such that for any $t,t'\in[0,T]$ and any $\hat{x},\hat{x}'\in\mathbb{R}^{d+1}$:

$$|w(t,\hat{x}) - w(t',\hat{x}')| \leq C'(1+\|\hat{x}\|)(|t-t'| + \|\hat{x}-\hat{x}'\|).$$

□

Chapter 4

Optimistic Planning Algorithms For Constrained Optimal Control Problems

Publication from this chapter

O.Bokanowski, N.Gammoudi and H.Zidani, *Optimistic Planning Algorithms For State-Constrained Optimal Control Problems*, Computers and Mathematics with Applications, 2020 (submitted).

4.1 Introduction

In this chapter, we study optimistic planning methods to solve some state-constrained optimal control problems in finite horizon. While classical methods for calculating the value function are generally based on a discretization in the state space, optimistic planning algorithms have the advantage of using adaptive discretization in the control space. These approaches are therefore very suitable for control problems where the dimension of the control variable is low and allow to deal with problems where the dimension of the state space can be very high. Our algorithms also have the advantage of providing, for given computing resources, the best control strategy whose performance is as close as possible to optimality while its corresponding trajectory comply with the state constraints.

Let $T > 0$ be a fixed time horizon and let $\mathcal{K}$ and $\mathcal{C}$ be two closed subsets of $\mathbb{R}^d$, with $d \in \mathbb{N}^*$. For any $(t, x) \in [0, T] \times \mathbb{R}^d$, consider the following state-constrained optimal control problem:

$$v(t,x) := \inf_{a(\cdot)\in\mathcal{A}} \Big\{ \int_t^T \ell(y^a_{t,x}(s), a(s))ds + \Phi(y^a_{t,x}(T)) \mid y^a_{t,x}(s) \in \mathcal{K},\ \forall s \in [t,T] \text{ and } y^a_{t,x}(T) \in \mathcal{C} \Big\}, \tag{4.1}$$

where $\mathcal{A}$ is the set of controls taking values in some compact subset A of $\mathbb{R}^q$, $q \geq 1$, and $y^a_{t,x}(\cdot)$, representing the system trajectory, is the unique absolutely continuous solution of the following dynamical system:

$$\begin{cases} \dot{y}(s) = f(y(s), a(s)) \ \ a.e.\ \ s \in [t,T], \\ y(t) = x. \end{cases} \tag{4.2}$$

The dynamics f and the cost functions ℓ and Φ are supposed to be Lipschitz continuous, see section 4.2 for the precise assumptions. Recall that if the problem (4.1) is free of state constraints, $\mathcal{K} = \mathcal{C} = \mathbb{R}^d$, then the value function v is Lipschitz continuous and can be characterized as the unique viscosity solution of an appropriate HJ equation. Such characterization allows to compute an approximation of v and hence to synthesize approximated optimal controls in feedback form which gives sub-optimal solutions. In this context, several numerical methods have been proposed in the literature to approximate the solutions of HJ equations such as finite differences ([51]) and semi-Lagrangian ([61, 63]). Since the computations are

done over a grid on the state space, the complexity of this class of methods depends strongly on the state dimensionality which reduces the ability of solving optimal control problems with high state dimension (*curse of dimensionality*).

In order to reduce this curse of dimensionality, several techniques have been proposed in the literature. Among those approaches, one can mention max-plus finite element method, see [1] and references therein. Furthermore, the value function of an optimal control problem can be approximated by use of occupation measures, see for instance [95]. Another idea is to seek for an approximation of the value function by means of sparse grid schemes [27], which requires strong regularity properties on the value function. One can mention also domain decomposition techniques for partial differential equations [110]. In particular, an approximated scheme was proposed in [65] to solve Hamilton-Jacobi equations by splitting the original problem into simpler problems on two sub-domains with a linking condition and by imposing constraints on the system state. Recently, a state-tree-structure method has appeared in order to approximate the solution of a dynamic programming equation, see [3, 120, 4]. This approach eliminates the space discretization and constructs a tree, starting from a given initial state, by adding only the states that will be encountered by a discrete time dynamics and a finite number of controls. Then, the value function will be computed by the dynamic programming principle on the constructed tree.
Another important class of approaches solving the curse of dimensionality is *On-line* methods where computations are done locally for just current states that will be encountered during the control process. For instance, Rollout algorithm, at a current state, looks for the best control choice improving the cost by use of heuristic ways and uses it to move to the next time step, see [21]. Another method is model predictive control, see [94, 21], where at each time step, an optimization problem, over finite control sequences, is solved and only the first value of the resulting sequence is used to move to the next time step.

In this chapter, we investigate the optimistic planning approach that, instead of discretizing the state space, it refines the set of controls and tries, after a discretization of the time interval, to find the best control value that should be applied on each time sub-interval. This approach is very interesting especially for many applications where the control dimension q is very lower compared to the state dimension d.

Optimistic planning algorithms are based on the principles of optimistic optimisation (see [105]). This approach performs the best control search by refining, iteratively and in an *optimistic* way, the control set (the notion *optimistic* will be clarified in section 4.4). A main strength of this approach is the relation between the convergence rates to the optimal solution and the computational resources allowed, which is established using some ideas of bandit theory and reinforcement learning [106].

Several optimistic planning methods have been proposed with heuristic rules for the refinement selection and without providing convergence analysis, see for instance [131, 100, 75] for finite time horizon and [76, 101, 36] for infinite time horizon with a discount factor. In [39, 38], other variants are proposed with adaptive selection rules of the control set refinement and convergence analysis results are also provided. We refer also to [37] for the case of two-person games in infinite horizon and without state constraints.

Our contribution in this chapter consists in extending the optimistic planning approach to deal with finite horizon problems in presence of state constraints. Recall that in this case, $\mathcal{K} \neq \mathbb{R}^d$ or $\mathcal{C} \neq \mathbb{R}^d$, the value function is in general discontinuous and its characterization as solution to an HJ equation is no longer valid, unless some controllability assumptions are satisfied. First, we follow the level set approach, introduced in [5, 6], which is showed to be relevant to characterize the constrained problem by an auxiliary control problem free of state constraints. Then, we adapt two algorithms from [39, 38], OPC - "*Optimistic Planning with Continuous actions*", and SOPC - "*Simultaneous* OPC", in order to solve the auxiliary problem and hence to get an approximation of the optimal solution for the original state-constrained problem. Moreover, even in this framework, we prove convergence results similar to those established in [39]. Besides, we improve the analysis of the complexity of these algorithms and provide simplified proof arguments for this analysis. Finally, we propose an algorithm which combines both the optimistic planning approach with ideas from the MPC (Model Predictive Control). This algorithm gives better numerical results than those obtained by

the previous methods and reduce significantly the computational time. We illustrate the relevance of our algorithms on several non-linear optimal control problems (in one of these examples, the dimension of the state is 10^3).

We organize this chapter as follows. Section 4.2 formulates the state-constrained problem and presents its associated auxiliary reformulation. Section 4.3 gives some preliminary results that will be used to design the algorithms. Section 4.4 is devoted to present the different optimistic planning algorithms with an analysis of their convergence. In section 4.5, we show how our approach can be extended to constrained optimal control problems with infinite time horizon. Several numerical examples are presented in section 4.6.

4.2 Problem formulation and discrete settings

For a fixed final time $T > 0$ and a given non-empty compact subset A of $\mathbb{R}^q$, $q \geq 1$, consider the following dynamical and controlled system:

$$\begin{cases} \dot{y}(s) = f(y(s), a(s)) \ a.e. \ s \in [t, T], \\ y(t) = x, \end{cases} \tag{4.3}$$

where $x \in \mathbb{R}^d$ and the input variable $a(\cdot)$ is a control in

$$\mathcal{A} := \{a(\cdot) : [0, T] \to A, \ \text{ measurable}\},$$

and the dynamics $f : \mathbb{R}^d \times A \to \mathbb{R}^d$ is a Lipschitz continuous function satisfying:

(H4.1) *There exist $L_{f,x}, L_{f,a} \geq 0$, such that for any $y, y' \in \mathbb{R}^d$ and $a, a' \in A$:*

$$\|f(y, a) - f(y', a')\| \leq L_{f,x}\|y - y'\| + L_{f,a}\|a - a'\|.$$

Moreover, consider a distributed cost ℓ and a final cost Φ verifying:

(H4.2) *$\ell : \mathbb{R}^d \times A \to \mathbb{R}$ is a Lipschitz continuous function i.e. there exist $L_{\ell,x}, L_{\ell,a} \geq 0$ such that for any $y, y' \in \mathbb{R}^d$ and $a, a' \in A$:*

$$|\ell(y, a) - \ell(y', a')| \leq L_{\ell,x}\|y - y'\| + L_{\ell,a}\|a - a'\|.$$

(H4.3) *$\Phi : \mathbb{R}^d \to \mathbb{R}$ is a Lipschitz continuous function i.e. there exists $L_\Phi \geq 0$ such that for any $y, y' \in \mathbb{R}^d$:*

$$|\Phi(y) - \Phi(y')| \leq L_\Phi\|y - y'\|.$$

Finally, consider the following additional convexity assumption:

(H4.4) *The set $\left\{ \begin{pmatrix} f(x,a) \\ \ell(x,a) + b \end{pmatrix}, \ a \in A, \ 0 \leq b \leq 2L_{\ell,x}\|x\| \right\}$ is convex, for any $x \in \mathbb{R}^d$.*

We are interested in the following optimal control problem:

$$v(t, x) := \inf_{a(\cdot) \in \mathcal{A}} \Big\{ \int_t^T \ell(y^a_{t,x}(s), a(s))ds + \Phi(y^a_{t,x}(T)) \mid y^a_{t,x}(s) \in \mathcal{K}, \ \forall s \in [t, T] \text{ and } y^a_{t,x}(T) \in \mathcal{C} \Big\}, \tag{4.4}$$

with the convention that $\inf \varnothing = +\infty$ and where $y^a_{t,x}(\cdot)$, the unique absolutely continuous trajectory satisfying (4.3), is said to be admissible if it remains in $\mathcal{K}$ and reachs $\mathcal{C}$ at the final time T. $\mathcal{K}$ and $\mathcal{C}$ are two non-empty closed subsets of $\mathbb{R}^d$ representing respectively the constraints set and the target.

Recall that in general, when $\mathcal{K} \neq \mathbb{R}^d$ or $\mathcal{C} \neq \mathbb{R}^d$ and without assuming further controllability assumptions, the value function v may become discontinuous and its characterization as the unique viscosity solution of an appropriate HJ equation is not guaranteed, see [124, 70, 85, 26]. Moreover, v may take infinite values since the set of admissible trajectories may be empty and its domain of definition (the set where it takes finite values) is not known a priori. Similarly to chapter 3, we follow in this chapter the level set approach, developed in [5], which consists in characterizing v by means of an auxiliary control problem free of state constraints. This approach allows to determinate the domain of definition of v and to compute its value. Even though the latter is infinite, one can exploit the auxiliary problem to compute trajectories that minimize the cost functional in (4.4) and remain as close as possible to the sets of constraints.

First, since the sets $\mathcal{K}$ and $\mathcal{C}$ are closed, there exist two Lipschitz continuous functions verifying the following characterization:

$$\forall y \in \mathbb{R}^d, \quad g(y) \leq 0 \iff y \in \mathcal{K} \quad \text{and} \quad \Psi(y) \leq 0 \iff y \in \mathcal{C}.$$

Denote by L_g and L_Ψ the Lipschitz constant of g and Ψ respectively. For instance, g and Ψ can be chosen, respectively, as the signed distance to $\mathcal{K}$ and the signed distance to $\mathcal{C}$.

Then, the auxiliary control problem corresponding to (4.4) is defined, for $(t,x,z) \in [0,T] \times \mathbb{R}^d \times \mathbb{R}$, as follows:

$$w(t,x,z) := \inf_{a(\cdot)\mathcal{A}} \left\{ \Big(\int_t^T \ell(y^a_{t,x}(s), a(s))ds + \Phi(y^a_{t,x}(T)) - z \Big) \bigvee \Big(\max_{s\in[t,T]} g(y^a_{t,x}(s)) \Big) \bigvee \Psi(y^a_{t,x}(T)) \right\}. \quad (4.5)$$

The following result gives a characterization of w and presents its link with the constrained problem (4.4). We refer to [5, 6] for more details and the proof of this result.

Proposition 4.2.1. *Assume that* **(H4.1)**-**(H4.3)** *hold then:*

(i) w *is a locally Lipschitz continuous function and is the unique viscosity solution of the following HJ equation:*

$$\begin{cases} \min\left(-\partial_t w(t,x,z) + H(x, D_x w(t,x,z), \partial_z w(t,x,z)), w(t,x,z) - g(x)\right) = 0, & \text{on } [0,T[\times\mathbb{R}^d \times \mathbb{R}, \\ w(T,x,z) = (\Phi(x) - z) \bigvee g(x) \bigvee \Psi(x), & \text{on } \mathbb{R}^d \times \mathbb{R}, \end{cases}$$

where the Hamiltonian H is given by:

$$H(x,p,p') := \max_{a\in A} \Big\{ -\langle f(x,a), p\rangle + p'\ell(x,a) \Big\}, \quad \text{for} \quad (x,p,p') \in \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}.$$

(ii) *Assume that* **(H4.4)** *also holds. For any $(t,x) \in [0,T] \times \mathbb{R}^d$, we have:*

$$v(t,x) = \inf\{z \in \mathbb{R} \mid w(t,x,z) \leq 0\},$$

and if $\overline{z} := v(t,x) < +\infty$ then any optimal solution of (4.5) with $z = \overline{z}$ is optimal for the constrained problem (4.4).

Assertion (ii) of the above Proposition shows the relevance of considering the problem formulation (4.5) in order to solve the original constrained problem (4.4). Moreover, the characterization of the auxiliary function w, as unique Lipschitz continuous solution of an HJ equation (i), provides a very useful framework for numerical approximation of w and the control feedback. However, it is known that the classical numerical methods for solving HJ equations, such as finite differences ([51]) and semi-Lagrangian

([61, 63]) are grid-dependent since computations are based on a discretization of the state space. This makes this approach applicable only for problems where the dimension of the state variable is low (curse of dimensionality) and henceforth it may seem unreasonable to apply it for the auxiliary problem where the state vector has been increased by one more variable (the auxiliary variable z). In this chapter, we propose a different approach which, instead of considering a grid in the state space, proposes an adaptive refinement of the control space. This approach becomes then interesting for many applications in high dimensional state space and with low dimensional control space.

Now, we will define the discrete setting allowing to approximate the auxiliary value function w. In order to simplify the presentation, from now on, the initial time for the controlled system is set to 0. First, for $N \geq 1$, consider a uniform partition of $[0, T]$ with N time steps: $s_0 = 0, ..., s_k = k\Delta t, ..., s_N = T$, where $\Delta t = \frac{T}{N}$ is the time steps size. Let F be a discrete dynamics associated to f and defined, through the Heun scheme, as follows:

$$F(x, a) := x + \frac{\Delta t}{2}\Big(f(x, a) + f(x + \Delta t f(x, a), a)\Big).$$

The discrete dynamical system, associated to (4.3), is given by:

$$\begin{cases} y_{k+1} = F(y_k, a_k), \quad k = 0, ..., N-1, \\ y_0 = x, \end{cases} \tag{4.6}$$

where $a := (a_k)_k \in A^N$ is a finite sequence of actions. The solution of (4.6), representing the discrete trajectory of the system, will be denoted by $(y_k^a)_k$. This trajectory is admissible if:

$$y_k^a \in \mathcal{K}, \quad \text{for any } \; k = 0, ..., N \quad \text{and} \quad y_N^a \in \mathcal{C}.$$

Moreover, consider an instantaneous cost ρ that approximates the integral of ℓ over an interval $[s_k, s_{k+1}]$, for $k = 0, ..., N-1$, by a quadrature rule, as follows:

$$\rho(x, a) := \frac{\Delta t}{2}\Big(\ell(x, a) + \ell(F(x, a), a)\Big).$$

Under assumptions **(H4.1)** and **(H4.2)**, the functions F and ρ are Lipschitz continuous:

$$\begin{aligned} \|F(y, a) - F(y', a')\| &\leq L_{F,x}\|y - y'\| + L_{F,a}\|a - a'\|, \\ |\rho(y, a) - \rho(y', a')| &\leq L_{\rho,x}\|y - y'\| + L_{\rho,a}\|a - a'\|, \end{aligned}$$

where

$$L_{F,x} := 1 + \Delta t L_{f,x}(1 + \frac{\Delta t}{2}L_{f,x}), \quad L_{F,a} := \Delta t L_{f,a}(1 + \frac{\Delta t}{2}L_{f,x}L_{f,a}), \tag{4.7}$$

and

$$L_{\rho,x} := \Delta t L_{\ell,x}\Big(1 + L_{F,x}\Big), \quad L_{\rho,a} := \Delta t\Big(L_{\ell,a} + \frac{1}{2}L_{\ell,x}L_{F,a}\Big). \tag{4.8}$$

The discrete auxiliary control problem, free of state constraints, is defined as follows:

$$W(x, z) := \inf_{(a_k)_k \in A^N} J(x, z, a), \tag{4.9}$$

where, for $(x, z) \in \mathbb{R}^d \times \mathbb{R}$, the cost functional J is defined by

$$J(x, z, a) := \Big(\sum_{k=0}^{N-1} \rho(y_k^a, a_k) + \Phi(y_N^a) - z\Big) \bigvee \Big(\max_{0 \leq k \leq N} g(y_k^a)\Big) \bigvee \Psi(y_N^a). \tag{4.10}$$

It is worth to mention that $W(x, z)$ converges to $w(0, x, z)$, over compact subsets of $\mathbb{R}^d \times \mathbb{R}$, as $N \to +\infty$ (i.e. $\Delta t \to 0$). More details and quantitative results about time approximation of finite horizon value functions and its convergence order can be found in [11, 64]. Moreover, the sequence of discrete time optimal

trajectories (for $N \geq 1$) provide convergent approximations of optimal trajectories of the continuous time auxiliary problem (4.5) and we refer to [6] for a precise claim and its proof.

In the particular case when the problem is free of state constraints, i.e. $\mathcal{K} = \mathcal{C} = \mathbb{R}^d$, a discrete time approximation of problem (4.4) can be solved by applying optimistic planning methods for finite time horizon, see for instance [131, 100, 75]. In our setting and in order to compute an approximation of W, we extend the optimistic planning approach to deal with maximum cost functions in finite horizon (which is the case of the cost functional J).

4.3 Preliminary results

The aim of this section is to give some preliminary results that will be useful to present our numerical methods and to analyse its convergence.

We start by the following Proposition which states that J satisfies some Lipschitz property w.r.t. the discrete control sequence.

Proposition 4.3.1. *Assume that hypothesis* **(H4.1)**, **(H4.2)** *and* **(H4.3)** *hold. For* $(x, z) \in \mathbb{R}^d \times \mathbb{R}$ *and* $(a_k)_k$, $(\bar{a}_k)_k \in A^N$, *we have the following estimate:*

$$|J(x, z, a) - J(x, z, \bar{a})| \leq \sigma := \Big(\sum_{k=0}^{N-1} \alpha_k \|a_k - \bar{a}_k\| \Big) \bigvee \Big(\sum_{k=0}^{N-1} \beta_k \|a_k - \bar{a}_k\| \Big) \tag{4.11}$$

where

$$\alpha_k := L_{\rho,x} L_{F,a} \frac{L_{F,x}^{N-k-1} - 1}{L_{F,x} - 1} + L_{\rho,a} + L_\Phi L_{F,a} L_{F,x}^{N-k-1},$$

and

$$\beta_k := \Big(L_g \bigvee L_\Psi \Big) L_{F,a} L_{F,x}^{N-k-1}.$$

Proof. We start the proof by introducing the following Lemma:

Lemma 4.3.2. *Given* $y_{k+1} = F(y_k, a_k)$ *and* $\bar{y}_{k+1} = F(\bar{y}_k, \bar{a}_k)$, *for* $0 \leq k \leq N-1$, *with* $y_0 = \bar{y}_0 = x$, *we have*

$$\|y_k - \bar{y}_k\| \leq L_{F,a} \sum_{j=0}^{k-1} L_{F,x}^{k-1-j} \|a_j - \bar{a}_j\|, \quad 0 \leq k \leq N.$$

Now, consider $(x, z) \in \mathbb{R}^d \times \mathbb{R}$, $(a_k)_k, (\bar{a}_k)_k \in A^N$ and the corresponding costs $J(a) := J(x, z, a)$ and $J(\bar{a}) := J(x, z, \bar{a})$. It holds that:

$$|J(a) - J(\bar{a})| \leq \Big(\sum_{k=0}^{N-1} L_{\rho,x} \|y_k - \bar{y}_k\| + \sum_{k=0}^{N-1} L_{\rho,a} \|a_k - \bar{a}_k\| + L_\Phi \|y_N - \bar{y}_N\| \Big)$$
$$\bigvee \Big(L_g \max_{0 \leq k \leq N} \|y_k - \bar{y}_k\| \Big) \bigvee \Big(L_\Psi \|y_N - \bar{y}_N\| \Big).$$

From Lemma 4.3.2, we obtain:

$$\begin{aligned} \sum_{k=0}^{N-1} \|y_k - \bar{y}_k\| &\leq L_{F,a} \sum_{k=0}^{N-1} \sum_{j=0}^{k-1} L_{F,x}^{k-1-j} \|a_j - \bar{a}_j\| \\ &\leq L_{F,a} \sum_{k=0}^{N-1} \frac{L_{F,x}^{N-k-1} - 1}{L_{F,x} - 1} \|a_k - \bar{a}_k\|, \end{aligned}$$

and for any $k = 0, ..., N$

$$\|y_k - \bar{y}_k\| \leq \sum_{j=0}^{k-1} \beta_{k,j} \|a_j - \bar{a}_j\|, \quad \text{where } \beta_{k,j} := L_{F,a} L_{F,x}^{k-1-j}.$$

On the other hand, we observe that the function $b(k) := \sum_{j=0}^{k-1} \beta_{k,j} \|a_j - \bar{a}_j\|$ is non-decreasing with respect to $k \geq 0$, hence we will use the bound $\max_{0 \leq k \leq N} b(k) \leq b(N)$, in order to deduce that:

$$\Big(L_g \max_{0 \leq k \leq N} \|y_k - \bar{y}_k\|\Big) \bigvee \Big(L_\Psi \|y_N - \bar{y}_N\|\Big) \leq \Big(L_g \bigvee L_\Psi\Big) L_{F,a} \sum_{k=0}^{N-1} L_{F,x}^{N-k-1} \|a_k - \bar{a}_k\|.$$

Combining the above inequalities gives the desired result (4.11). □

An estimate of the difference of performances is also established in [39, 38] for the case of an infinite horizon sum with a positive discount factor and under a boundedness assumption on the instantaneous reward. Here, the cost functional J is defined, in finite horizon, as a maximum of several terms. Henceforth, the error term σ, see (4.11), is also defined by taking the maximum between a first term corresponding to the cost functional and a second term representing the constraints.

Now, we give a second result that establishes a larger upper bound on the difference of performances between two different control sequences.

Corollary 4.3.3. *There exists a constant $C > 0$, independent of N, such that for any $(x, z) \in \mathbb{R}^d \times \mathbb{R}$ and $(a_k)_k, (\bar{a}_k)_k \in A^N$ we have:*

$$|J(x, z, a) - J(x, z, \bar{a})| \leq \sigma \leq \delta := C\Delta t \sum_{k=0}^{N-1} \Big(\frac{1}{L_{F,x}}\Big)^k \|a_k - \bar{a}_k\|. \tag{4.12}$$

Proof. For sake of simplicity and only in this proof, denote by $L := L_{F,x}$. From (4.7), we know that $L = 1 + \Delta t L'$ with $L' := L_{f,x}(1 + \frac{\Delta t}{2} L_{f,x})$. Hence, for any $k \in \{0, ..., N-1\}$:

$$\frac{L^{N-k-1} - 1}{L - 1} \leq L^{-k} \frac{L^{N-1}}{L - 1} \leq L^{-k} \frac{e^{N\Delta t L'}}{\Delta t L'}.$$

Moreover, from the definitions of $L_{F,a}$ and $L_{\rho,x}$, we deduce:

$$L_{\rho,x} L_{F,a} \frac{L^{N-k-1} - 1}{L - 1} \leq C_{1,1} \Delta t L^{-k},$$

with $C_{1,1} := \frac{L_{f,a} L_{\ell,x}}{2 L_{f,x}} \Big(2 + T L_{f,x}(1 + \frac{T}{2} L_{f,x})\Big) \Big(1 + \frac{T}{2} L_{f,x} L_{f,a}\Big) e^{T L_{f,x}(1 + \frac{T}{2} L_{f,x})}$ is a constant independent of N and where the last inequality holds by using the fact that

$$L \leq 1 + T L_{f,x}(1 + \frac{T}{2} L_{f,x}) \quad \text{and} \quad L_{f,x} \leq L' \leq L_{f,x}(1 + \frac{T}{2} L_{f,x}).$$

Now, from (4.8), we get:

$$L_{\rho,a} = L_{\rho,a} L^{-k} L^k \leq L_{\rho,a} L^{-k} e^{k\Delta t L'} \leq C_{1,2} \Delta t L^{-k},$$

with $C_{1,2} := \Big(L_{\ell,a} + \frac{1}{2} L_{\ell,x} L_{f,a} T(1 + \frac{T}{2} L_{f,x} L_{f,a})\Big) e^{T L_{f,x}(1 + \frac{T}{2} L_{f,x})}$ and where the last inequality holds since $L_{F,a} \leq L_{f,a} T(1 + \frac{T}{2} L_{f,x} L_{f,a})$. With similar arguments, we obtain:

$$L_\Phi L_{F,a} L^{N-k-1} \leq C_{1,3} \Delta t L^{-k} \quad \text{with} \quad C_{1,3} := L_\Phi L_{f,a} \Big(1 + \frac{T}{2} L_{f,x} L_{f,a}\Big) e^{T L_{f,x}(1 + \frac{T}{2} L_{f,x})}.$$

Finally, by setting $C_1 := C_{1,1} + C_{1,2} + C_{1,3}$, we conclude that $\alpha_k \leq C_1 \Delta t (\frac{1}{L})^k$, for $k = 0, ..., N-1$. By using similar arguments, we obtain an upper bound on β_k as follows:

$$\beta_k \leq C_2 \Delta t (\frac{1}{L})^k, \quad \text{with} \quad C_2 := \Big(L_g \bigvee L_\Psi\Big) L_{f,a} \Big(1 + \frac{T}{2} L_{f,x} L_{f,a}\Big) e^{T L_{f,x}(1+\frac{T}{2}L_{f,x})}.$$

In conclusion, we get the desired result by setting $C := C_1 \bigvee C_2$. □

Remark 4.3.4. *The upper bound δ, defined in (4.12) with $\frac{1}{L_{F,x}} < 1$, is quite similar to the bound established in [39, 38] for infinite time horizon problems. This new upper bound will be useful to prove the convergence results of our planning algorithms in the following section.*

4.4 Optimistic planning

In order to simplify the presentation and without loss of generality, we suppose that the control is of dimension $q = 1$ and we take $A = [0, 1]$. The unit interval here can be obtained by normalizing any compact set of $\mathbb{R}$. Furthermore, our approach can be generalized to control variables with multiple dimensions.

4.4.1 Optimistic planning approach

Planning algorithms are based on the principles of optimistic optimisation (see [56]). In order to minimize the objective function J, we refine, iteratively and in an optimistic way, the global search space A^N into smaller search spaces.

A subset of the global search space A^N, called a node and denoted by $\mathbb{A}_i$ with $i \in \mathbb{N}$, is a cartesian product of sub-intervals of A i.e. $\mathbb{A}_i := A_{i,0} \times A_{i,1} \times ... \times A_{i,N-1} \subseteq A^N$, where $A_{i,k} \subset [0, 1]$ represents the control interval at time step k, for $k = 0, ..., N-1$. The collection of nodes will be organized into a tree Υ that will be constructed progressively by expanding the tree nodes. Expanding a node $\mathbb{A}_i$ consists of choosing an interval $A_{i,k}$, for $k = 0...N-1$, and splitting it uniformly to M sub-intervals where $M > 1$ is a fixed parameter of the algorithm.

In figure 4.1, we represent a simple example of a tree construction to explain how it works.

At the beginning, the tree contains only the root $\mathbb{A}_0$ ($\mathbb{A}_0 = A^N$), from which we generate 3 children nodes $\mathbb{A}_1 = [0, \frac{1}{3}] \times A^{N-1}$, $\mathbb{A}_2 = [\frac{1}{3}, \frac{2}{3}] \times A^{N-1}$ and $\mathbb{A}_3 = [\frac{2}{3}, 1] \times A^{N-1}$, after splitting its first interval. Then, suppose that $\mathbb{A}_1$ is the second node to be refined by splitting its second interval. Hence, we get $\mathbb{A}_4 = [0, \frac{1}{3}] \times [0, \frac{1}{3}] \times A^{N-2}$, $\mathbb{A}_5 = [0, \frac{1}{3}] \times [\frac{1}{3}, \frac{2}{3}] \times A^{N-2}$ and $\mathbb{A}_6 = [0, \frac{1}{3}] \times [\frac{2}{3}, 1] \times A^{N-2}$. Finally, we choose $\mathbb{A}_6$ and we split its first interval $[0, \frac{1}{3}]$ to generate $\mathbb{A}_7 = [0, \frac{1}{9}] \times [\frac{2}{3}, 1] \times A^{N-2}$, $\mathbb{A}_8 = [\frac{1}{9}, \frac{2}{9}] \times [\frac{2}{3}, 1] \times A^{N-2}$ and $\mathbb{A}_9 = [\frac{2}{9}, \frac{1}{3}] \times [\frac{2}{3}, 1] \times A^{N-2}$. In this example, the order of expanding the nodes and splitting the intervals is arbitrary. The true selection rules will be clarified later.

Some useful notations related to the tree Υ

- We associate, for any node $\mathbb{A}_i \in \Upsilon$, a sample control sequence $a_i := (a_{i,k})_{k=0}^{N-1}$ such that $a_{i,k}$ corresponds to the midpoint of the interval $A_{i,k}$ for any $k = 0, ..., N-1$.
- Denote by $d_{i,k}$, for $k = 0, ..., N-1$, the diameter of the interval $A_{i,k}$ of some node $\mathbb{A}_i \in \Upsilon$. For example, in figure 4.1, we have $d_{7,0} = d_{8,0} = d_{9,0} = \frac{1}{9}$, $d_{7,1} = d_{8,1} = d_{9,1} = \frac{1}{3}$ and $d_{7,k} = d_{8,k} = d_{9,k} = 1$ for $k \geq 2$.
- For any node $\mathbb{A}_i \in \Upsilon$ corresponds a split function $s_i(\cdot)$ such that $s_i(k)$ indicates the number of splits needed to obtain the interval $A_{i,k}$ for $k = 0, ..., N-1$. For example, in figure 4.1, $s_0(\cdot) \equiv 0$ for the root $\mathbb{A}_0$. As for $\mathbb{A}_7$, we have $s_7(0) = 2$, $s_7(1) = 1$ and $s_7(k) = 0$ for $k \geq 2$.

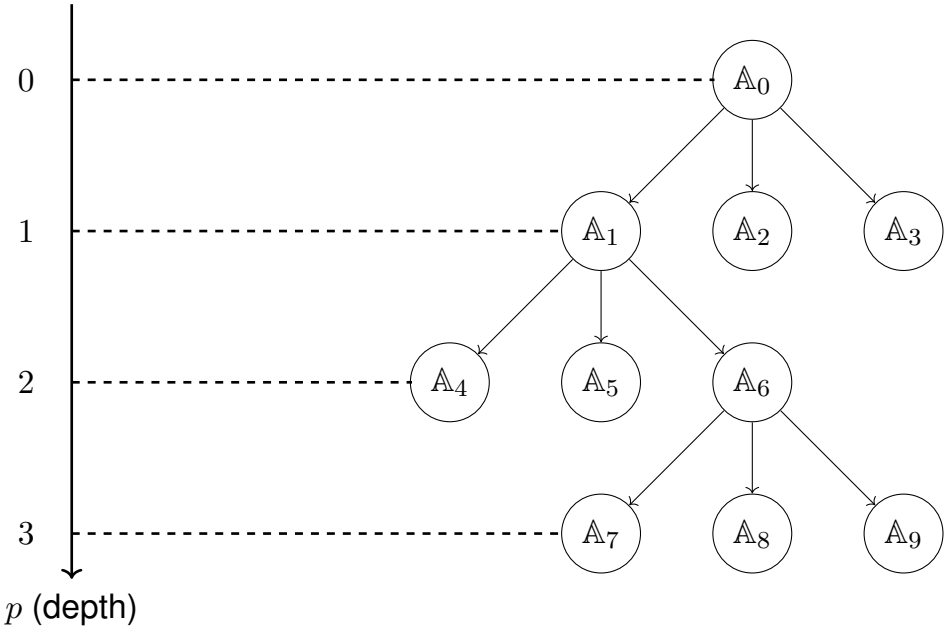

Figure 4.1: Illustrative example of refinement of A^N with $M = 3$ after splitting 3 nodes $\mathbb{A}_0$, $\mathbb{A}_1$ and $\mathbb{A}_6$.

- The depth of a node $\mathbb{A}_i$, denoted by p_i, is to the total number of splits effectuated to obtain this node:

$$p_i := \sum_{k=0}^{N-1} s_i(k). \tag{4.13}$$

- A node $\mathbb{A}_i$ is called a tree leaf if it has not been expanded. The set of the tree leaves is denoted by Λ. In figure 4.1, at this level the set of the tree leaves is formed by $\Lambda = \{\mathbb{A}_2, \mathbb{A}_3, \mathbb{A}_4, \mathbb{A}_5, \mathbb{A}_7, \mathbb{A}_8, \mathbb{A}_9\}$.
- Finally, we denote by Λ_p the set of leaves of Υ at some depth $p \in \mathbb{N}$:

$$\Lambda_p := \Big\{ \mathbb{A}_i \in \Upsilon \quad \text{s.t.} \quad p_i = p \Big\}.$$

Remark 4.4.1. *The control value $a_{i,k}$ can be selected somewhere else in the corresponding interval $A_{i,k}$ for any $\mathbb{A}_i \in \Upsilon$ and $k = 0, ..., N-1$.*

Remark 4.4.2. *We can express the intervals diameters by means of the split function as follows: $d_{i,k} = M^{-s_i(k)}$.*

Remark 4.4.3. *By selecting the control sequence a_i at the centers of intervals of the node $\mathbb{A}_i$ and by taking M odd, we guarantee that after expanding $\mathbb{A}_i$ we generate at least one node $\mathbb{A}_j$ with $J(x, z, a_j) \leq J(x, z, a_i)$. Indeed, the middle child $\mathbb{A}_j$ contains the same control sequence as $\mathbb{A}_i$ hence $J(x, z, a_j) = J(x, z, a_i)$.*

Thanks to Proposition 4.3.1 and Corollary 4.3.3, we get a lower and an upper bound on the optimal value $W(x, z)$ of the discrete auxiliary problem.

Proposition 4.4.4. *By the tree construction, we guarantee that there exists at least a leaf node $\mathbb{A}_i \in \Lambda$ containing an optimal control sequence and verifying:*

$$J(x, z, a_i) - \sigma_i \leq W(x, z) \leq J(x, z, a_i) \tag{4.14}$$

where a_i is the sample control sequence associated to the node $\mathbb{A}_i$ and

$$\sigma_i := \frac{1}{2}\Big(\sum_{k=0}^{N-1}\alpha_k d_{i,k}\Big)\bigvee\Big(\sum_{k=0}^{N-1}\beta_k d_{i,k}\Big), \tag{4.15}$$

with α_k and β_k are given in Proposition 4.3.1.

Proof. By construction of the tree Υ, the set of leaves covers the entire space A^N. Therefore, there exists at least a leaf node $\mathbb{A}_i \in \Lambda$ that contains an optimal control sequence $a^* := (a_k^*)_k \in A^N$. From Proposition 4.3.1 and since $W(x,z) = J(x,z,a^*)$, we get:

$$|J(x,z,a_i) - W(x,z)| = |J(x,z,a_i) - J(x,z,a^*)| \leq \Big(\sum_{k=0}^{N-1}\alpha_k|a_{i,k} - a_k^*|\Big)\bigvee\Big(\sum_{k=0}^{N-1}\beta_k|a_{i,k} - a_k^*|\Big),$$

where a_i is the sample control sequence associated to the node $\mathbb{A}_i$. Moreover, since a_i is chosen at the centers of the intervals of $\mathbb{A}_i$, we obtain $|a_{i,k} - a_k^*| \leq \frac{d_{i,k}}{2}$, for any $k = 0, ..., N-1$. It follows that

$$|J(x,z,a_i) - W(x,z)| \leq \sigma_i.$$

□

In the optimistic planning algorithms, at each iteration, one or several optimistic nodes are chosen and expanded to generate from each node M children. The choice of the optimistic nodes is based on some criterion that we explain hereafter.

Recall that to expand a node $\mathbb{A}_i$, we choose an interval from $A_{i,0} \times A_{i,1} \times ... \times A_{i,N-1}$ and we partition it uniformly to M sub-intervals. If we choose to split the interval $A_{i,k}$, for some $k = 0, ..., N-1$, then we will generate M nodes with an error term $\sigma_i^+(k)$ defined by:

$$\sigma_i^+(k) := \Big(\sum_{j=0, j\neq k}^{N-1}\alpha_j d_{i,j} + \alpha_k\frac{d_{i,k}}{M}\Big)\bigvee\Big(\sum_{j=0, j\neq k}^{N-1}\beta_j d_{i,j} + \beta_k\frac{d_{i,k}}{M}\Big).$$

Henceforth, in order to minimize the error $\sigma_i^+(k)$, the best choice of the interval to split, k_i^*, is defined by:

$$k_i^* := \underset{0\leq k\leq N-1}{\text{argmin}}\ \sigma_i^+(k). \tag{4.16}$$

The following result gives an upper bound on the error term σ_i, of any node $\mathbb{A}_i \in \Upsilon$. This result will be used later in the convergence analysis of optimistic planning algorithms. The proof of this Theorem is given in Appendix 4.7.1.

Theorem 4.4.5. *Assume that $M > L_{F,x} > 1$ and $N \geq 2\tau$ where $\tau := \lceil \log(M)/\log(L_{F,x})\rceil$. For any node $\mathbb{A}_i \in \Upsilon$ at some depth $p := p_i$ large enough, the corresponding error σ_i is bounded as follows:*

$$\sigma_i \leq \delta_p := c_1(N)\Delta t M^{-\frac{p}{N}}, \tag{4.17}$$

where $c_1(N) := C\frac{1}{1-\frac{M^{\frac{1}{\tau}}}{L_{F,x}}}M^{q(N)}$ with $q(N) := 2 - (N-1)\frac{\tau-2}{2\tau(\tau-1)}$ and $C > 0$ is a real constant independent of N and p.

Notice that, by definition, we have $\tau \geq 2$ and $\frac{M^{\frac{1}{\tau}}}{L_{F,x}} \leq 1$. Hence (4.17) is meaningful only when the strict inequality $\frac{M^{\frac{1}{\tau}}}{L_{F,x}} < 1$ holds. Moreover, $q(N) \leq 2$ hence $c_1(N)$ is bounded independently of N.

We introduce some additional definitions that will be useful for the convergence analysis of the algorithms. At any depth $p \geq 0$ of the tree Υ, let Υ_p^* be defined as follows:

$$\Upsilon_p^* := \{\mathbb{A}_i \in \Upsilon \text{ at depth } p \mid J(x,z,a_i) - \delta_p \leq W(x,z)\},$$

where δ_p is defined as in (4.17). We will see later that optimistic planning algorithms will expand only nodes in Υ_p^*, for $p \geq 0$. Finally, we define the *asymptotic branching factor* m (see [39]):

Definition 4.4.6. *The asymptotic branching factor m is the smallest real in $[1, M]$ such that there exists $R \geq 1$ verifying:*

$$|\Upsilon_p^*| \leq Rm^p, \quad \text{for any depth } p \geq 0 \text{ of the tree } \Upsilon,$$

where $|\Upsilon_p^*|$ denotes the cardinality of Υ_p^*.

Remark 4.4.7. *The asymptotic branching factor m lies in $[1, M]$ because at any depth p, there is at least one node in Υ_p^* (the one containing the optimal solution) and at most M^p. Moreover, m measures the complexity of the optimistic algorithms. Indeed, the nodes that will be expanded are contained in Υ_p^*, $p \geq 0$, whose size is bounded by Rm^p. A problem with low resolution complexity will correspond to a small value of m.*

4.4.2 Optimistic Planning (*OP*) Algorithm

At each iteration, we select from the tree leaves, the node $\mathbb{A}_{i^*}$ minimizing the lower bound $(J(x,z,a_i) - \sigma_i)$ and we split it to produce M children, see Algorithm 4.1. More precisely, we identify the interval $A_{i^*,k_{i^*}^*}$ whose partition in M sub-intervals will produce the lowest error $\sigma_{i^*}^+(k_{i^*}^*)$, as described in (4.16).

Algorithm 4.1: Optimistic Planning (*OP*)

Require: The initial state x, the auxiliary variable z, the number of intervals N, the split factor M, the maximal number of expanded nodes $I_{\max}$.

1: Initialize Υ with a root $\mathbb{A}_0 := A^N$ and set $n = 0$ ($n :=$ number of expanded nodes).
2: **while** $n < I_{\max}$ **do**
3: Select the optimistic node to expand: $\mathbb{A}_{i^*} = \underset{\mathbb{A}_i \in \Lambda}{\operatorname{argmin}} (J(x,z,a_i) - \sigma_i)$.
4: Select $k_{i^*}^*$, defined in (4.16), the interval to split for the node $\mathbb{A}_{i^*}$.
5: Update Υ by expanding $\mathbb{A}_{i^*}$ along $k_{i^*}^*$ and adding its M children.
6: Update $n = n + 1$.
7: **end while**
8: **return** Control sequence $a_{i^*} \in A^N$ of the node $\mathbb{A}_{i^*} = \underset{\mathbb{A}_i \in \Lambda}{\operatorname{argmin}} J(x,z,a_i)$ and $J^*(x,z) = J(x,z,a_{i^*})$.

In this algorithm, the number of expanded nodes corresponds to the number of elapsed iterations since at each iteration only one node will be expanded. The number $I_{\max}$ represents a maximum available computational resource. The following Lemma characterizes the near-optimality of the *OP* Algorithm by giving a computable bound on its sub-optimality. This result is similar to [38, Proposition 3] and for sake of convenience we give its proof in Appendix 4.7.1.

Lemma 4.4.8. *The OP Algorithm expands only nodes satisfying $J(x,z,a_i) - \sigma_i \leq W(x,z)$ (thus only nodes in $\bigcup_{p \geq 0} \Upsilon_p^*$). Furthermore, the returned value $J(x,z,a_{i^*})$ verifies*

$$0 \leq J(x,z,a_{i^*}) - W(x,z) \leq \sigma_{\min},$$

with $\sigma_{\min}$ is to the smallest computed error value σ_i among all the expanded nodes.

Finally, by combining Theorem 4.4.5 with Lemma 4.4.8, we derive the following result whose proof is given in Appendix 4.7.1.

Theorem 4.4.9. *With the assumptions of Theorem 4.4.5, let $J^*(x,z)$ be the returned value of the OP Algorithm. There exists an upper bound $B(N,M,m,I_{\max})$ that verifies:*

$$0 \leq J^*(x,z) - W(x,z) \leq B(N,M,m,I_{\max}) \underset{I_{\max}\to\infty}{\longrightarrow} 0,$$

with

$$B(N,M,m,I_{\max}) := \begin{cases} c_1(N)\Delta t M^{-\frac{I_{\max}-1}{RN}}, & \text{if} \quad m=1, \\ c_1(N)\Delta t M^{-\log(\frac{(I_{\max}-1)(m-1)}{R})/N\log m}, & \text{else}, \end{cases}$$

where $c_1(N)$ is defined in Theorem 4.4.5 and m, R are given in Definition 4.4.6.

4.4.3 Simultaneous Optimistic Planning (*SOP*) Algorithm

The Simultaneous Optimistic Planning Algorithm expands, at each iteration, several nodes which are supposed to be optimistic, see Algorithm 4.2. Indeed at each depth $p \geq 0$ of the tree, one node minimizing the objective function J is chosen to be expanded. Then, from every selected node an interval is chosen to be split in order to generate M sub-intervals and hence M children.

Algorithm 4.2: Simultaneous Optimistic Planning (*SOP*)

Require: The initial state x, the auxiliary variable z, the number of intervals N, the split factor M, the maximal number of expanded nodes $I_{\max}$ and the maximal depth $P_{\max}$.

Initialize Υ with root $\mathbb{A}_0 := A^N$ and set $n = 0$ ($n :=$ number of expanded nodes).
while $n < I_{\max}$ **do**
  $p = \min_{\mathbb{A}_i \in \Lambda} p_i$: the minimal depth among the tree leaves.
  while $p \leq P_{\max}$ **do**
    Select the optimistic node at depth p: $\mathbb{A}_{i^*} := \underset{\mathbb{A}_i \in \Lambda_p}{\text{argmin}}\, J(x,z,a_i)$.
    Select $k^*_{i^*}$, defined in (4.16), the interval to split for the node $\mathbb{A}_{i^*}$.
    Update Υ by expanding $\mathbb{A}_{i^*}$ along $k^*_{i^*}$ and adding its M children at depth $p+1$.
    Update $p = p+1$ and $n = n+1$.
  end while
end while
return Control sequence $a_{i^*} \in A^N$ of the node $\mathbb{A}_{i^*} := \underset{\mathbb{A}_i \in \Lambda}{\text{argmin}}\, J(x,z,a_i)$ and $J^*(x,z) := J(x,z,a_{i^*})$.

In Algorithm 4.2, $P_{\max}$ denotes the maximal depth that the tree should not exceed in order to avoid an infinite expansion of Υ. Moreover, as in Algorithm 4.1, $I_{\max}$ represents a maximum available computational resource.

Lemma 4.4.10 gives a lower bound on the depth of the deepest expanded optimal node generated by the *SOP* Algorithm. This result can be proven by adapting some ideas from the proof of [39, Lemma 10] and for the convenience of the reader we give its proof in Appendix 4.7.1.

Lemma 4.4.10. *After exploring n nodes, let $p(n)$ be the smallest depth such that:*

$$R\sum_{p'=0}^{p(n)} m^{p'} \geq \frac{n}{P_{\max}}. \tag{4.18}$$

Therefore the SOP Algorithm has expanded an optimal node at depth $p_n^* := \min\{p(n)-1, P_{\max}\}$ and the error of the solution obtained is bounded by $\delta_{p_n^*}$.

Now, combining the lower bound on the depth defined in Lemma 4.4.10 with the upper bound on the error estimate from Theorem 4.4.5 gives the following convergence result of the *SOP* Algorithm whose proof is also postponed to Appendix 4.7.1.

Theorem 4.4.11. *With the assumptions of Theorem 4.4.5, let $J^*(x,z)$ be the returned value of the SOP Algorithm. When choosing $P_{\max} = I_{\max}^{\eta}$, with $\eta \in]0,1[$, there exists an upper bound $B(N,M,m,I_{\max})$ that verifies:*

$$0 \leq J^*(x,z) - W(x,z) \leq B(N,M,m,I_{\max}) \underset{I_{\max}\to\infty}{\longrightarrow} 0,$$

with

$$B(N,M,m,I_{\max}) := \begin{cases} c_1(N)\Delta t M^{-\frac{\sqrt{I_{\max}}}{RN}}, & \text{if} \quad m=1, \\ c_1(N)\Delta t M^{\frac{2}{N}} M^{-\frac{\log((m-1)I_{\max}^{1-\eta}/R)}{N\log m}}, & \text{else}, \end{cases}$$

where $c_1(N)$ is defined in Theorem 4.4.5 and m, R are given in Definition 4.4.6.

4.4.4 Simultaneous Optimistic Planning with Multiple Steps (*SOPMS*) Algorithm

We start by introducing Algorithm 4.3 (*Update Tree SOP*) which is a generic elementary algorithm that describes how some given tree Υ' should be updated in a similar way to the *SOP* Algorithm.

Algorithm 4.3: ***Update Tree SOP***

Require: An initial state y, the auxiliary variable z, a tree Υ', a maximal number of expanded nodes I and a maximal depth P.

Initialize $n = 0$ ($n :=$ number of expanded nodes).
while $n < I$ **do**
  $p = \min_{\mathbb{A}_i \in \Lambda} p_i$: the minimal depth among the tree leaves.
  while $p \leq P$ **do**
    Select the optimistic node at depth p: $\mathbb{A}_{i^*} = \underset{\mathbb{A}_i \in \Lambda_p}{\operatorname{argmin}}\, J(y,z,a_i)$.
    Select $k_{i^*}^*$, defined in (4.16), the interval to split for the node $\mathbb{A}_{i^*}$.
    Update Υ' by expanding $\mathbb{A}_{i^*}$ along $k_{i^*}^*$ and adding its M children at depth $p+1$.
    Update $p = p+1$ and $n = n+1$.
  end while
end while
return Control sequence a_{i^*} of the node $\mathbb{A}_{i^*} = \underset{\mathbb{A}_i \in \Lambda}{\operatorname{argmin}}\, J(y,z,a_i)$ and $J^*(y,z) = J(y,z,a_{i^*})$.

The *SOPMS* Algorithm uses the elementary algorithm *Update Tree SOP* in order to optimize the objective function from an initial state $x \in \mathbb{R}^d$ and an auxiliary variable $z \in \mathbb{R}$, see Algorithm 4.4. To this end, we define the following cost function J_k, starting from a time step k, for $k = 0, ..., N-1$, as follows:

$$J_k(y,z,a) := \Big(\sum_{i=k}^{N-1} \rho(y_i^a, a_i) + \Phi(y_N^a) - z\Big) \bigvee \Big(\max_{k\leq i\leq N} g(y_i^a)\Big) \bigvee \Psi(y_N^a),$$

where $y \in \mathbb{R}^d$, $z \in \mathbb{R}$, $a := (a_i)_i \in A^{N-k}$ and $(y_i^a)_i$ is the discrete trajectory starting from y and associated to the control sequence $(a_i)_i$.

Algorithm 4.4: Simultaneous Optimistic Planning algorithm with Multiple Steps (*SOPMS*)

Require: The initial state x, the auxiliary variable z, the number of intervals N, the split factor M, the total maximal number of expanded nodes $I_{\max}$, the local number of expanded nodes I_{eval}, the maximal depth $P_{\max}$ and the tolerance $\epsilon > 0$.

1: Initialize $k = 0$, $y_0 = x$ and $n = 0$ ($n :=$ total number of expanded nodes).
2: **while** $k \leq N - 1$ **do**
3: Initialize Υ_k with root $\mathbb{A}_0 := A^{N-k}$, select $a^k := (a_i^k)_i \in A^{N-k}$ and set $W_k := J_k(y_k, z, a^k)$.
4: **while** $n < I_{\max}$ **do**
5: Expand I_{eval} nodes to get $W_k' :=$ *Update Tree SOP*$(\Upsilon_k, I_{eval}, P_{\max})$.
6: Update $n = n + I_{eval}$.
7: **if** $n \geq I_{\max}$ **then**
8: Accept the control sequence $(a_k^*, a_{k+1}^*, ..., a_{N-1}^*) \in A^{N-k}$ and go to line 18.
9: **else**
10: **if** $|\frac{W_k' - W_k}{W_k}| \leq \epsilon$ **then**
11: Accept only the first control value $a_k^* \in A$, set $y_{k+1} = F(y_k, a_k^*)$ and $k = k + 1$.
12: **else**
13: $W_k = W_k'$.
14: **end if**
15: **end if**
16: **end while**
17: **end while**
18: **return** Control sequence $a^* := (a_k^*)_k \in A^N$ and $J^*(x, z) = J(x, z, a^*$.

At each time step $k = 0, ..., N - 1$, the *SOPMS* Algorithm optimizes the cost functional J_k over control sequences in A^{N-k}. However, only the first value of the computed control sequence will be exploited to simulate the next system state. In order to reduce the number of expanded nodes and hence the complexity of the algorithm, we run the *Update Tree SOP* algorithm, in line 5, with a local and reduced number of expanded nodes I_{eval}. This parameter I_{eval} is chosen in an heuristic way. When the cost functional cannot anymore be ameliorated (line 10), we cut the optimization procedure for the current time step, we accept only the first control value to simulate the next system state and we move to the next time step. We should mention that there is a compromise in the choice of the parameter I_{eval}. Small values for I_{eval} may generate local minimums for the objective function. However, large values of I_{eval} will increase the complexity of resolution. On the other hand, in a normal case, the condition in line 7 should not be activated in order to allow the algorithm to iterate on all time steps. To this end, the total maximal numbers of expanded nodes $I_{\max}$ should be chosen large enough.

It is worth to mention that by choosing $I_{eval} = I_{\max}$, *SOPMS* Algorithm becomes equivalent to *SOP* and provides the same error estimate. On the other hand, because of its heuristic parameters, I_{eval} and ϵ, we have not established a convergence guarantee for the *SOPMS* Algorithm for the moment as we have done for *OP* and *SOP* algorithms.

4.4.5 Resolution procedure for a constrained problem

We describe here the procedure to get an approximation of the optimal value and an approximated optimal trajectory for the constrained problem (4.4), starting from an initial state $x \in \mathcal{K}$. First, we already know

that, see section 4.2:

$$v(0,x) = \inf\{z \in \mathbb{R} \mid w(0,x,z) \leq 0\}.$$

Then, by studying the evolution of the dynamics f and by bounding efficiently the cost functions ℓ and Φ, one can get two bounds on the auxiliary variable z, $Z_{\min}$ and $Z_{\max}$, verifying:

$$v(0,x) = \inf\{z \in [Z_{\min}, Z_{\max}] \mid w(0,x,z) \leq 0\}.$$

Since $W(x,z)$ converges to $w(0,x,z)$ as $\Delta t \to 0$, see section 4.2, $v(0,x)$ can be approximated by $\inf\{z \in [Z_{\min}, Z_{\max}] \mid W(x,z) \leq 0\}$. Therefore, the computations will be executed, by dichotomy on z, as follows:

- We iterate the auxiliary variable z on $[Z_{\min}, Z_{\max}]$.
- For any encountered value of z, we compute, by *SOP* or *SOPMS*, an approximation of $W(x,z)$ that will be denoted by $J^*(x,z)$.
- We stop this procedure when reaching a given tolerance on some variable bounds of the auxiliary variable.
- We return the smallest encountered value of z verifying $J^*(x,z^*) \leq 0$. This particular value will be denoted by z^*.
- The returned control sequence for the constrained problem (4.4) corresponds to the one that achieves $J^*(x,z^*)$.

4.5 Extension to infinite horizon problems

In this section, we investigate the case of an optimal control problem with infinite time horizon, nonlinear dynamics and state constraints. Throughout this section, assume that f and ℓ satisfy hypothesis **(H4.1)**, **(H4.2)** and **(H4.4)** and let $\mathcal{A}$ be the set of measurable function $a(\cdot) : [0,+\infty[\to A$. For any $a(\cdot) \in \mathcal{A}$, consider the dynamical system:

$$\begin{cases} \dot{y}(s) = f(y(s), a(s)) \ \ a.e. \ \ s \in [0,+\infty[, \\ y(0) = x, \end{cases} \tag{4.19}$$

and denote by $y_x^a(\cdot)$ its solution. This solution, representing the system trajectory, is said to be admissible if it verifies the following state constraints:

$$y_x^a(s) \in \mathcal{K}, \quad \forall s \in [0,+\infty[.$$

We are interested in the following state-constrained optimal control problem:

$$v(x) := \inf_{a(\cdot)\in\mathcal{A}} \left\{ \int_0^{+\infty} e^{-\gamma s} \ell(y_x^a(s), a(s)) ds \mid y_x^a(s) \in \mathcal{K}, \ \ \forall s \in [0,+\infty[\right\} \tag{4.20}$$

with the convention that $\inf \varnothing = +\infty$ and where $\gamma > 0$ corresponds to a discount factor.

In this case, the auxiliary control problem, free of state constraints, is defined, for any $(x,z) \in \mathbb{R}^d \times \mathbb{R}$, by:

$$w(x,z) := \inf_{a(\cdot)\in\mathcal{A}} \left\{ \left(\int_0^{+\infty} e^{-\gamma s} \ell(y_x^a(s), a(s)) ds - z \right) \bigvee \left(\max_{s\in[0,+\infty[} e^{-\gamma s} g(y_x^a(s)) \right) \right\}. \tag{4.21}$$

Under assumptions **(H4.1)**, **(H4.2)** and **(H4.4)**, the constrained problem can be characterized through the auxiliary problem as follows, see [5] for more details:

$$v(x) = \inf\{z \in \mathbb{R} \mid w(x,z) \leq 0\}.$$

Now we will define the discrete setting in order to approximate the auxiliary value function w. For $\Delta t > 0$, consider the uniform partition $s_k = k\Delta t$, $\forall k \geq 0$ and let F and ρ be respectively the discrete dynamics and the instantaneous cost defined as in section 4.2. The discrete dynamical system is given by:

$$\begin{cases} y_{k+1} = F(y_k, a_k), \quad k \geq 0, \\ y_0 = x, \end{cases} \tag{4.22}$$

where $a_k \in A$. The discrete trajectory $(y_k^a)_k$, solution of (4.22) associated with a control sequence $(a_k)_{k\geq 0}$, is admissible if it verifies the following state constraints:

$$y_k^a \in \mathcal{K}, \quad \forall k \geq 0.$$

The discrete auxiliary control problem takes the following form, for $(x, z) \in \mathbb{R}^d \times \mathbb{R}$:

$$W(x,z) := \inf_{(a_k)_k \in A^\infty} \left\{ \left(\sum_{k=0}^{\infty} \lambda^k \rho(y_k^a, a_k) - z \right) \bigvee \left(\max_{k\geq 0} \lambda^k g(y_k^a) \right) \right\}, \tag{4.23}$$

where $\lambda := 1 - \gamma\Delta t$. One can check that $W(x,z)$ converges to $w(x,z)$, over compact subsets of $\mathbb{R}^d \times \mathbb{R}$, as $\Delta t \to 0$. Now, consider a truncation of problem (4.23) with finite time horizon $N \geq 1$:

$$W_N(x,z) := \inf_{(a_k)_k \in A^N} \left\{ \left(\sum_{k=0}^{N-1} \lambda^k \rho(y_k^a, a_k) - z \right) \bigvee \left(\max_{0\leq k\leq N} \lambda^k g(y_k^a) \right) \right\}. \tag{4.24}$$

The next result shows that W_N is a good approximation for $W(x,z)$, with an exponentially decreasing error with respect to N. As a consequence, the algorithms developed in section 4.4 allow to approximate $W_N(x,z)$ and so $W(x,z)$.

Proposition 4.5.1. *Assume that $\rho \geq 0$, $\lambda L_{F,x} < 1$ and $N \geq \max(2, 1 - \log(\lambda L_{F,x}))$. The following bound holds:*

$$0 \leq W(x,z) - W_N(x,z) \leq 2\max(\frac{C_1}{2}, C_g + L_g\|x\| + L_g C_F N)(\lambda L_{F,x})^N, \tag{4.25}$$

where $C_1, C_g, L_g, C_F \geq 0$ will be made explicit in the proof.

Proof. For a given $a = (a_k)_k \in A^\infty$ and a given $(x,z) \in \mathbb{R}^d \times \mathbb{R}$ let

$$e_{N-1}(a) := \sum_{k=0}^{N-1} \lambda^k \rho(y_k^a, a_k) - z, \quad r_N(a) := \max_{0\leq k\leq N} \lambda^k g(y_k^a),$$

$$e_\infty(a) := \sum_{k=0}^{\infty} \lambda^k \rho(y_k^a, a_k) - z \quad \text{and} \quad r_\infty(a) := \max_{k\geq 0} \lambda^k g(y_k^a).$$

Then we can write

$$W_N(x,z) = \inf_{a:=(a_k)_k \in A^\infty} \max(e_{N-1}(a), r_N(a)), \quad \text{and} \quad W(x,z) = \inf_{a:=(a_k)_k \in A^\infty} \max(e_\infty(a), r_\infty(a)).$$

It is clear that $e_{N-1}(a) \leq e_\infty(a)$ (because $\rho \geq 0$), and $r_N(a) \leq r_\infty(a)$, therefore $W_N(x,z) \leq W(x,z)$. Moreover, it holds

$$\begin{aligned} W(x,z) - W_N(x,z) &\leq \sup_{a\in A^\infty} \left\{ \max(e_\infty(a), r_\infty(a)) - \max(e_{N-1}(a), r_N(a)) \right\} \\ &\leq \sup_{a\in A^\infty} \left\{ \max(e_\infty(a) - e_{N-1}(a), r_\infty(a) - r_N(a)) \right\} \end{aligned} \tag{4.26}$$

with $e_\infty(a) - e_{N-1}(a) = \sum_{k=N}^{\infty} \lambda^k \rho(y_k^a, a_k)$, and by using the fact that

$$r_\infty(a) = \max\left(r_N(a), \max_{k\geq N+1} \lambda^k g(y_k^a)\right)$$

we obtain:

$$r_\infty(a) - r_N(a) \leq \max\Big(0, \max_{k\geq N+1} \lambda^k g(y_k^a) - r_N(a)\Big). \tag{4.27}$$

First, by using the Lipschitz continuity of ρ, we have $\rho(y,a) \leq \rho(0,a) + L_{\rho,x}\|y\| + L_{\rho,a}\|a\|$, and since $a \in A$ where A is a compact set, it holds

$$\rho(y,a) \leq C_\rho + L_{\rho,x}\|y\|,$$

for some constant $C_\rho \geq 0$. In the same way, it also holds

$$\|F(y,a)\| \leq C_F + L_{F,x}\|y\|,$$

for some constant $C_F \geq 0$. Hence $\|y_{k+1}^a\| \leq C_F + L_{F,x}\|y_k^a\|$, $\forall k \geq 0$. In particular, it holds, for any $(a_k)_k \in A^\infty$, with $y_0^a = x$:

$$\|y_k^a\| \leq C_F(1 + L_{F,x} + \cdots + L_{F,x}^{k-1}) + L_{F,x}^k\|x\|.$$

Since $L_{F,x} > 1$, we get $1 + L_{F,x} + \cdots + L_{F,x}^{k-1} \leq kL_{F,x}^{k-1}$ and hence we can bound the state y_k^a as follows:

$$\|y_k^a\| \leq C_F k L_{F,x}^{k-1} + L_{F,x}^k\|x\|.$$

Now, let $u_N(\lambda) := \sum_{k\geq N} \lambda^k = \frac{\lambda^N}{1-\lambda}$ for $\lambda < 1$. We have also

$$\lambda u_N'(\lambda) = \sum_{k\geq N} k\lambda^k = \frac{N\lambda^N - (N-1)\lambda^{N-1}}{(1-\lambda)^2} \leq \frac{N\lambda^N}{(1-\lambda)^2} \quad \text{for } \lambda < 1 \text{ and } N \geq 1.$$

Combining the previous bounds implies

$$\begin{aligned} e_\infty - e_{N-1} &= \sum_{k=N}^{\infty} \lambda^k \rho(y_k^a, a_k) \leq \sum_{k=N}^{\infty} \lambda^k (C_\rho + L_{\rho,x}\|y_k^a\|) \\ &\leq C_\rho u_N(\lambda) + L_{\rho,x}(C_F \lambda u_N'(\lambda L_{F,x}) + \|x\| u_N(\lambda L_{F,x})) \\ &\leq (C_\rho + L_{\rho,x}\|x\|)\frac{(\lambda L_{F,x})^N}{1-\lambda L_{F,x}} + C_F L_{\rho,x}\frac{N(\lambda L_{F,x})^N}{(1-\lambda L_{F,x})^2} \leq C_1(\lambda L_{F,x})^N \end{aligned} \tag{4.28}$$

with $C_1 := \frac{(C_\rho + L_{\rho,x}\|x\|)}{1-\lambda L_{F,x}} + C_F \frac{N L_{\rho,x}}{(1-\lambda L_{F,x})^2}$, where we have used the fact that $u_N(\lambda) \leq u_N(\lambda L_{F,x})$ and the hypothesis $\lambda L_{F,x} < 1$. By using the fact that $|g(y)| \leq C_g + L_g\|y\|$ (with $C_g := |g(0)|$), it holds

$$\begin{aligned} \max_{k\geq N+1} \lambda^k |g(y_k^a)| &\leq \max_{k\geq N+1} \lambda^k (C_g + L_g\|y_k^a\|) \\ &\leq C_g \lambda^{N+1} + L_g \max_{k\geq N+1} \Big\{C_F k \lambda^k L_{F,x}^{k-1} + \|x\|\lambda^k L_{F,x}^k\Big\}. \end{aligned}$$

For a given parameter $t \in [0,1]$, using the fact that $k \mapsto kt^k$ is non-increasing for $k \geq -\log(t)$, we see that, for $k \geq N$ and $N \geq -\log(\lambda L_{F,x})$, it holds $k(\lambda L_{F,x})^k \leq N(\lambda L_{F,x})^N$. Henceforth

$$\max_{k\geq N+1} k\lambda^k L_{F,x}^{k-1} \leq \frac{1}{L_{F,x}} \max_{k\geq N} k(\lambda L_{F,x})^k \leq \frac{1}{L_{F,x}} N(\lambda L_{F,x})^N \leq N(\lambda L_{F,x})^N.$$

It follows that

$$\max_{k\geq N+1} \lambda^k |g(y_k^a)| \leq C_N(\lambda L_{F,x})^N. \tag{4.29}$$

where $C_N := C_g\lambda + L_g\|x\| + L_g C_F N$. On the other hand, $r_N(a) \geq \lambda^N g(y_N^a)$. As for the previous bounds, assuming now that $N - 1 \geq -\log(\lambda L_{F,x})$, we have $\lambda^N |g(y_N^a)| \leq C_N(\lambda L_{F,x})^N$. Hence, combined with (4.27) and (4.29), we obtain

$$r_\infty(a) - r_N(a) \leq 2C_N(\lambda L_{F,x})^N.$$

Together with (4.26) and (4.28), we deduce the desired bound (4.25). □

4.6 Numerical experimentation

4.6.1 Choice of the numerical parameters

In section 4.4, we have established some theoretical error bounds relative to the *OP* and *SOP* algorithms (see theorems 4.4.9 and 4.4.11 respectively). Even though those bounds depend on the computational budget $I_{\max}$, they depend also on the asymptotic branching factor m for which we have only a theoretic estimation given in definition 4.4.6. Unfortunately, this definition is not simple to exploit since we have not a precise idea on the computational budget $I_{\max}$ that we should use to reach some given error bound. In addition to that, it is known that the complexity of optimistic planning methods depends on the horizon of resolution which corresponds to the number of intervals N in our case. Indeed, if we solve a discrete problem of length N, with *OP* or *SOP*, by expanding a number of nodes $I_{\max}$, the question is what is the computational budget $I'_{\max}$ to use in order to guarantee at least the same error when solving a discrete problem with a number of intervals $N' \neq N$. Let for example $N' = 2 \times N$.

When using the *OP* Algorithm and from Theorem 4.4.9, $I'_{\max}$ should satisfy the following condition:

$$I'_{\max} = \begin{cases} 2 \times I_{\max}, & \text{if} \quad m = 1 \\ 1 + (M-1) \times I_{\max}^2 & \text{else.} \end{cases}$$

With the *SOP* Algorithm and by exploiting Theorem 4.4.11, $I'_{\max}$ should be chosen as follows:

$$I'_{\max} = \begin{cases} 4 \times I_{\max}, & \text{if} \quad m = 1 \\ (M-1)^{\frac{1}{1-\eta}} \times I_{\max}^2 & \text{else,} \end{cases}$$

where η is defined in Theorem 4.4.11.

We observe that $I'_{\max}$ becomes overestimated when $m > 1$ for both algorithms *OP* and *SOP* which increases the numerical resolution complexity for $N' = 2N$. Henceforth, in our numerical simulations, we will use the following rules to set the algorithms parameters.

1. The computational budget $I_{\max}$: multiply $I_{\max}$ by 10 when the number of intervals N doubles. The reference value for the computational budget is $I_{\max} = 10^3$ for $N = 10$ in the case where the control dimension $q = 1$. However, when $q = 2$ (as in Example 2 in the next section), we start with an initial budget $I_{\max} = 500$, for $N = 10$, since the complexity of algorithms depends on the control dimension. Only for *SOPMS*, thanks to its low complexity (as we will see and explain later), we can set $I_{\max}$ very large (free $I_{\max}$). In this case, the total number of nodes expansions is limited by the choice of the other parameters of *SOPMS* which are I_{eval} and ϵ.

2. The maximal depth to not exceed, $P_{\max}$, when using *SOP* or *SOPMS*: following the assumption given in Theorem 4.4.11, this parameter should be chosen as $P_{\max} := I_{\max}^\eta$ with $\eta \in [0, 1[$. A small value of η leads to the exploration of the tree at its width. However, if η is near to 1, the search will be deep. In our simulations, we set $\eta = 0.5$ hence $P_{\max} = \sqrt{I_{\max}}$.

3. The number of evaluations I_{eval} and the precision ϵ: Those two parameters are specific to *SOPMS* algorithm. When N doubles, we multiply I_{eval} by 10 with a reference value $I_{eval} = 30$ ($\simeq P_{\max}$) for $N = 10$. Moreover, we set $\epsilon = 10^{-6}$.

4. As explained in remark 4.4.3, the splitting factor M should be odd. Here, we set $M = 3$. A larger value of M increases the complexity.

5. The different algorithms are implemented in C++ and all the computations are done with a computer that uses an Intel XEON E5-2695 CPU at 2.4 GHz with 128 Go RAM.

4.6.2 Numerical examples

Example 1: 2D example

Consider a control problem without state constraints and where the state dimension $d = 2$. The dynamics f is given by $f(x, a) = \begin{pmatrix} -x_2 \\ a \end{pmatrix}$ for $x = (x_1, x_2) \in \mathbb{R}^2$ and $a \in A = [-1, 1]$, the time horizon $T = 1$, the distributed and final cost functions are given by:

$$\ell(x, a) = 0 \qquad \text{and} \quad \Phi(x) = \|x\|.$$

Let $V(x)$ be the value function of the discrete problem, starting from the initial state x, that can be computed, with $\Delta t = \frac{T}{N}$ for different values of N[1]. Henceforth, we define the relative error w.r.t. the discrete value function as follows:

$$E_{disc}(x) := \frac{|\mathcal{J}^*(x) - V(x)|}{|V(x)|},$$

where $\mathcal{J}^*(x)$ is the value returned by optimistic planning algorithms. Furthermore, we define the relative errors w.r.t the continuous time optimal control problem as follows:

$$E_{cont}(x) := \frac{|\mathcal{J}^*(x) - v(0, x)|}{|v(0, x)|},$$

where $v(0, x)$ is the exact value function of the continuous problem. Indeed, for any time horizon $T > 0$ the optimal trajectory aims to get closer to the origin $(0, 0)$ and can be computed analytically.[2]

In tables 4.1, we present the relative errors w.r.t. the discrete value function obtained by *OP*. We observe that the *OP* Algorithm does not succeed to keep the same errors when increasing N even for a large number of iterations $I_{\max}$.

[1] The optimal control sequence $(a_k^*)_k$ of the discrete problem is determined as follows:

$$\begin{cases} a_k^* = 1, & k < \bar{k} \\ a_k^* = -1, & k \geq \bar{k}, \end{cases}$$

where $\bar{k}$ is a switching time step.

[2] The optimal trajectory $x^*(\cdot) := (x_1^*(\cdot), x_2^*(\cdot))$, for an initial state $x := (x_1, x_2)$, is defined as follows:

- In $[0, \bar{t}]$, the optimal control is $a^* \equiv 1$, $x_1^*(t) = -\frac{t^2}{2} - x_2 t + x_1$, and $x_2^*(t) = t + x_2$.
- In $[\bar{t}, T]$, the optimal control is $a^* \equiv -1$, $x_1^*(t) = \frac{t^2}{2} - \bar{y}t + \bar{x}_1$ and $x_2^*(t) = -t + \bar{x}_2$.

where $\bar{t}$, $\bar{x}_1$ and $\bar{x}_2$ are deduced from the continuity of the solution on $\bar{t}$:

$$\begin{cases} -\frac{\bar{t}^2}{2} + x_1 = \frac{\bar{t}^2}{2} - \bar{x}_2 \bar{t} + \bar{x}_1, \\ \bar{t} + x_2 = -\bar{t} + \bar{x}_2, \end{cases}$$

and the optimality condition on $\bar{t}$:

$$x_2^*(T) = x_1^*(T)(T - \bar{t}).$$

Parameters		$x=(1,0)$			$x=(1.5,0)$			$x=(2,0)$		
N	$I_{\max}$	$V(x)$	$E_{disc}(x)$	CPU(s)	$V(x)$	$E_{disc}(x)$	CPU(s)	$V(x)$	$E_{disc}(x)$	CPU(s)
10	10^3	6.89 e-01	1.04 e-01	0.0	1.161	5.65 e-02	0.0	1.639	4.42 e-02	0.0
20	10^4	6.88 e-01	1.81 e-01	5.3	1.161	1.03 e-01	5.1	1.639	7.22 e-02	4.9
	2×10^4	6.88 e-01	1.81 e-01	29.6	1.161	1.00 e-01	33.7	1.639	7.22 e-02	33.7
	2×10^5	6.88 e-01	1.47 e-01	7167.9	1.161	8.64 e-02	6987.1	1.639	6.12 e-02	6239.6

Table 4.1: (Example 1): Relative errors w.r.t. the discrete value function obtained by the *OP* Algorithm for different values of N and $I_{\max}$ and from different initial states.

Moreover, we fix the initial state $x=(1.5,0)$ and we present in table 4.2 the number of iterations I needed to reach a relative error w.r.t. the discrete value function less than a given tolerance κ. We deduce from this table that the complexity of the *OP* Algorithm is very high.

N	$\kappa=10^{-2}$		$\kappa=10^{-3}$		$\kappa=10^{-4}$	
	I	CPU(s)	I	CPU(s)	I	CPU(s)
10	—	> 1 day	—	> 1 day	—	> 1 day

Table 4.2: (Example 1): Necessary number of iterations I to reach the error κ by *OP* Algorithm from the initial state $x=(1.5,0)$.

Finally, we present in table 4.3 the relative errors w.r.t. the continuous value function obtained by *OP*. We observe that the error does not decrease when moving from $N=10$ to $N=20$ even when expanding a large number of nodes $I_{\max}=2\times10^5$ which increases enormously the resolution complexity in time. All those observations can be explained by the fact that the *OP* Algorithm consumes the budget $I_{\max}$ in expanding nodes at lower depths of the tree Υ and does not search deeply.

Parameters		$x=(1,0)$		$x=(1.5,0)$		$x=(2,0)$	
N	$I_{\max}$	$E_{cont}(x)$	CPU(s)	$E_{cont}(x)$	CPU(s)	$E_{cont}(x)$	CPU(s)
10	10^3	1.08 e-01	0.0	5.78 e-02	0.0	4.48 e-02	0.0
20	10^4	1.84 e-01	5.8	1.04 e-01	5.8	7.29 e-02	5.2
	2×10^4	1.80 e-01	30.7	9.97 e-02	35.2	7.01 e-02	34.3
	2×10^5	1.50 e-01	7213.4	8.78 e-02	7165.4	6.18 e-02	6414.7

Table 4.3: (Example 1): Relative errors $w.r.t.$ the continuous value function obtained by the *OP* Algorithm for different values of N and $I_{\max}$ and from different initial states.

Now, we present the numerical results obtained by the *SOP* Algorithm. We start by presenting the relative errors w.r.t. the discrete value function in table 4.4. We observe an amelioration of the error estimates when increasing the number of time steps N and the computational budget $I_{\max}$.

Parameters		$x=(1,0)$			$x=(1.5,0)$			$x=(2,0)$		
N	$I_{\max}$	$V(x)$	$E_{disc}(x)$	CPU(s)	$V(x)$	$E_{disc}(x)$	CPU(s)	$V(x)$	$E_{disc}(x)$	CPU(s)
10	10^3	6.89 e-01	9.07 e-03	0.01	1.161	5.81 e-03	0.01	1.639	4.89 e-03	0.01
20	10^4	6.88 e-01	2.14 e-03	1.23	1.161	1.68 e-04	1.29	1.639	4.29 e-04	1.30
40	10^5	6.87 e-01	2.05 e-03	144.5	1.159	4.08 e-05	144.0	1.638	2.95 e-05	152.2

Table 4.4: (Example 1): Relative errors $w.r.t.$ the discrete value function obtained by the *SOP* Algorithm for different values of N and $I_{\max}$ and from different initial states.

Furthermore, we present in table 4.5 the necessary number of iterations I to reach a relative error w.r.t. the discrete value function less than a given tolerance κ for a fixed initial state $x = (1.5, 0)$ by the *SOP* Algorithm. We observe that the complexity increases when N increases for $\kappa = 10^{-2}$ and also for $N = 40$ when decreasing the tolerance κ. On the other hand, we remark that there are some tolerances κ that cannot be easily reached for lower values of N.

N	$\kappa = 10^{-2}$		$\kappa = 10^{-3}$		$\kappa = 10^{-4}$	
	I	CPU(s)	I	CPU(s)	I	CPU(s)
10	30	0.0	—	> 1 day	—	> 1 day
20	58	0.0	90	0.0	—	> 1 day
40	120	0.0	247	0.0	1859	0.1

Table 4.5: (Example 1): Necessary number of iterations I to reach the error κ by *SOP* Algorithm from the initial state $x = (1.5, 0)$.

Parameters		$x = (1, 0)$		$x = (1.5, 0)$		$x = (2, 0)$	
N	$I_{\max}$	$E_{cont}(x)$	CPU(s)	$E_{cont}(x)$	CPU(s)	$E_{cont}(x)$	CPU(s)
10	10^3	1.20 e-02	0.01	7.02 e-03	0.01	5.48 e-03	0.01
20	10^4	2.20 e-03	1.23	1.37 e-03	1.29	1.01 e-03	1.30
40	10^5	2.12 e-03	144.5	6.02 e-05	144.0	5.76 e-05	152.2

Table 4.6: (Example 1): Relative errors $w.r.t.$ the continuous value function obtained by the *SOP* Algorithm for different values of N and $I_{\max}$ and from different initial states.

Finally, we present in table 4.6 the relative errors w.r.t. the continuous value function obtained by the *SOP* Algorithm. In contrary to the *OP* Algorithm, the error decreases for the *SOP* Algorithm when increasing N and $I_{\max}$ for all the initial positions. Those observations are explained by the fact that the *SOP* Algorithm construct the search tree Υ by adding nodes at higher depths recursively.

On the other hand, we observe in table 4.6 that the error decreases slightly for $x = (1, 0)$ compared to $x = (1.5, 0)$ or $x = (2, 0)$. This observation can be explained by the fact that the resolution complexity depends not only on the type of the problem but also on the initial state x.

Now, we study the performance of *SOPMS* Algorithm. By comparing the numerical results presented in tables 4.6 and 4.7, we deduce that, for a fixed N, *SOPMS* Algorithm becomes more precise than the *SOP* Algorithm when $I_{\max}$ is large enough (free $I_{\max}$). Indeed, a large value of $I_{\max}$ allows to the *SOPMS* Algorithm to iterate on all the time steps $k = 0, ..., N-1$ which leads to a better estimation of the solution.

Parameters			$x = (1, 0)$		$x = (1.5, 0)$		$x = (2, 0)$	
N	I_{eval}	$I_{\max}$	$E_{cont}(x)$	CPU(s)	$E_{cont}(x)$	CPU(s)	$E_{cont}(x)$	CPU(s)
10	30	10^3	7.97 e-03	0.0	5.21 e-03	0.0	2.03 e-03	0.0
		free	7.96 e-03	0.0	5.21 e-03	0.0	2.03 e-03	0.0
20	300	10^4	1.26 e-02	0.1	3.44 e-04	0.1	4.66 e-04	0.0
		free	1.37 e-03	0.4	6.73 e-04	0.2	4.65 e-04	0.2
40	3000	10^5	8.69 e-03	34.7	7.73 e-04	13.0	2.93 e-04	7.1
		free	1.33 e-03	100.1	1.32 e-04	80.2	4.44 e-05	18.9

Table 4.7: (Example 1): Relative errors $w.r.t.$ the continuous value function obtained by the *SOPMS* Algorithm with $\epsilon = 10^{-6}$ for different values of N, I_{eval} and $I_{\max}$ and from different initial states.

In table 4.8, we present the relative errors w.r.t. the continuous value function obtained by the *SOPMS* algorithm for different values of the tolerance parameter ϵ.

Parameters		Initial	$\epsilon = 10^{-2}$			$\epsilon = 10^{-4}$			$\epsilon = 10^{-6}$		
N	I_{eval}	state x	$E_{cont}(x)$	CPU(s)	I_{tot}	$E_{cont}(x)$	CPU(s)	I_{tot}	$E_{cont}(x)$	CPU(s)	I_{tot}
10	30	$(1,0)$	1.76 e-02	0.0	750	8.22 e-03	0.0	1050	7.96 e-03	0.0	1200
		$(1.5,0)$	5.21 e-03	0.0	630	5.21 e-03	0.0	900	5.21 e-03	0.0	990
		$(2,0)$	6.11 e-03	0.0	600	2.03 e-03	0.0	780	2.03 e-03	0.0	1080
20	300	$(1,0)$	5.32 e-03	0.0	13200	2.13 e-03	0.1	24300	1.37 e-03	0.4	35700
		$(1.5,0)$	8.51 e-04	0.0	12000	7.90 e-04	0.1	18300	6.73 e-04	0.2	23700
		$(2,0)$	4.97 e-04	0.0	11700	4.97 e-04	0.1	14700	4.65 e-04	0.2	22800
40	3000	$(1,0)$	2.72 e-03	3.1	240 $\times 10^3$	1.63 e-03	57.3	633 $\times 10^3$	1.33 e-03	100.1	801000
		$(1.5,0)$	4.90 e-04	2.5	237$\times 10^3$	2.48 e-04	28.2	462$\times 10^3$	1.32 e-04	80.2	669000
		$(2,0)$	1.06 e-04	2.1	234$\times 10^3$	8.04 e-05	11.0	3$\times 10^5$	4.44 e-05	18.9	384000

Table 4.8: (Example 1): Relative errors $w.r.t.$ the continuous value function obtained by the *SOPMS* algorithm for different values of the tolerance parameter ϵ.

We notice that the *SOPMS* performance does not depend only on $I_{\max}$ but also on the heuristic parameters I_{eval} and ϵ as it is presented in table 4.8 where we choose $I_{\max}$ large enough in such a way that the *SOPMS* algorithm iterate on all the time steps $k = 0, ..., N$. In table 4.8, I_{tot} denotes the total number of expanded nodes which depends on x, I_{eval} and ϵ. We remark that better error estimations are obtained with small values of ϵ which is not a surprising result. Furthermore, we observe that by following our chosen heuristic for I_{eval}, the error decreases when N doubles.

On the other hand, exploiting the CPU time given in the different tables, we remark that the *SOP* algorithm consumes less time than the *OP* algorithm. Indeed, for the *OP* algorithm, the selection of the node to expand, which minimizes the lower bound on the optimal value, is done among all the tree leaves. For this reason, we should iterate over the whole set of the tree leaves. However, the *SOP* algorithm selects the node to expand, which minimizes the criterion value J_i, among only the leaves of a given depth of the tree. Therefore, iterations are done over a subset of the tree leaves with the same depth.

Furthermore, we remark that in general, the *SOPMS* algorithm consumes less time than the *SOP* algorithm. Note that after expanding a given number of nodes $I \geq 1$, the tree contains $(M-1)I+1$ leaves. Therefore, the *SOP* algorithm will construct a tree of $(M-1)I_{\max}+1$ leaves with control and trajectory sequences of lengths N and $N+1$ respectively. However, the *SOPMS* algorithm, for each $k = 0...N-1$, will expand less than $I_{\max}$ nodes. Therefore, for any k, we will construct a tree of fewer leaves comparing to the *SOP* algorithm, with control and trajectory sequences of lengths $N-k$ and $N-k+1$ respectively. On other words, constructing and exploring only one tree of a total number of leaves equal to $(M-1)I_{\max}+1$ is more complex than working on N trees with reduced number of nodes.

Finally, we represent in figure 4.2 the trajectories obtained by the *SOP* Algorithm from different initial positions.

Example 2: Zermelo problem

Consider the Zermelo problem where a boat tries to reach a circular target $\mathcal{C}$ with radius $r_0 > 0$ at time $T > 0$ with minimal fuel consumption. The dynamics is given by:

$$\begin{cases} \dot{x}_1(s) = u(s)\cos(\theta(s)) - bx_2(s)^2 + c, \\ \dot{x}_2(s) = u(s)\sin(\theta(s)), \end{cases}$$

where $u(s) \in [0, u_{\max}]$ and $\theta(s) \in [0, 2\pi]$, for $s \in [0, T]$, denote the speed and the angle of orientation of the boat respectively and the term $c - bx_2^2$ represents the current drift along the x_1-axis.

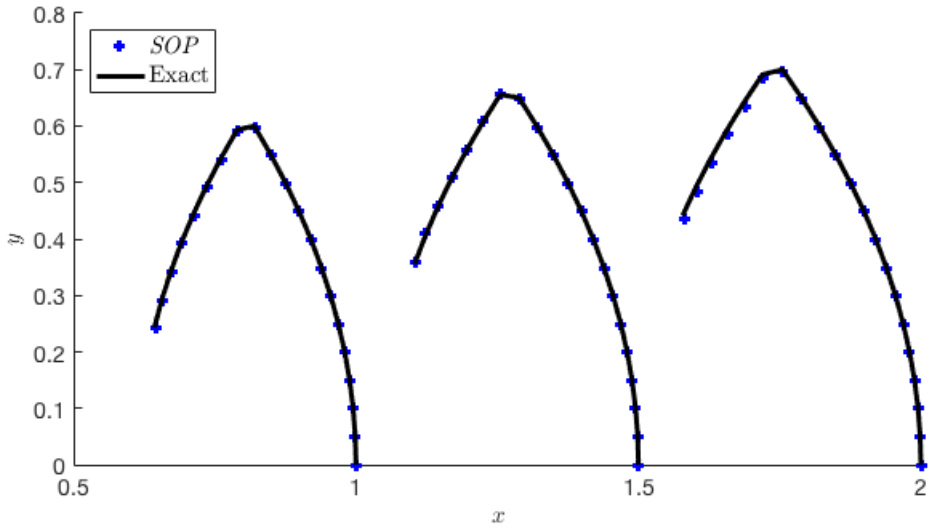

Figure 4.2: (Example 1): Trajectories obtained by the *SOP* algorithm for $N = 20$, $I_{\max} = 10^4$ and from different initial positions.

We set $r_0 = 0.1$, $T = 1$, $u_{\max} = 2.5$, $b = 0.5$ and $c = 2$. The center of the circular target $\mathcal{C}$ is of coordinates $(1.5, 0)$. We consider also two rectangular obstacles with centers $(-0.5, 0.5)$ for the first and $(-1, -1.5)$ for the second one. The horizontal and the vertical radius of the first obstacle are $r_x = r_y = r_1 = 0.4$ and $r_x = r_2 = 0.2$, $r_y = 5r_2$ for the second obstacle.

The cost that we want to minimise is given by $Q(x, a) = \int_0^T u(s)ds$, where x is the initial position of the boat and $a(\cdot) := (u(\cdot), \theta(\cdot))$ is the control. Hence, the discrete cost functional $\mathcal{J}$ becomes given by:

$$\mathcal{J}(x, a) = \sum_{k=0}^{N-1} \Delta t \times u_k, \quad \text{with} \quad \Delta t = \frac{T}{N}.$$

Moreover, the obstacle and the target functions g and Ψ are given by:

$$g(x) := \Big(r_1 - \|x - (-0.5, 0.5)\|_\infty\Big) \bigvee \Big(r_2 - \max(|x_1 + 1|, r_2|x_2 + 1.5|)\Big) \quad \text{and} \quad \Psi(x) = \|x - (1.5, 0)\|_2 - r_0.$$

Henceforth, the discrete auxiliary value function is defined as:

$$W(x, z) = \inf_{(a_k)_k \in A^N} \Big\{ \Big(\mathcal{J}(x, a) - z\Big) \bigvee \Big(\max_{0 \le k \le N} g(y_k^a)\Big) \bigvee \Psi(y_N^a) \Big\},$$

with $A := [0, u_{\max}] \times [0, 2\pi]$, and an approximation of the constrained problem value function is given by:

$$z^* = \inf\{z \in \mathbb{R} \mid W(x, z) \le 0\}.$$

In table 4.9, we represent $\mathcal{J}^*(x)$, the optimal value of the cost functional $\mathcal{J}$, obtained by the *SOP* algorithm for different values of N and $I_{\max}$ and from different initial states x.

We observe in table 4.9 that $\mathcal{J}^*$ decreases when N doubles. In addition to that, we remark that the value of the approximation z^* is very close to $\mathcal{J}^*(x)$.

On the other hand, we represent in figure 4.3 the trajectories obtained by the *SOP* Algorithm from different initial positions. We remark that all the trajectories verify the constraints by avoiding the obstacles and reach the target at the final time step.

Parameters		$x_1 = (-2.5, -1)$			$x_2 = (-2, 0.5)$			$x_3 = (-1.5, 1.5)$		
N	$I_{\max}$	$\mathcal{J}^*(x)$	z^*	CPU(s)	$\mathcal{J}^*(x)$	z^*	CPU(s)	$\mathcal{J}^*(x)$	z^*	CPU(s)
10	500	2.379	2.379	0.3	1.861	1.861	0.1	2.018	2.018	0.2
20	5000	2.361	2.364	11.3	1.893	1.899	8.2	2.009	2.013	6.0
40	5×10^4	2.290	2.291	4204.1	1.667	1.669	4251.4	1.996	1.998	3898.0

Table 4.9: (Example 2): Values of the cost function $\mathcal{J}$ obtained by the *SOP* Algorithm from different initial states.

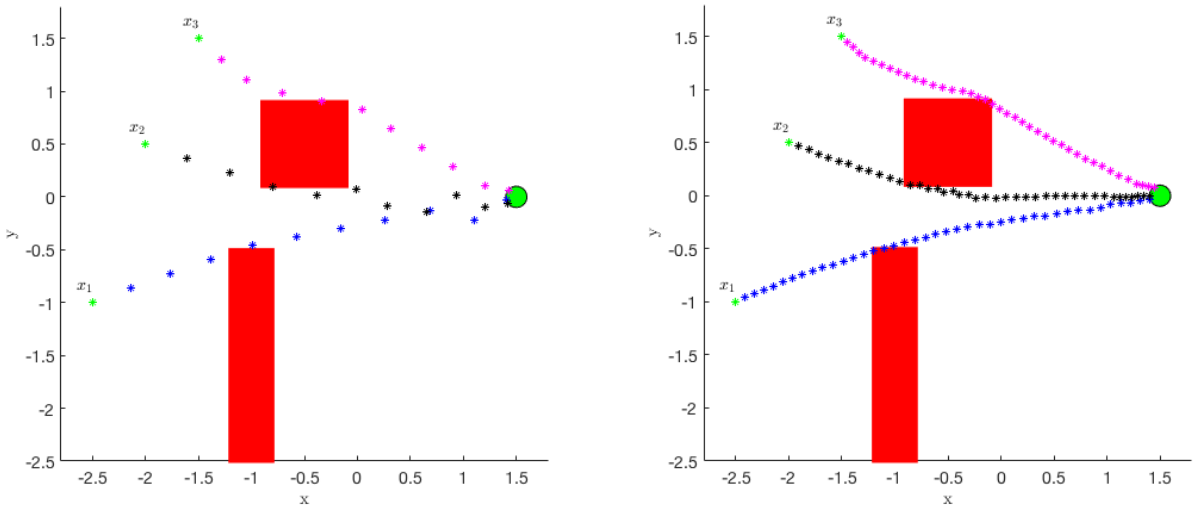

Figure 4.3: (Example 2): Trajectories obtained by dichotomy with *SOP* Algorithm to reach the target (in green) and to avoid the obstacles (in red), from different initial positions and with $N = 10$, $I_{\max} = 500$ (left) and $N = 40$, $I_{\max} = 5 \times 10^4$ (right).

Example 3: Optimal control of the heat equation

In this example, taken from [3], we want to illustrate the performances of our approach for solving a control problem of a partial differential equation. The discrete formulation of the problem leads to a control problem in a high dimensional state space. Consider the following heat equation:

$$\begin{cases} \frac{\partial y}{\partial t}(s,x) = \sigma \frac{\partial^2 y}{\partial x^2}(s,x) + y_0(x)a(s), \quad \text{for} \quad (s,x) \in [0,T] \times [0,1], \\ y(s,x) = 0, \quad \text{for} \quad (s,x) \in [0,T] \times \{0,1\}, \\ y(0,x) = y_0(x), \quad \text{for} \quad x \in [0,1], \end{cases} \tag{4.30}$$

where $\sigma = 0.1$, the control $a(\cdot)$ takes values in $A = [-1,1]$, $T = 1$ and $y_0(x) = -x^2 + x$, for $x \in [0,1]$. Our purpose is to minimize, by using the control input $a(\cdot)$, the temperature $y^a(t,x)$, solution of (4.30), for $t \in [0,T]$ and $x \in [0,1]$. For this reason, we consider the following cost functional of type Bolza:

$$Q(y_0, a) = \int_0^T \Big(\int_0^1 (y^a(s,x))^2 dx + \gamma a^2(s) \Big) ds + \int_0^1 (y^a(T,x))^2 dx, \tag{4.31}$$

where $\gamma > 0$. In order to approximate the solution of (4.30), we will use the implicit scheme. Indeed, consider a time grid with $N = 20$ time steps, $t_k = k\Delta t$ for $k = 0, ..., N$, where $\Delta t = \frac{T}{N}$ and a space grid with $d = 10^3$ points on $]0,1[$, $x_j = j\Delta x$ for $j = 1, ..., d$ and where the space step is given by $\Delta x = \frac{1}{d+1}$.

Hence, the implicit scheme approximating (4.30) is given by:

$$\begin{cases} \frac{y_{k+1}^j - y_k^j}{\Delta t} = \sigma \frac{y_{k+1}^{j+1} - 2y_{k+1}^j + y_{k+1}^{j-1}}{\Delta x^2} + y_0(x_j)a_k, \quad 1 \leq j \leq d, \\ y_k^0 = y_k^{d+1} = 0, \end{cases} \tag{4.32}$$

for $0 \leq k \leq N-1$ and where y_k^j is an approximation of $y^a(t_k, x_j)$ and a_k is the control value at time t_k. The implicit scheme (4.32) can be rewritten as follows:

$$Y_{k+1} = F(Y_k, a_k) := \Big(I_d + \frac{\sigma \Delta t}{\Delta x^2} P\Big)^{-1} \Big(Y_k + \Delta t \times a_k \times Y_0\Big), \tag{4.33}$$

where $Y_k := (y_k^j)_{1 \leq j \leq d} \in \mathbb{R}^d$ represents the system state and P is a $d \times d$ matrix given explicitly by:

$$P = \begin{pmatrix} 2 & -1 & \cdots & 0 \\ -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ 0 & \cdots & -1 & 2 \end{pmatrix}.$$

Notice that (4.33) is meaningful since the matrix $\Big(I_d + \frac{\sigma \Delta t}{\Delta x^2} P\Big)$ is invertible. In order to solve (4.33), we will use a Cholesky decomposition for tridiagonal matrix since $(I_d + \frac{\sigma \Delta t}{\Delta x^2} P)$ can be written as follow:

$$I_d + \frac{\sigma \Delta t}{\Delta x^2} P = L \times {}^t L, \quad \text{where} \quad L := \begin{pmatrix} \alpha_1 & 0 & \cdots & 0 \\ \beta 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & \beta_{d-1} & \alpha_d \end{pmatrix}$$

with

$$\begin{cases} \alpha_1^2 = \beta_1^2 + \alpha_2^2 = \cdots = \beta_{d-1}^2 + \alpha_d^2 = 1 + 2\frac{\sigma \Delta t}{\Delta x^2} \\ \alpha_1 \beta_1 = \alpha_2 \beta_2 = \cdots = \alpha_{d-1}\beta_{d-1} = -\frac{\sigma \Delta t}{\Delta x^2}. \end{cases}$$

On the other hand, we have $\|(I_d + \frac{\sigma \Delta t}{\Delta x^2} P)^{-1}\|_2 \leq 1$ hence the Lipschitz constants of the dynamics F can be chosen as follow:

$$L_{F,x} = 1 \quad \text{and} \quad L_{F,a} = \Delta t \|Y_0\|_2.$$

Furthermore, the cost functional Q, defined in (4.31), can be approximated by:

$$\mathcal{J}(Y_0, a) = \sum_{k=0}^{N-1} \rho(Y_k, a_k) + \Phi(Y_N),$$

where the instantaneous cost ρ and the terminal cost Φ are given by:

$$\rho(Y, a) = \frac{\Delta t}{2}\Big(\|Y\|_2^2 + \|F(Y, a)\|_2^2 + 2\gamma a^2\Big) \quad \text{and} \quad \Phi(Y) = \|Y\|_2^2,$$

where $Y \in \mathbb{R}^d$ and $a \in A$. Moreover, the Lipschitz constants associated to the ρ and Φ are given by:

$$L_{\rho,x} = \Delta t \|Y_0\|_2 \Big(1 + L_{F,x}\Big) \quad \text{and} \quad L_{\rho,a} = \Delta t \Big(\|Y_0\| L_{F,a} + 2\gamma\Big).$$

The uncontrolled solution, presented in the top left panel of figure 4.4, corresponds to a numerical solution of (4.30) while taking $a(\cdot) \equiv 0$. As expected, the controlled solutions, obtained by *OP*, *SOP* and

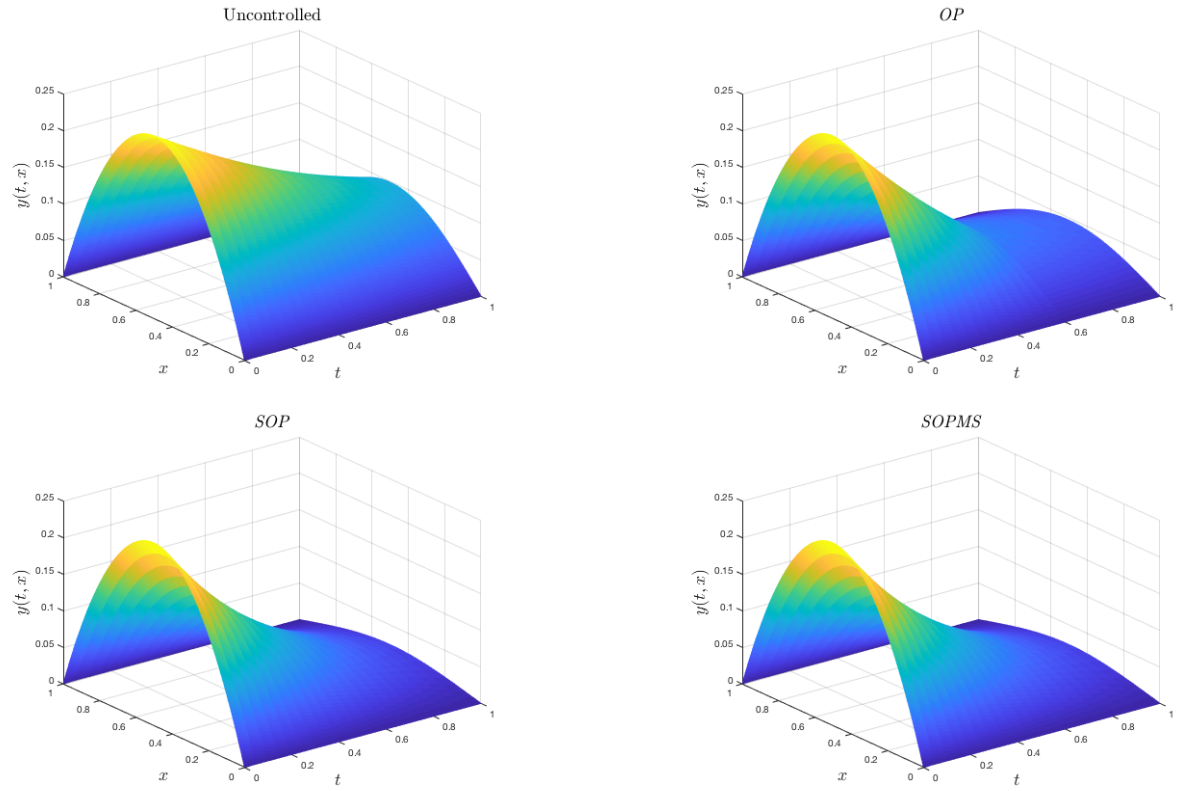

Figure 4.4: (Example 3): Uncontrolled solution (top left) and controlled solutions with *OP* (top right), *SOP* (bottom left) and *SOPMS* (bottom right), for $I_{\max} = 10^4$, $\gamma = 0.01$ and $d = 10^3$.

Algorithm	*OP*	*SOP*	*SOPMS*
CPU(s)	238.43	101.12	145.37

Table 4.10: CPU time required to compute the controlled solutions by the different optimistic planning algorithms

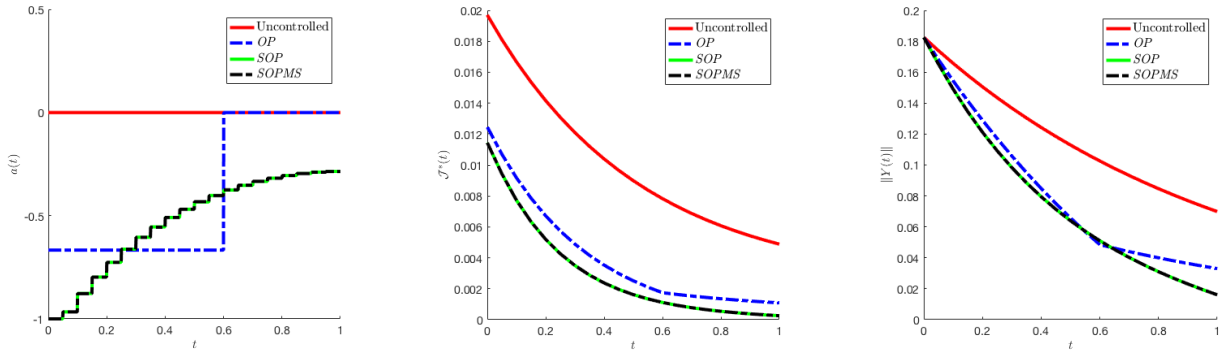

Figure 4.5: (Example 3): Controls computed by *OP*, *SOP* and *SOPMS* algorithms (left), time comparison of cost functions (middle), norms of the solutions (right), for $I_{\max} = 10^4$, $\gamma = 0.01$ and $d = 10^3$.

SOPMS algorithms are below the uncontrolled solution. Furthermore, the solutions obtained by the *SOP* and *SOPMS* algorithms are similar and are better than the solution computed with the *OP* Algorithm (see figure 4.4). This observation can be confirmed by a comparison between the solutions norms and the cost

values computed with different planning algorithms (see figure 4.5).

On the other hand, in figure 4.6, we remark that the controlled solution corresponding to $\gamma = 10^{-4}$ or $\gamma = 10^{-6}$ is below the controlled solution for $\gamma = 10^{-2}$. This is can be explained by the control difference between the two cases. Indeed, when γ is equal to 10^{-4} or 10^{-6}, we allow values of the control with larger norms. In addition to that, due to its important weight in the distributed cost function, the control corresponding to $\gamma = 10^{-2}$ is more regular than the control simulated with $\gamma = 10^{-4}$ or $\gamma = 10^{-6}$.

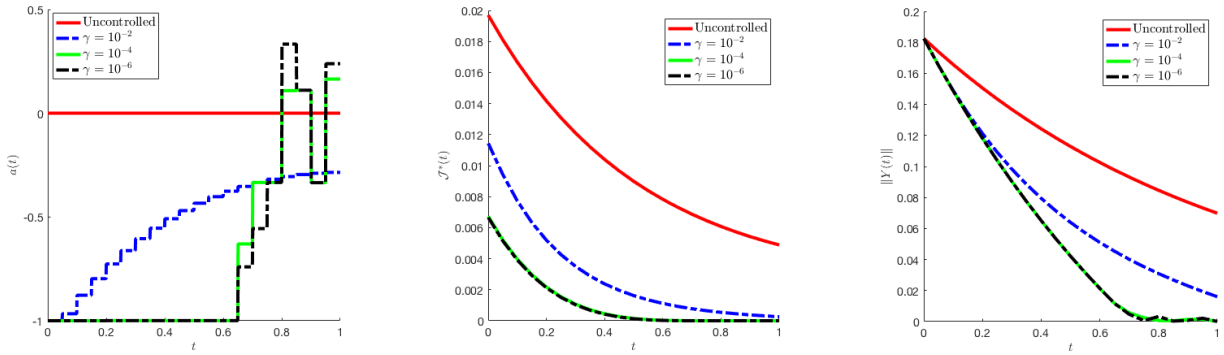

Figure 4.6: (Example 3): Controls (left), time comparison of the cost function (middle) and norms of the solutions (right) computed by the *SOP* algorithm, for different values of the parameter γ, $I_{\max} = 10^4$ and $d = 10^3$.

Finally, we add constraints on the solution of the heat equation $y^a(\cdot,\cdot)$, corresponding to some control $a(\cdot)$, of the form $y^a(t,x) \geq \theta y_0(x)$ for any $t \in [0,T]$ and $x \in [0,1]$ and where $\theta \in]0,1[$. To this end, we consider the following obstacle function g given by:

$$g(y^a(t,\cdot)) := \max_{x\in[0,1]} \Big\{\theta y_0(x) - y^a(t,x)\Big\}.$$

The bounds on the auxiliary variable z are taken as follow:

$$Z_{\min} = 0 \quad \text{and} \quad Z_{\max} = 2\int_0^T y_0^2(x)dx + \gamma.$$

In figure 4.7, we represent the uncontrolled solution $(a(\cdot) \equiv 0)$, the unconstrained solution obtained with *SOP* and the constrained solution obtained by dichotomy with *SOP* Algorithm. We observe that the constrained solutions for both cases, $\theta = \frac{1}{3}$ and $\theta = \frac{2}{3}$, verify the constraints for any $t \in [0,T]$ and $x \in [0,1]$.

Example 4: Windshear problem

Consider the abort landing problem studied in chapter 3 as a game where the wind disturbances are unknown and modelled as a second player. This problem was also studied in [6] by the HJ approach. Here, we propose to solve it by use of optimistic planning algorithms.

Consider the flight of an aircraft in a vertical plane over a flat earth. We assume that all the forces act on the center of gravity G of the aircraft and lie in the same plane of symmetry. From the Newton's law, the

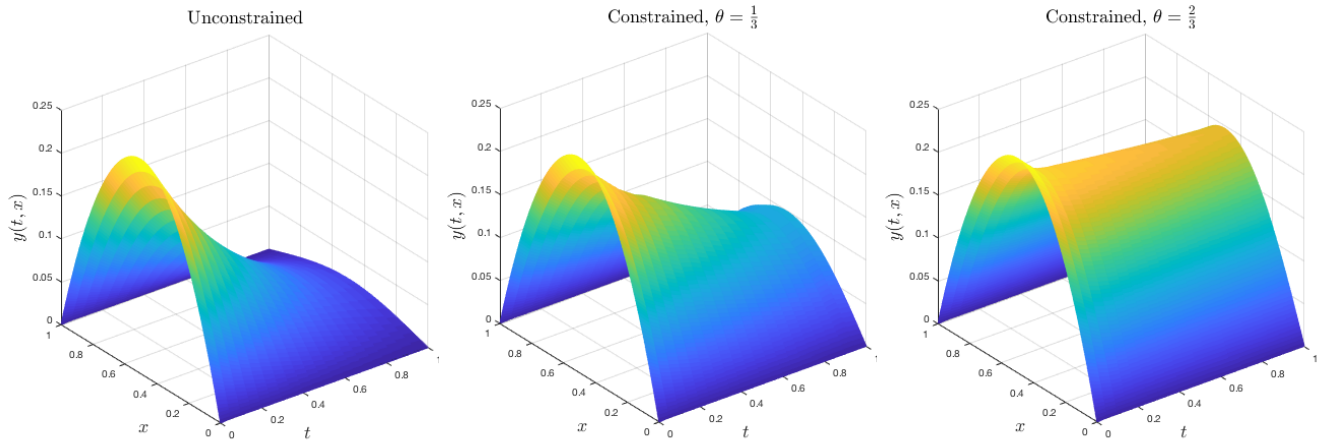

Figure 4.7: (Example 3): Unconstrained solution with *SOP* (left), constrained solutions for $\theta = \frac{1}{3}$ (middle) and $\theta = \frac{2}{3}$ (right) obtained by dichotomy with *SOP* Algorithm for $d = 10^3$, $\gamma = 0.01$ and $I_{\max} = 10^4$.

aircraft's motion is described by (see [6, 33] for more details):

$$\begin{cases} \dot{x}(s) &= u(s)\cos(\gamma(s)) + \omega_x(x(s)) \\ \dot{h}(s) &= u(s)\sin(\gamma(s)) + \omega_h(x(s), h(s)) \\ \dot{u}(s) &= \frac{F_T(u(s))}{m}\cos(\theta(s)+\delta) - \frac{F_D(u(s),\theta(s))}{m} - g\sin(\gamma(s)) - \dot{\omega}_x(x(s))\cos(\gamma(s)) \\ & \quad -\dot{\omega}_h(x(s), h(s))\sin(\gamma(s)) \\ \dot{\gamma}(s) &= \frac{1}{u(s)}\Big(\frac{\beta F_T(u(s))}{m}\sin(\theta(s)+\delta) + \frac{F_L(u(s),\theta(s))}{m} - g\cos(\gamma(s)) + \dot{\omega}_x(x(s))\sin(\gamma(s)) \\ & \quad -\dot{\omega}_h(x(s), h(s))\cos(\gamma(s))\Big) \\ \dot{\theta}(s) &= a(s), \end{cases} \tag{4.34}$$

where x is the horizontal distance, h denotes the altitude, u is the aircraft velocity, γ is the relative path inclination, θ is the angle of attack, $\delta > 0$ is a parameter of the model, a represents the control variable. Moreover, ω_x and ω_h are respectively the horizontal and the vertical components of the wind velocity vector, $\dot{\omega}_x$ and $\dot{\omega}_h$ are their derivatives, F_T, F_L and F_D denote respectively the thrust, lift and drag forces whose expressions can be found in [6, 33, 34] and are given in the Appendix 4.7.2.

We represent the state variables by a vector $y \in \mathbb{R}^5$ given by $y := (x, h, u, \gamma, \theta)^\top$, hence the dynamical system (4.34) can be written as $\dot{y}(s) = f(y(s), a(s))$.

The sets of control, A, and of state constraints, $\mathcal{K}$, are of the form:

$$A := [a_{\min}, a_{\max}] \quad \text{and} \quad \mathcal{K} := \mathbb{R}^4 \times [\theta_{\min}, \theta_{\max}],$$

with $a_{\min} = -a_{\max} = -3$ deg s^{-1}, $\theta_{\min} = -180$ deg and $\theta_{\max} = 17.2$ deg.

In order to determinate the Lipschitz constant $L_{f,x}$, let y and y' be two vectors of $\mathbb{R}^5$. For any $i = 1, ..., 5$, there exist non-negative coefficients $(l_{i,j})_{j=1}^5$ such that:

$$|f_i(y, a) - f_i(y', a)| \leq \sum_{j=1}^{5} l_{i,j}|y_j - y'_j|, \quad \text{for any} \quad a \in A.$$

Henceforth, $L_{f,x} = \|P\|_2$ where the matrix P is given by $P := (l_{i,j})_{1\leq i,j\leq 5}$. Moreover, the Lipschitz constant $L_{f,a}$ is equal to 1.

Recall that the aim is to steer the aircraft to the maximum altitude that can be reached during an interval of time. The maximum running cost function Φ is defined as $\Phi(y) := h_* - h$, where h is the aircraft altitude and $h_* > 0$ is a given reference altitude.

The value function of this problem, for $T = 40$ and starting from an initial position $y \in \mathbb{R}^5$, is defined by:

$$v(0,y) = \inf_{a(\cdot)} \Big\{ \max_{s\in[0,T]} \Phi(y^a_{0,y}(s)) \;\mid\; y^a_{0,y}(s) \in \mathcal{K}, \quad \text{for all} \quad s \in [0,T] \Big\}.$$

In order to solve this problem, we discretize uniformly $[0,T]$ with N sub-intervals and the auxiliary problem to be solved is of the form:

$$W(y,z) = \inf_{(a_k)\in A^N} \max_{0\le k\le N} \Big\{ \Big(\Phi(y^a_k) - z\Big) \bigvee g(y^a_k) \Big\},$$

where $(y^a_k)_k$ is the discrete trajectory corresponding to the control sequence $(a_k)_k \in A^N$, starting from the initial state y and the obstacle function g is given by:

$$g(y) = \max(\theta_{\min} - \theta, \theta - \theta_{\max}).$$

We already know that an approximation of the constrained problem value is given by:

$$z^* = \inf \Big\{ z \in [0, h_*] \quad \mid \quad W(y,z) \le 0 \Big\}.$$

On the other hand, the value of the criterion, associated to the trajectory sequence $(y^*_k)_k$ returned by optimistic planning algorithms, is defined by:

$$\mathcal{J}^*(y) := \max_{0\le k\le N} \Phi(y^*_k).$$

We consider the following initial configurations:

$$y_0 = (0, 600, 239.7, -2.249 \text{ deg}, 7.373 \text{ deg}) \quad \text{and} \quad y_1 = (0, 650, 239.7, -3.400 \text{ deg}, 7.373 \text{ deg}).$$

Initial configurations	Parameters N	Parameters $I_{\max}$	$\mathcal{J}^*$ (ft)	z^* (ft)	CPU (s)
y_0	10	10^3	544.116	544.189	1.79
	20	10^4	520.624	524.185	31.7
	40	10^5	480.713	480.746	2911.5
y_1	10	10^3	521.720	521.729	1.85
	20	10^4	487.649	487.661	30.3
	40	10^5	479.735	479.739	2793.9

Table 4.11: (Example 4): Performance of the dichotomy with *SOP* Algorithm, for $T = 40$, different values of N and $I_{\max}$ and from different initial configurations.

We observe in tables 4.11 and 4.12 that the performances of trajectories, from y_0 and y_1, are ameliorated when N doubles (decrease of the cost functional $\mathcal{J}^*$ and hence increase of the lowest altitude). Moreover, we remark that the value of the approximation z^* is very close to $\mathcal{J}^*$.

On the other hand, by comparing tables 4.11 and 4.12, we observe that *SOP* performances are better than *SOPMS* performances for the initial state y_0 for a value of I_{eval} not large enough. This is not the case for larger values of I_{eval}. This is due to the compromise in the choice of the parameter I_{eval} between the solution quality and the resolution complexity (see subsection 4.4.4).

Finally, the difference on the altitude evolution observed in figures 4.8 and 4.9 can be explained by the fact that the control actions applied by the *SOPMS* Algorithm are computed and updated while progressing in time in contrary to the *SOP* Algorithm where all the actions are first computed then applied to the dynamical system.

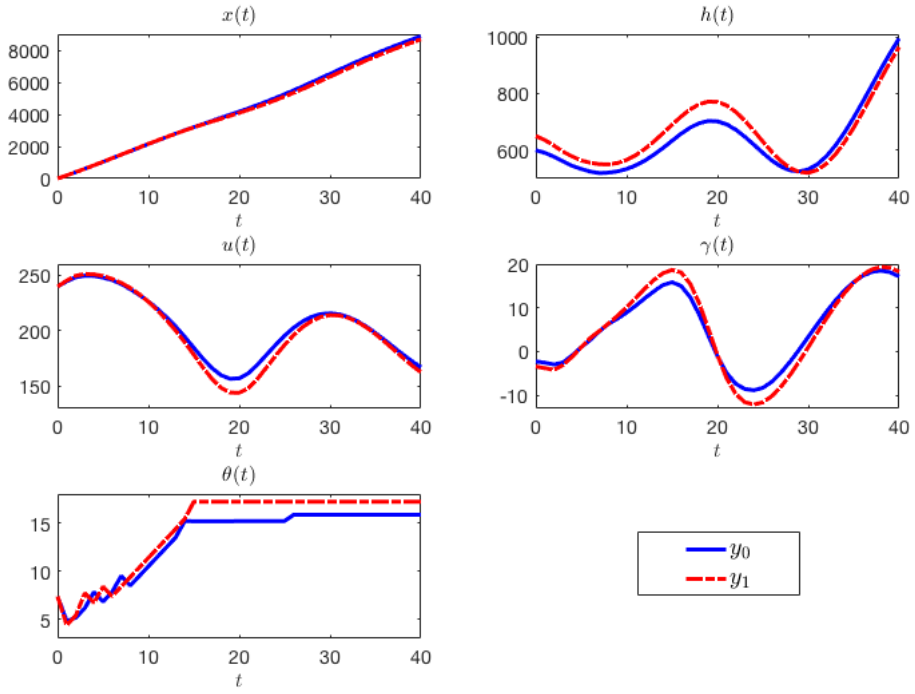

Figure 4.8: (Example 4): Trajectories obtained by the dichotomy with *SOP* Algorithm, for $T = 40$, $N = 40$, $I_{\max} = 10^5$ and from different initial configurations.

Initial configurations	Parameters N	I_{eval}	$\mathcal{J}^*$ (ft)	z^* (ft)	CPU (s)
y_0	10	30	519.498	520.047	1.2
	20	300	509.953	510.221	9.7
	40	3000	494.862	495.745	714.2
	40	6000	477.281	478.638	3252.9
y_1	10	30	517.106	517.986	1.4
	20	300	508.532	509.844	8.3
	40	3000	459.393	459.981	673.4
	40	6000	452.100	453.354	3017.1

Table 4.12: (Example 4): Performance of the dichotomy with *SOPMS* Algorithm, for $T = 40$, different values of N and I_{eval} and from different initial configurations.

4.7 Appendix

4.7.1 Appendix: Proofs of convergence results for *OP* and *SOP* algorithms

Proof of Theorem 4.4.5. Let $\mathbb{A}_i$ be a node of the tree Υ at some depth p_i. From Corollary 4.3.3, the error σ_i is bounded by:

$$\sigma_i \leq \delta_i := \frac{C}{2}\Delta t \sum_{k=0}^{N-1} (\frac{1}{L_{F,x}})^k d_{i,k},$$

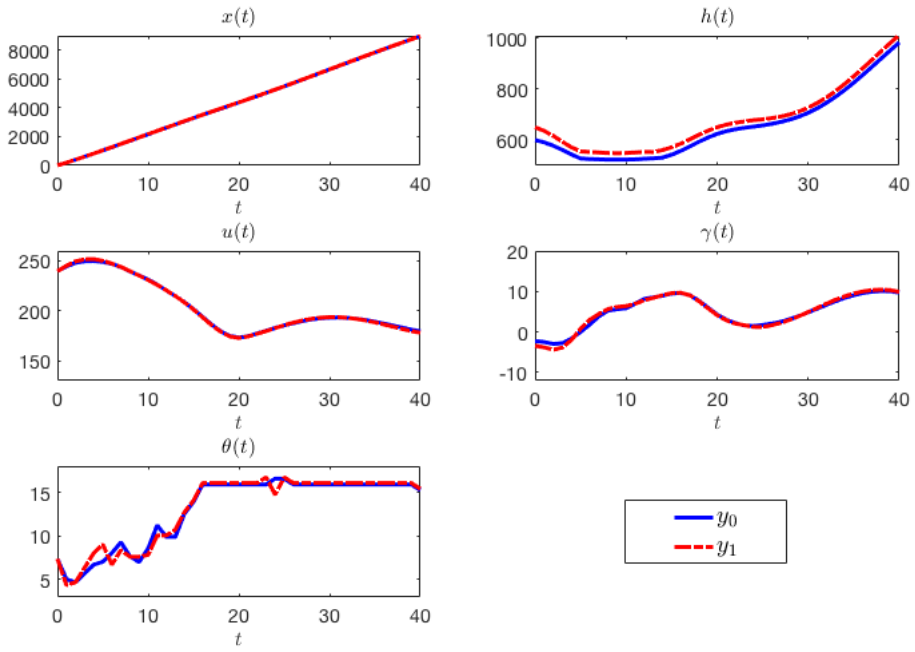

Figure 4.9: (Example 4): Trajectories obtained by the dichotomy with *SOPMS* Algorithm, for $T = 40$, $N = 40$, $I_{eval} = 6000$ and from different initial configurations.

where $C > 0$ is the same real constant as in Corollary 4.3.3. Consider a new criterion to choose the interval to split among the intervals $A_{i,k}$, for $k = 0, ..., N-1$, composing the node $\mathbb{A}_i$, relative to the new bound δ_i, given by:

$$k_i^\sharp = \underset{0 \le k \le N-1}{\text{argmax}} \, (\frac{1}{L_{F,x}})^k d_{i,k}. \tag{4.35}$$

Only in this proof, denote by $\gamma := \frac{1}{L_{F,x}} < 1$ and we will omit the node index i for all the derivation.

By using similar arguments as in [39], the split function $s(\cdot)$, indicating the number of splits per time step, is decreasing and decreases of at most 1 i.e.

$$s(k-1) - 1 \le s(k) \le s(k-1), \quad \text{for any} \quad k \in \{1, ..., N-1\}.$$

Indeed, let $1 \le k \le N-1$ and $s(k-1) = s(k) + 1$. The gap between $s(k-1)$ and $s(k)$ becomes equal to 2 if the interval of rank $k-1$ will be split before the one of rank k. In this case we should have $\gamma^{k-1} M^{-s(k-1)} = \gamma^{k-1} M^{-s(k)-1} \ge \gamma^k M^{-s(k)}$ from the selection principle (4.35) which implies that $M \le \frac{1}{\gamma}$ and this contradicts the assumption $M > L_{F,x}$.

Now consider τ_0, τ_1,...,τ_n the lengths of the ranges constant in s, for $n \in \mathbb{N}$ such that $n \le N+1$. The last range, of length τ_n, is where $s \equiv 0$ which can be empty *i.e.* $\tau_n = 0$ if all the intervals are split at least one time. Since we have supposed that the depth p is large enough, we can consider for the sequel $\tau_n = 0$.

The interval that will be split, if this node is selected at a future iteration, is the first interval of some range of index $0 \le e \le n-1$. Let k be the rank of this interval, $0 \le k \le N-1$. Again by using similar arguments as in [39], we obtain:

$$\tau_e \ge \frac{\log M}{\log(\frac{1}{\gamma})} \quad \text{and if} \quad e \ge 1, \tau_{e-1} \le \frac{\log M}{\log(\frac{1}{\gamma})}. \tag{4.36}$$

Indeed the first interval of the range $e-1$ is of rank $k_1 = k - \tau_{e-1}$ and the first one of the range $e+1$ is of rank $k_2 = k + \tau_e$. On the other hand, $s(k_1) = s(k) + 1$ and $s(k_2) = s(k) - 1$ since the gap between two

consecutive ranges is equal to 1. Now the interval of rank k is the first to be split at a future iteration which means that this interval is preferred to intervals of rank k_1 and k_2. From the selection procedure (4.35), we obtain $\gamma^k M^{-s(k)} \geq \gamma^{k_1} M^{-s(k_1)}$ and $\gamma^k M^{-s(k)} \geq \gamma^{k_2} M^{-s(k_2)}$. Replacing k_1, k_2, $s(k_1)$ and $s(k_2)$ by their expressions concludes the proof of (4.36). From (4.36) we deduce that $\tau_e \geq \tau$ and $\tau_{e-1} \leq \tau - 1$ if $e \geq 1$, where $\tau := \lceil \log M / \log(\frac{1}{\gamma}) \rceil$.

Now, we will prove that τ_0, τ_1,...,τ_{n-1} (recall that $\tau_n = 0$ for p large enough) verify:

$$\begin{cases} \tau_0 \leq \tau, \\ \tau_e \in \{\tau - 1, \tau\} \quad \text{for } \ 1 \leq e \leq n-1. \end{cases} \tag{4.37}$$

The assertion (4.37) can be also proven by the same arguments as in [39]. On the other hand, in our case, we have assumed that the number of intervals N verifies $N \geq 2\tau$. This assumption is necessary to guarantee that the first $p_0 := 3\tau + 2$ splits are done in the same order as given in figure 4.10 (see the small dashed squares with numbers indicating the order inside the squares). Indeed, one can consider any depth $p' \leq p_0$ and try to split an interval without respecting the order given in figure 4.10 and in this way (4.36) will be violated.

Now, suppose that (4.37) holds for an arbitrary node at depth $p \geq p_0$ and try to prove it for one of its descendant at depth $p+1$ (proof by induction). We denote by s' and $(\tau'_e)_e$ respectively the split function and the constant range lengths of the descendant node.

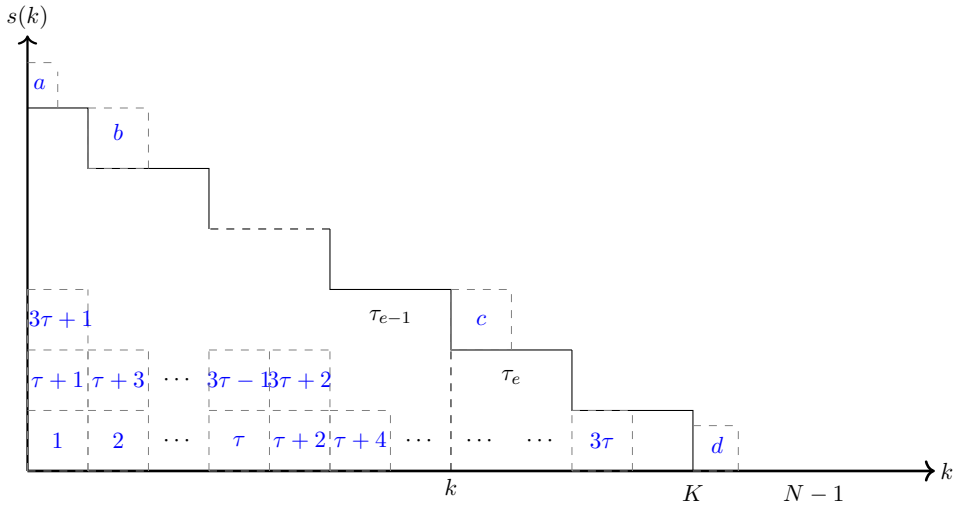

Figure 4.10: An example of a split function $s(\cdot)$ with 4 possibilities of the next interval split.

Four possible cases for the next interval to be split should be considered which are represented by dashed squares with letters a, b, c and d inside the squares in figure 4.10:

- Case a: The first interval A_0 (was denoted by $A_{i,0}$ before omitting the index i) is split. From (4.36) we get that $\tau_0 \geq \tau$ and from (4.37) we know that $\tau_0 \leq \tau$, hence $\tau_0 = \tau$. As for the descendent node, we get $\tau'_0 = 1$ and $\tau'_1 = \tau_0 - 1 = \tau - 1$. The other ranges are unchanged.
- Case b: The second interval A_1 is split. From (4.36) we have $\tau_0 \leq \tau - 1$ and $\tau_1 \geq \tau$ and from (4.37) we know that $\tau_1 \in \{\tau - 1, \tau\}$, hence $\tau_1 = \tau$. We deduce that for the descendant node, $\tau'_0 = \tau_0 + 1 \leq \tau$ and $\tau'_1 = \tau_1 + 1 = \tau$. The other ranges are unchanged.
- Case c: The first interval A_k of some arbitrary range τ_e is split, $1 < e < n$. From (4.36) we have $\tau_{e-1} \leq \tau - 1$ and $\tau_e \geq \tau$ and from (4.37) we know that $\tau_{e-1}, \tau_e \in \{\tau - 1, \tau\}$, hence $\tau_{e-1} = \tau - 1$

and $\tau_e = \tau$. This implies that $\tau'_{e-1} = \tau_{e-1} + 1 = \tau$ and $\tau'_e = \tau_e - 1 = \tau - 1$. The other ranges are unchanged.

- Case d: An interval of rank $K \leq N-1$ will be split for the first time (see figure 4.10). From (4.36), we have $\tau_{n-1} \leq \tau - 1$ and from (4.37), $\tau_{n-1} \in \{\tau - 1, \tau\}$, hence $\tau_{n-1} = \tau - 1$. We deduce that $\tau'_{n-1} = \tau_{n-1} + 1 = \tau$. The other ranges are unchanged.

From this point, our analysis departs slightly from [39]. The previous results enable us to bound $s(k)$ as follows:

$$s(k) \geq \underline{s}(k) := r - 1 - \frac{k - (N-1)}{\tau}, \tag{4.38}$$

and

$$s(k) \leq \overline{s}(k) := r + 1 - \frac{k - (N-1)}{\tau - 1}, \tag{4.39}$$

where $r = s(N-1)$ (see figure 4.11).

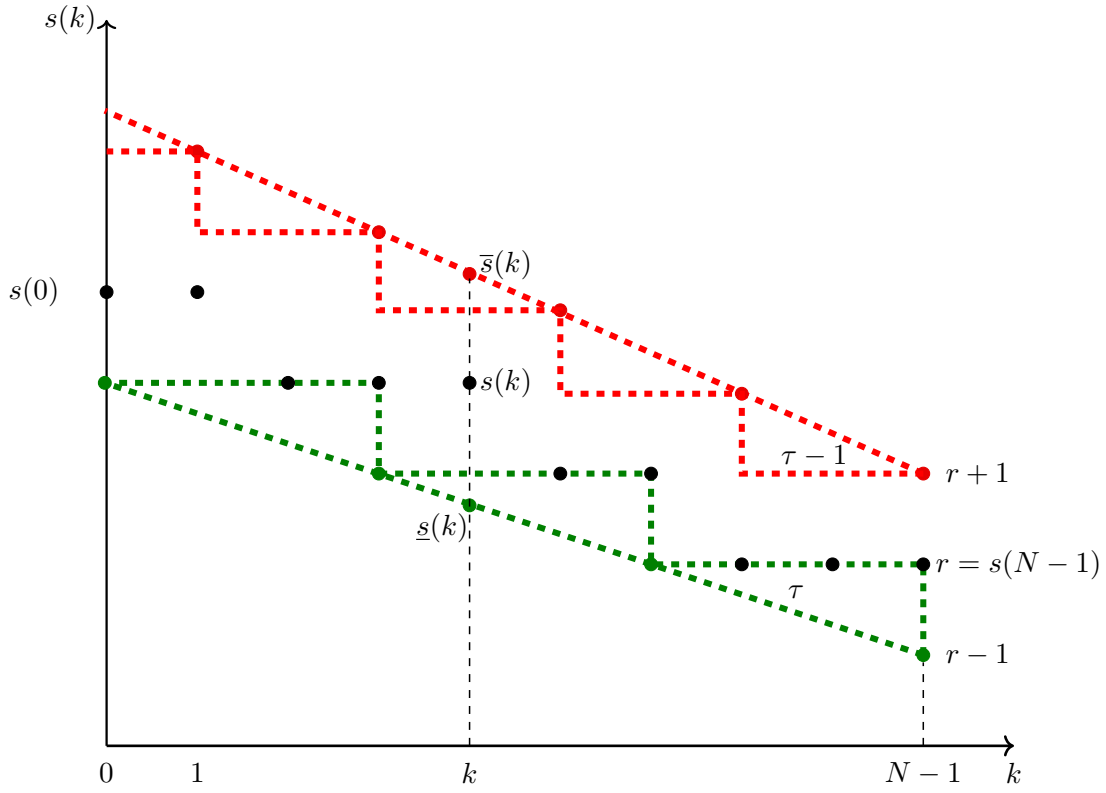

Figure 4.11: Example of split function $s(\cdot)$ (black dots) and the lower and upper bounds $\underline{s}(\cdot)$ (in green) and $\overline{s}(\cdot)$ (in red).

Recall that the diameter of the interval A_k can be obtained by $d_k = M^{-s(k)}$. Henceforth, from (4.38), we get:

$$\delta = \frac{C}{2}\Delta t \sum_{k=0}^{N-1} \gamma^k M^{-s(k)} \leq \frac{C}{2}\Delta t \sum_{k=0}^{N-1} \gamma^k M^{-\underline{s}(k)} \leq \frac{C}{2}\Delta t \frac{1}{1 - \gamma M^{\frac{1}{\tau}}} M^{-r+1-\frac{N-1}{\tau}}. \tag{4.40}$$

It remains to find a lower bound for r. By using (4.39), we have:

$$p = \sum_{k=0}^{N-1} s(k) \leq \sum_{k=0}^{N-1} \overline{s}(k) \leq N(r+1) + \frac{N(N-1)}{2(\tau - 1)}.$$

From the last inequality, we deduce that $r \geq \frac{p}{N} - 1 - \frac{N-1}{2(\tau-1)}$ together with (4.40), we conclude that:

$$\sigma \leq \delta \leq \frac{C}{2}\Delta t \frac{1}{1-\gamma M^{\frac{1}{\tau}}} M^{q(N)} M^{-\frac{p}{N}}$$

with $q(N) := 2 - (N-1)\frac{\tau-2}{2\tau(\tau-1)}$. □

Proof of Lemma 4.4.8. Denote by $\mathbb{A}_{i_n} \in \Lambda$ the node chosen at some iteration n to be expanded by the *OP* Algorithm. By construction of the tree, the set of leaves Λ, at any iteration of the algorithm, covers the entire space A^N. Therefore, there exists $\mathbb{A}_{i_n^*} \in \Lambda$ containing an optimal control sequence that verifies:

$$J(x, z, a_{i_n^*}) - \sigma_{i_n^*} \leq W(x, z) \leq J(x, z, a_{i_n^*}).$$

On the other hand, $\mathbb{A}_{i_n} = \underset{\mathbb{A}_i \in \Lambda}{\operatorname{argmin}} \Big\{ J(x, z, a_i) - \sigma_i \Big\}$, therefore

$$J(x, z, a_{i_n}) - \sigma_{i_n} \leq J(x, z, a_{i_n^*}) - \sigma_{i_n^*} \leq W(x, z).$$

Now, among the descendants of $\mathbb{A}_{i_n}$, there exists a leaf $\mathbb{A}_j$ such that $J(x, z, a_j) \leq J(x, z, a_{i_n})$ (see remark 4.4.3). Moreover, the returned node of the *OP* Algorithm verifies $\mathbb{A}_{i^*} = \underset{\mathbb{A}_i \in \Lambda}{\operatorname{argmin}} J(x, z, a_i)$. Therefore

$$J(x, z, a_{i^*}) \leq J(x, z, a_j) \leq J(x, z, a_{i_n}) \leq W(x, z) + \sigma_{i_n}.$$

As a conclusion, we get $J(x, z, a_{i^*}) - W(x, z) \leq \sigma_{i_n}$, for any expanded node $\mathbb{A}_{i_n}$, which gives the desired result. □

Proof of Theorem 4.4.9. The cost of expanding one node is one iteration. Recall that the *OP* Algorithm expands only nodes in $\bigcup_{p \geq 0} \Upsilon_p^*$ and that Υ_p^* contains at most Rm^p.

Suppose that $m > 1$ and let $I_{\max}$ be the maximal number of iterations. At any depth $p \geq 0$, the algorithm expands at most Rm^p nodes thus at most Rm^p iterations. Now let p be the smallest depth such that $1 + R\sum_{p'=0}^{p-1} m^{p'} \geq I_{\max}$. Hence at least one node at depth p was expanded such that

$$p \geq p^* := \frac{\log\left(\frac{(I_{\max}-1)(m-1)}{R}\right)}{\log m}.$$

Therefore the smallest error term among all the expanded nodes verifies $\sigma_{\min} \leq \sigma_p \leq \delta_p$, where δ_p is defined in (4.17), and from Lemma 4.4.8, we conclude that:

$$J^*(x, z) - W(x, z) \leq \sigma_{\min} \leq \delta_p \leq c_1(N)\Delta t M^{-\frac{p^*}{N}},$$

which gives the result for $m > 1$. The case $m = 1$ can be deduced by similar arguments. □

Proof of Lemma 4.4.10. We claim that for any depth $p \in \{0, ..., P_{\max}\}$, if the number of expanded nodes n verifies:

$$n \geq RP_{\max} \sum_{p'=0}^{p} m^{p'}, \tag{4.41}$$

then at least one node containing an optimal control sequence, at depth p, has been expanded.

Now, let $p(n)$ be the smallest depth verifying (4.18). Therefore, (4.41) is verified by $p = p(n) - 1$ which implies that, by using the above claim, the *SOP* Algorithm has expanded at least one optimal node at depth $p(n) - 1$ if $p(n) - 1 \leq P_{\max}$. However, if $p(n) - 1 > P_{\max}$ and since *SOP* Algorithm does not expand nodes with depth grater than $P_{\max}$, the deepest expanded optimal node is at depth $p = P_{\max}$, which

concludes the proof of Lemma 4.4.10.

Finally, let's prove the above claim by induction. For $p = 0$, if $n \geq RP_{\max} \geq 1$, the algorithm has expanded the tree root $\mathbb{A}_0 = A^N$ which contains an optimal control sequence.

Now suppose that the claim is true for $p \geq 0$ and let's prove it for $p+1$. Consider $n \in \mathbb{N}$ such that:

$$n \geq RP_{\max} \sum_{p'=0}^{p+1} m^{p'}.$$

and let $n' := n - RP_{\max} m^{p+1}$. By the induction hypothesis and since $n' \geq RP_{\max} \sum_{p'=0}^{p} m^{p'}$, we deduce that the algorithm has expanded an optimal node, $\mathbb{A}_{i_p^*}$, at depth p, after expanding n' nodes. The optimal node $\mathbb{A}_{i_p^*}$ will generate another optimal node at depth $p+1$ denoted by $\mathbb{A}_{i_{p+1}^*}$.

Let $\mathbb{A}_{i'_{p+1}}$ be a node that will be expanded by the *SOP* Algorithm, at depth $p+1$, before expanding $\mathbb{A}_{i_{p+1}^*}$. This node verifies certainly $J(x, z, a_{i'_{p+1}}) \leq J(x, z, a_{i_{p+1}^*})$. Moreover, since $\mathbb{A}_{i_{p+1}^*}$ is optimal and $\sigma_{i_{p+1}^*} \leq \delta_{p+1}$, see Theorem 4.4.5, we get:

$$J(x, z, a_{i'_{p+1}}) - \delta_{p+1} \leq J(x, z, a_{i_{p+1}^*}) - \delta_{p+1} \leq J(x, z, a_{i_{p+1}^*}) - \sigma_{i_{p+1}^*} \leq W(x, z),$$

which means that the node $\mathbb{A}_{i'_{p+1}}$ belongs to Υ_{p+1}^*. In conclusion, any node $\mathbb{A}_{i'_{p+1}}$ that will be expanded before expanding $\mathbb{A}_{i_{p+1}^*}$ belongs certainly to Υ_{p+1}^*.

Finally, recall that $|\Upsilon_{p+1}^*| \leq Rm^{p+1}$ and that the difference between the maximal tree depth and the smallest depth with unexpanded nodes, at any iteration of the *SOP* Algorithm, is smaller than $P_{\max}$. Henceforth, the *SOP* Algorithm is sure to expand $\mathbb{A}_{i_{p+1}^*}$ after expanding at most $P_{\max} R m^{p+1}$ nodes. As a conclusion, after expanding at most n nodes $(n = n' + P_{\max} R m^{p+1})$, the *SOP* Algorithm will expand $\mathbb{A}_{i_{p+1}^*}$. □

Proof of Theorem 4.4.11. In the case where $m > 1$ and $P_{\max} = I_{\max}^{\eta}$, for $\eta \in]0, 1[$, and after expanding $I_{\max}$ nodes, let $p(I_{\max})$ be defined as in Lemma 4.4.10. Therefore, $p(I_{\max}) - 1$ verifies:

$$I_{\max} \geq RP_{\max} \sum_{p'=0}^{p(I_{\max})-1} m^{p'},$$

which implies that:

$$p(I_{\max}) \leq b_1 \log(I_{\max}^{1-\eta}) + b_2,$$

for some real constants $b_1, b_2 > 0$. We deduce from the last inequality that for $I_{\max}$ large enough, $p(I_{\max}) << P_{\max}$ since $P_{\max} = I_{\max}^{\eta}$. Henceforth, the depth of the deepest expanded optimal node p^*, defined in Lemma 4.4.10, is equal to $p(I_{\max}) - 1$. From (4.18) and by taking $n = I_{\max}$ and $P_{\max} = I_{\max}^{\eta}$, we get:

$$p^* \geq \frac{\log(I_{\max}^{1-\eta})}{\log m} - \frac{\log(\frac{R}{m-1})}{\log m} - 2.$$

Since $J^*(x, z) - W(x, z) \leq \delta_{p^*} = c_1(N) \Delta t M^{-\frac{p^*}{N}}$, using the lower bound on p^* found in the above inequality gives the desired result for $m > 1$.

Now suppose that $m = 1$ and let $P_{\max} = \sqrt{I_{\max}}$. After expanding $I_{\max}$ nodes, from (4.41) and since $R \geq 1$, we get:

$$p(I_{\max}) < \frac{\sqrt{I_{\max}}}{R} \leq \sqrt{I_{\max}} = P_{\max}.$$

Therefore, $p^* = p(I_{\max}) - 1$. Again from (4.18) and by taking $n = I_{\max}$ and $P_{\max} = \sqrt{I_{\max}}$, we get $\frac{\sqrt{I_{\max}}}{R} - 1 \leq p(I_{\max})$ and hence $p^* \geq \frac{\sqrt{I_{\max}}}{R} - 2$. As a conclusion:

$$J^*(x,z) - W(x,z) \leq \delta_{p^*} = c_1(N)\Delta t M^{-\frac{p^*}{N}} \leq c_1(N)\Delta t M^2 M^{-\frac{\sqrt{I_{\max}}}{RN}}.$$

□

4.7.2 Appendix B. Numerical parameters of example 4

The model of the wind disturbances (ω_x, ω_h), considered here, is represented in figure 4.12.

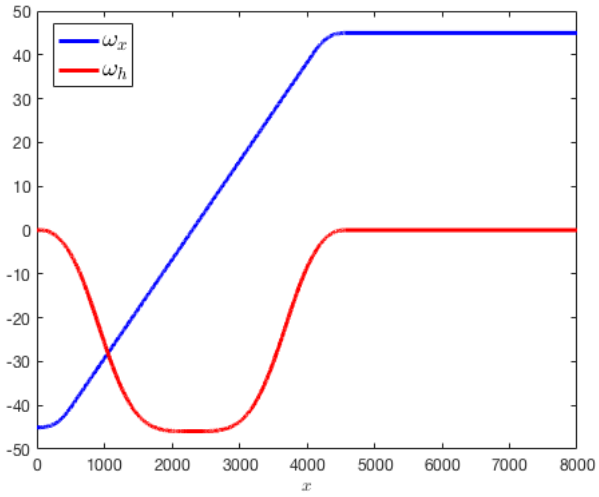

Figure 4.12: Horizontal and vertical wind velocity components, $\omega_x(x)$ and $\omega_h(x,h)$, as functions of x for fixed $h = 1000$ ft.

The mathematical expressions of the wind velocity components are given by:

$$\omega_x(x) = kC(x) \quad \text{and} \quad \omega_h(x,h) = k\frac{h}{h_*}D(x),$$

where $C(\cdot)$ and $D(\cdot)$ depend only on the horizontal position x as follow:

$$C(x) = \begin{cases} -50 + ax^3 + bx^4, & 0 \leq x \leq 500 \\ (x - 2300)/40, & 500 \leq x \leq 4100 \\ 50 - a(4600 - x)^3 - b(4600 - x)^4, & 4100 \leq x \leq 4600 \\ 50, & 4600 \leq x, \end{cases}$$

$$D(x) = \begin{cases} dx^3 + ex^4, & 0 \leq x \leq 500, \\ -51\ \exp\Big(-c(x - 2300)^4\Big), & 500 \leq x \leq 4100 \\ d(4600 - x)^3 + e(4600 - x)^4, & 4100 \leq x \leq 4600 \\ 0, & 4600 \leq x. \end{cases}$$

The derivatives of the horizontal and the vertical components of the wind velocity vector, $\dot{\omega}_x$ and $\dot{\omega}_h$, are defined as:

$$\dot{\omega}_x = \frac{\partial \omega_x}{\partial x}(u\cos(\gamma) + \omega_x) + \frac{\partial \omega_x}{\partial h}(u\sin(\gamma) + \omega_h),$$
$$\dot{\omega}_h = \frac{\partial \omega_h}{\partial x}(u\cos(\gamma) + \omega_x) + \frac{\partial \omega_h}{\partial h}(u\sin(\gamma) + \omega_h).$$

On the other hand, the modulus of the thrust, lift and drag forces are given by the following expressions:

$$F_T(u) := A_0 + A_1 u + A_2 u^2, \quad F_L(u,\theta) = \frac{1}{2}\rho S u^2 c_\ell(\theta), \quad F_D(u,\theta) = \frac{1}{2}\rho S u^2 c_d(\theta),$$

where c_ℓ and c_ℓ are polynomials of the variable θ given by:

$$c_d(\theta) = B_0 + B_1\theta + B_2\theta^2,$$

$$c_\ell(\theta) = \begin{cases} C_0 + C_1\theta, & \theta \leq \theta_*, \\ C_0 + C_1\theta + C_2\theta^2, & \theta \geq \theta_*. \end{cases}$$

The different parameters of the aircraft model and the wind velocities are presented in the following table:

Parameter	Value	Unit
h_*	1000	ft
δ	3.49 e-02	rad
k	0.9	
θ_*	0.2094	rad
S	1560.0	ft^2
ρ	2.203 10^{-3}	lb s^2 ft^{-4}
a	6 $\times 10^{-6}$	s^{-1} ft^{-2}
b	- 4 $\times 10^{-11}$	s^{-1} ft^{-3}
c	- $\log(\frac{25}{30.6}) \times 10^{-12}$	ft^{-4}
d	-8.02881 $\times 10^{-8}$	s^{-1} ft^{-2}

Parameter	Value	Unit
e	6.28083 $\times 10^{-11}$	s^{-1} ft^{-3}
A_0	4.456 $\times 10^4$	lb
A_1	-23.98	lb s ft^{-1}
A_2	1.42 $\times 10^{-2}$	lb s^2 ft^{-2}
B_0	0.1552	
B_1	0.1237	rad^{-1}
B_2	2.4203	rad^{-2}
C_0	0.7125	
C_1	6.0877	rad^{-1}
C_2	-9.0277	rad^{-2}

Table 4.13: (Example 4): Numerical data of the model.

Chapter 5

Deep Learning Numerical Methods For Dynamic Programming

5.1 Introduction

This chapter is devoted to study numerical methods for deterministic optimal control problems based on deep learning. We propose two approaches in order to deal with state-constrained problems. The first one is based on the dynamic programming principle while the second method tries to approximate the solutions of HJ equations. Both approaches can be extended to handle two-person zero-sum differential games under constraints on the system state.

Thanks to advances in deep learning and data analysis, Deep Neural Networks (DNN) have been exploited in diverse scientific fields such as genomics [2], natural language processing [93], image recognition [90] and cognitive sciences [96]. DNN have shown to be relevant in approximating a large class of complex non linear functions in finite dimensional space. This relevance can be theoretically justified by the Kolmogorov-Arnold representation theorem and the universal approximation theorem, see [74, 98, 52, 72, 99].

It is known that the value function of an optimal control problem, under suitable assumptions, is the solution of a dynamic programming equation. Nevertheless, the dynamic programming approach suffers from the curse of dimensionality since the value function should be projected on a grid of the state space. One alternative solution is to discretize in time and then try to approximate the discrete time value function, at each time step, by neural networks after its learning on a training grid with reduced size [21]. For instance, in [78, 77], deep learning algorithms are proposed in order to solve high-dimensional stochastic control problems. We focus on the *Hyprid-Now* Algorithm for which the optimal policy is first estimated by neural networks and dynamic programming. Then, this estimated policy is injected in a backward process in the aim of approximating the discrete value function by neural networks. This approach is very interesting especially when the optimal policy is regular. In this chapter, we propose to adapt this algorithm to deterministic control problems for which the optimal control is not always regular enough. To this end, we will try to approximate only the value function by using neural networks and by exploiting the dynamic programming principle. Moreover, we extend this approach to deal with constraints on the system state.

On the other hand, DNN have been successfully used to solve some nonlinear partial differential equations (PDE) derived from physics and mathematics, see [115, 114, 113, 112, 71, 128]. Indeed, the solution of the PDE can be directly approximated by neural networks that will be learned, on a reduced training domain, in order to satisfy the boundary conditions and the given equation law. In this context, one can hope to use neural networks in order to solve Hamilton–Jacobi equations derived from non linear optimal control problems [47, 107, 123]. Moreover, it has been shown in [53] that some neural network architectures, under certain conditions, can be shown to be viscosity solutions to some particular HJ equations

with hamiltonians and initial data defined from the neural networks parameters. Another approach, consisting in estimating the solution and its gradient by neural networks, was introduced and discussed in [79, 80]. In this chapter, we propose to approximate the value function, solution of some HJ equation, by use of spatio-temporal function approximators (neural networks) while computing its derivatives by means of automatic differentiation, see [17].

Consider an optimal control problem with finite time horizon $T > 0$ and state constraints:

$$v(t,x) := \inf_{a(\cdot)\in\mathcal{A}} \Big\{ \int_t^T \ell(y_{t,x}^a(s), a(s))ds + \Phi(y_{t,x}^a(T)) \mid y_{t,x}^a(s) \in \mathcal{K}, \ \ \forall s \in [t,T] \Big\}, \tag{5.1}$$

where $\mathcal{A}$ is the set of controls taking values in a compact set $A \subset \mathbb{R}^q$, with $q \geq 1$, $\mathcal{K}$ is a closed subset of $\mathbb{R}^d$ representing the state constraints set and $y_{t,x}^a(\cdot)$, representing the system trajectory, is the continuous solution of the following dynamical system

$$\begin{cases} \dot{y}(s) = f(y(s), a(s)) \quad a.e. \quad s \in [t,T], \\ y(t) = x \in \mathbb{R}^d. \end{cases} \tag{5.2}$$

The functions $f : \mathbb{R}^d \times \mathbb{R}^q \to \mathbb{R}^d$, $\ell : \mathbb{R}^d \times \mathbb{R}^q \to \mathbb{R}$ and $\Phi : \mathbb{R}^d \to \mathbb{R}$ are supposed to be continuous, see section 5.3 for more definitions and precise assumptions.

The resolution of the constrained problem (5.1) becomes more difficult without assuming any controllability assumption, see chapter 4 and even chapter 3 for the case of state-constrained two-person differential games. For this reason, we follow here the level set approach, introduced in [5], where we consider an auxiliary optimal control problem free of state constraints:

$$w(t,x,z) := \inf_{a(\cdot)\in\mathcal{A}} \Big\{ \Big(\int_t^T \ell(y_{t,x}^a(s), a(s))ds + \Phi(y_{t,x}^a(T)) - z \Big) \bigvee \Big(\max_{s\in[t,T]} g(y_{t,x}^a(s)) \Big) \Big\}, \tag{5.3}$$

for $(t,x,z) \in [0,T] \times \mathbb{R}^d \times \mathbb{R}$ and where g is a continuous function that characterizes the constraints set $\mathcal{K}$ as follows:

$$\forall y \in \mathbb{R}^d, \quad g(y) \leq 0 \iff y \in \mathcal{K}.$$

The auxiliary value function w characterizes v in the following way:

$$v(t,x) = \inf\{z \in \mathbb{R} \ \mid \ w(t,x,z) \leq 0\},$$

and can be exploited to determinate the optimal controls and trajectories of problem (5.1), see [5, 6] and chapter 4 for more details. In this chapter, we propose to approximate w by means of neural networks. First, under some assumptions on the problem data, w is approximated by a discrete time auxiliary value function $(W_k)_{k=0}^N$, $N \geq 1$, that verifies a discrete dynamic programming equation. Then, by backward induction and by using $\widehat{W}_{k+1}$, an approximation of W_{k+1} for $k = 0, ..., N-1$, we first compute $\widehat{W}_k$ on a generated training grid. This approximation, computed on a reduced domain, will be used later as an input data to extend $\widehat{W}_k$, by means of neural networks and stochastic optimization, to the whole computational domain which yields an approximation of W_k. This first approach will be compared to another one that consists in approximating w, at any time instant $t \in [0,T]$, by training neural networks, on a reduced domain, in order to satisfy the following HJ equation whose unique solution is w:

$$\begin{cases} \min\left(-\partial_t w(t,\hat{x}) + H(x, D_{\hat{x}} w(t,\hat{x})), w(t,\hat{x}) - g(x)\right) = 0, \quad \text{on } [0,T[\times\mathbb{R}^{d+1}, \\ w(T,\hat{x}) = (\Phi(x) - z) \bigvee g(x), \quad \text{on } \mathbb{R}^{d+1}, \end{cases}$$

for $\hat{x} = (x,z) \in \mathbb{R}^d \times \mathbb{R}$ and where H is the Hamiltonian function that will be made precise later.

This chapter is organized in the following form. Section 5.2 presents some preliminaries on the neural

network approximations and the stochastic optimization widely exploited to train neural networks. Section 5.3 presents the state-constrained optimal control problem of type Bolza and formulates its associated auxiliary problem. Section 5.4 introduces the deep learning numerical methods that will be used to approximate the auxiliary value function. Finally, we present in section 5.5 some illustrative numerical examples to compare the performances of the proposed approaches.

5.2 Neural networks for functions approximation

Consider the problem of approximating a function $J(\cdot) : \mathbb{R}^r \to \mathbb{R}$, $r \geq 1$, given only on some restricted domain $\Gamma \subset \mathbb{R}^r$. This is can be done by considering a *parametric* function $\pi(\cdot;\theta)$, involving some parameters vector $\theta \in \mathbb{R}^p$, $p \geq 1$, that will be chosen in order to minimize some distance measure between $J(\cdot)$ and $\pi(\cdot;\theta)$ expressed on the restricted domain Γ. In this chapter, the set of all the parametric functions $\pi(\cdot;\theta)$, $\theta \in \mathbb{R}^p$, corresponding to some given architecture will be denoted by $\mathcal{W}_r$.

The process of choosing the optimal parameters vector is called *training* or *learning* of the parametric function $\pi(\cdot;\theta)$ and the distance measure between $J(\cdot)$ and $\pi(\cdot;\theta)$ is called the *loss* function, denoted by $L(\cdot;\theta)$. The most popular training method uses least squares optimization (called also least squares *regression*) corresponding to a choice of the loss function L as:

$$L(x;\theta) := \Big(J(x) - \pi(x;\theta)\Big)^2, \quad \text{for any } \ (x,\theta) \in \Gamma \times \mathbb{R}^p. \tag{5.4}$$

In practice, even though $J(\cdot)$ is known on Γ, for numerical issues it may be complicated to evaluate the loss function $L(\cdot;\theta)$, for a given θ, on the whole restricted domain Γ. For this reason, consider a random variable X that lies in Γ and let μ be a given general probability distribution of X ($X \sim \mu$), called the *training distribution*. The stochastic optimization problem to be considered in order to fit $J(\cdot)$ with the approximation $\pi(\cdot;\theta)$ is given by:

$$\inf_{\theta \in \mathbb{R}^p} \ \mathbb{E}\Big[L(X;\theta)\Big]. \tag{5.5}$$

When L is chosen as in (5.4), the stochastic optimization problem (5.5) becomes:

$$\inf_{\pi(\cdot;\theta) \in \mathcal{W}_r} \ \mathbb{E}\Big[\Big(J(X) - \pi(X;\theta)\Big)^2\Big].$$

From the training distribution μ, a training samples X^m, $m = 1, ..., M$, of the random variable X is drawn on the domain Γ where $M \geq 1$ is the number of training points. Henceforth, an estimation of the objective function in the stochastic optimization problem (5.5) can be given by the empirical mean as follows:

$$\mathcal{L}(\theta) = \frac{1}{M} \sum_{m=1}^{M} L(X^m;\theta), \quad \text{for any } \ \theta \in \mathbb{R}^p.$$

Among the most chosen algorithms to deal with large scale optimization for machine learning, one can find Stochastic Gradient Descent (SGD) methods, see [117, 30] and see also [31] for a recent survey. SGD updates the solution of problem (5.5), in an iterative way, as follows:

$$\theta_{k+1} = \theta_k - \gamma_k G(X;\theta_k), \quad \text{for } \ k \geq 0,$$

where $(\gamma_k)_{k=0}^{\infty}$ is a deterministic non-negative sequence representing the learning rates and $G(X;\theta_k)$ is an estimation of the gradient of the loss function L in θ_k, depending on samples of the random variable X. We distinguish essentially 3 types of SGD methods:

- Stochastic Gradient Descent: The gradient of L is approximated over a single random instance among the training samples X^m, for $m = 1, ..., M$. Then, θ_k is updated as follows:

$$\theta_{k+1} = \theta_k - \gamma_k D_\theta L(X^m; \theta_k), \quad \text{for some} \quad m \in \{1, ..., M\},$$

  where X^m is chosen randomly from the training set. This method is fast since it computes the gradient by using only one random instance but it may be unstable.

- Batch Gradient Descent: The gradient of the loss function is computed by using all the training samples. Given θ_k, the next parameters vector is given by:

$$\theta_{k+1} = \theta_k - \gamma_k \frac{1}{M} \sum_{m=1}^{M} D_\theta L(X^m; \theta_k).$$

  Although this method is stable, using the full training set to approximate the gradient of L makes it very slow especially when the size of the training set M is large.

- Mini Batch Gradient Descent: this approach is faster than the Batch Gradient Descent and may be more stable than the SGD because it computes an approximation of the gradient of L over random small subsets of the training samples representing mini-batches. Denote by M_b the size of mini-batches. At any iteration k:

  1. Draw randomly a subset $(X^m)_{m=1}^{M_b}$ from the training set.
  2. Iterate: $\theta_{k+1} = \theta_k - \gamma_k \frac{1}{M_b} \sum_{m=1}^{M_b} D_\theta L(X^m; \theta_k)$.

  Mini Batch Gradient Descent can be seen as a generalization of the two above methods since it coincides with SGD when taking $M_b = 1$ and with Batch Gradient Descent if $M_b = M$.

It is worth to mention that the convergence of SGD methods is heavily affected by the choice of the learning rates $(\gamma_k)_{k=0}^{\infty}$. This was a great motivation to develop new variants of SGD, that adapt, at each iteration of the algorithm, the learning rate γ_k by means of the loss function gradients already computed during the past iterations, see [97, 132, 133]. Examples of such adaptive stepsizes stochastic optimization methods include AdaGrad [55], Adadelta [134], RMSProp [127] and ADAM [88]. Furthermore, we refer to [54, 48, 130, 135] for a study of the convergence of those different adaptive methods. In practice, one can find the different above algorithms are already implemented in TensorFlow, which is an open source platform for machine learning. Tensorflow includes also other stochastic optimization algorithms allowing to approximate the solution of problem (5.5).

In this chapter, we focus on deep neural networks to set the architecture of the parametric functions $\pi(\cdot; \theta)$, $\theta \in \mathbb{R}^p$, that will be used to approximate different unknown target functions. In contrast with the additive approximation theory designed by basis functions, such as polynomials, neural networks are defined through the composition of simple functions.

The mathematical architecture of a deep neural network is presented as a function given by:

$$x \in \mathbb{R}^r \mapsto \pi(x; \theta) := \sigma_I \circ P_I \circ \sigma_{I-1} \circ P_{I-1} \circ ... \circ \sigma_1 \circ P_1(x) \in \mathbb{R}^{d_I}, \tag{5.6}$$

where $(P_i)_{i=1}^I$ are polynomial functions, $(\sigma_i)_{i=1}^I$ are nonlinear monotone functions, called activation functions and $d_I \in \mathbb{N}^*$.

Each polynomial function P_i, for $i = 1, ..., I$, is defined from a matrix $\omega_i \in \mathbb{R}^{d_{i-1} \times d_i}$, called the weight matrix, and a vector $b_i \in \mathbb{R}^{d_i}$, called the bias vector:

$$y \in \mathbb{R}^{d_{i-1}} \mapsto P_i(y) = \omega_i y + b_i \in \mathbb{R}^{d_i}.$$

The collection of the weights matrices and bias terms will be aggregated to define the parameters vector $\theta \in \mathbb{R}^p$ of the neural network $\pi(\cdot;\theta)$. Hence, the size p of θ is directly deduced as follows:

$$p = \sum_{i=1}^{I} d_i(1 + d_{i-1}).$$

The activation function $\sigma_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}$, for $i = 1, ..., I$, will be applied to the outputs of the polynomial function P_i. Among standard examples of activation functions, one can find:

- Sigmoid function, $\sigma : y \in \mathbb{R}^d \mapsto (\frac{1}{1+e^{-y_1}}, ..., \frac{1}{1+e^{-y_d}})$, has a smooth gradient.
- Softmax function, $\sigma : y \in \mathbb{R}^d \mapsto (\frac{e^{y_1}}{\sum_{i=1}^{d} e^{y_i}}, ..., \frac{e^{y_d}}{\sum_{i=1}^{d} e^{y_i}})$, used only for the output layer when classifying the inputs into different categories for example images classification problems.
- Exponential linear unit ELU $\sigma : y \in \mathbb{R}^d \mapsto (\sigma'(y_1), ..., \sigma'(y_d))$ where

  $$\sigma' : x \in \mathbb{R} \mapsto \begin{cases} x, & x > 0 \\ e^x - 1, & x \leq 0. \end{cases}$$

  Elu has a smooth gradient and allows to get non-positive values.
- Rectified linear units ReLU, $\sigma : y \in \mathbb{R}^d \mapsto (\max(0, y_1), ..., \max(0, y_d))$, is less computationally expensive because it involves simple mathematical operations.

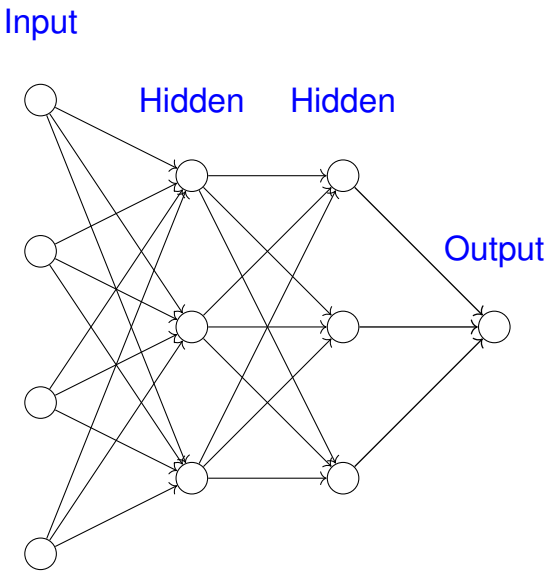

Figure 5.1: Architecture of a deep neural network with input layer $d_0 = 4$, two hidden layers $d_1 = d_2 = 3$ and output layer $d_3 = 1$.

For a given vector $x \in \mathbb{R}^r$, consider $y_0 = x$ and $y_{i+1} = \sigma_{i+1} \circ P_{i+1}(y_i) \in \mathbb{R}^{d_{i+1}}$, $i = 0, ..., I-1$. The neural network (5.6) can be represented by $I+1$ layers where the layer of rank i represent the vector y_i, for $i = 0, ..., I$. In figure 5.1, we represent an example of a deep neural network with $I = 3$.

- The neurons of the layer of rank i represent the coordinates of the vector y_i and hence their number is equal to d_i which is the dimension of y_i. The nodes in figure 5.1 represent the neurons.
- The link between two successive layers of ranks i and $i+1$ respectively, for $i = 0, ..., I-1$, corresponds to the application of the function $\sigma_{i+1} \circ P_{i+1}$ to the vector y_i. Those links are represented by the arrows in figure 5.1. In particular, the arrows drawn from the input layer corresponds to the application of $\sigma_1 \circ P_1$ to the vector $x \in \mathbb{R}^4$ to obtain $y_1 = \sigma_1 \circ P_1(x) \in \mathbb{R}^3$.
- The first layer, with rank $i = 0$, corresponds to the input layer with $d_0 = r$ neurons. In figure 5.1, $r = 4$.
- $I-1$ hidden layers ranked from 1 to $I-1$. The are two hidden layers for the example given in figure 5.1 with numbers of neurons $d_1 = d_2 = 3$.
- The last layer of rank $i = I$ corresponds to the output layer with d_I neurons. When approximating real-valued functions, $d_I = 1$ which is the case in figure 5.1 where $y_3 \in \mathbb{R}$.

The relevance of using neural networks to approximate complex and non-linear functions can be theoretically justified by the universal approximation theorems, see [74, 98, 52, 72, 99]. In particular, we have the following approximation theorem:

Theorem 5.2.1 (Universal approximation theorem, see [73]). *Any measurable function $J : \mathbb{R}^r \to \mathbb{R}$ can be approximated by a neural network with a single hidden layer and a continuous non-constant activation function. Moreover, when the activation function, used in the neural networks architecture, is of class C^k, $k \geq 1$, then this class of neural networks approximates also the derivatives of J up to order k.*

Moreover, deep neural networks are capable of approximating real-valued continuous functions over compact subsets of $\mathbb{R}^r$ with an arbitrary accuracy, see [74].

A more precise approximation result is established when using neural networks with a single hidden layer to fit a Lipschitz continuous function J, see [77]. Let $\mathcal{W}_r^{K,\gamma}$ be the set of neural networks composed by only one hidden layer with K neurons, ReLU activation function for the hidden layer, a total variation[1] smaller than γ and no activation function for the output layer

$$\mathcal{W}_r^{K,\gamma} := \Big\{ \pi(\cdot;\theta) : \mathbb{R}^r \to \mathbb{R},\ \theta = \Big((\omega_i)_{1\le i\le K}, (b_i)_{1\le i\le K}, (\nu_i)_{0\le i\le K}\Big),\ \text{s.t. for any } i = 1, ..., K$$
$$\omega_i \in \mathbb{R}^r,\ b_i, \nu_i, , \nu_0 \in \mathbb{R},\ \sum_{i=0}^{K} \nu_i \le \gamma \ \text{ and } \ \pi(y;\theta) = \sum_{i=1}^{K} \nu_i \max(\langle \omega_i, y\rangle + b_i, 0) + \nu_0 \Big\}. \quad (5.7)$$

In this setting, we get a rate of convergence of the approximation error that depends on the Lipschitz constant L_J of J, the dimension r, the number of neurons K and γ:

Theorem 5.2.2. *Given $K \in \mathbb{N}^*$ and $\gamma > 0$, there exists a neural network $\pi(\cdot;\theta^*) \in \mathcal{W}_r^{K,\gamma}$, with $\theta^* \in \mathbb{R}^{(r+2)K+1}$, such that:*

$$\|\pi^* - J\|_\infty \le L_J \Big(\frac{\gamma}{L_J}\Big)^{\frac{-2}{r+1}} \log\Big(\frac{\gamma}{L_J}\Big) + \gamma K^{-\frac{r+3}{2r}},$$

over compact subsets of $\mathbb{R}^r$.

5.3 Problem settings

For a finite time horizon $T > 0$ and a non-linear dynamics f, consider the following dynamical system:

$$\begin{cases} \dot{y}(s) = f(y(s), a(s)) \quad \text{a.e.} \quad s \in [t, T], \\ y(t) = x \in \mathbb{R}^d, \end{cases} \quad (5.8)$$

where the input variable $a(\cdot)$ takes values in A, a compact set of $\mathbb{R}^q$ (for $q \ge 1$).

The state-constrained optimal control problem with a distributed cost function ℓ and a final cost function Φ, is defined as follows:

$$v(t,x) := \inf_{a(\cdot)\in\mathcal{A}} \Big\{ \int_t^T \ell(y_{t,x}^a(s), a(s))ds + \Phi(y_{t,x}^a(T)) \mid y_{t,x}^a(s) \in \mathcal{K},\ \ \forall s \in [t,T] \Big\}, \quad (5.9)$$

where $\mathcal{K} \subset \mathbb{R}^d$ is a non-empty and closed set representing the set of state constraints, $y_{t,x}^a(\cdot)$ is the absolutely continuous solution of (5.8) and $\mathcal{A}$ is the set of admissible controls defined by:

$$\mathcal{A} := \{a(\cdot) : [0,T] \to A,\ \ \text{measurable}\}.$$

Throughout this chapter the dynamics f and the distributed cost ℓ are continuous functions and in contrary to chapter 4, they are supposed to be Lipschitz continuous w.r.t. only the state variable. Moreover, asume that $\Phi : \mathbb{R}^d \to \mathbb{R}$ is a Lipschitz continuous function.

As we have mentioned before in chapters 1, 3 and 4, in the presence of state constraints, $\mathcal{K} \neq \mathbb{R}^d$, some difficulties concerning the regularity of v and its characterization may appear. In particular, v may be discontinuous unless some controllability assumptions are satisfied and therefore one cannot guarantee an accurate approximation of such function by means of neural networks. For this reason and similarly to the

[1] The total variation of $\mathcal{W}_r^{K,\gamma}$ is equal $\sum_{i=0}^{K} \nu_i$.

previous chapters, we follow here the level set approach, introduced in [5], which consists in characterizing the constrained problem (5.9) by means of an auxiliary optimal control problem free of state constraints.

First, since the set of constraints $\mathcal{K}$ is a closed subset of $\mathbb{R}^d$, there exists a Lipschitz continuous function g characterizing $\mathcal{K}$ in the following way:

$$\forall y \in \mathbb{R}^d, \quad g(y) \leq 0 \iff y \in \mathcal{K}.$$

The value function of the auxiliary control problem associated to the constrained problem (5.9) is given by:

$$w(t,x,z) := \inf_{a(\cdot)\in\mathcal{A}} \Big\{ \Big(\int_t^T \ell(y^a_{t,x}(s), a(s))ds + \Phi(y^a_{t,x}(T)) - z \Big) \bigvee \Big(\max_{s\in[t,T]} g(y^a_{t,x}(s)) \Big) \Big\}, \tag{5.10}$$

for $(t,x,z) \in [0,T] \times \mathbb{R}^d \times \mathbb{R}$.

It is known that the auxiliary value function w is Lipschitz continuous, satisfies a dynamic programming principle and hence can be characterized as the unique viscosity solution of an HJ equation, see [5, 6] and chapter 4 for single-controller problems and even chapter 3 for zero-sum differential games. Moreover, v can be determined as follows:

$$v(t,x) = \inf\{z \in \mathbb{R} \mid w(t,x,z) \leq 0\}, \tag{5.11}$$

when some convexity assumption is verified (see Assumption **(H4.4)** in chapter 4). In addition to that, the optimal controls and trajectories of problem (5.9) can be characterized by use of w. The auxiliary value function is approximated in chapter 3, for the case of two-person differential games, by solving numerically the corresponding HJ equation and by means of optimistic planning methods in chapter 4. In this chapter, we propose two different numerical methods based on the training of neural networks to approximate w by exploiting either the dynamic programming principle or its associated HJ equation.

5.4 Deep learning numerical methods

The aim of this section is to present the different numerical approaches that will be used to approximate the auxiliary value function w.

5.4.1 Neural networks for dynamic programming (DP)

Consider a uniform partition of $[0,T]$ with N time steps, $s_0 = 0, ..., s_k = kh, ..., s_N = Nh = T$, where $N \in \mathbb{N}^*$ is the number of time steps and $h := \frac{T}{N}$ is the time steps size. An approximation in time of w can be given by:

$$w_h(t,x,z) := \min_{(a_k)_k \in A^{N-k}} \left\{ \Big((s_{k+1} - t)\ell(x, a_k) + h \sum_{i=k+1}^{N-1} \ell(y^a_i, a_i) + \Phi(y^a_N) - z \Big) \bigvee \Big(\max_{k\leq i\leq N} g(y^a_i) \Big) \right\}, \tag{5.12}$$

for $t \in [s_k, s_{k+1}[$, with $0 \leq k \leq N-1$, $(x,z) \in \mathbb{R}^d \times \mathbb{R}$ and $(y^a_i)_i$ is a discrete trajectory associated to (5.8) (see chapters 3 and 4 for more details).

For the sequel, let's define $W_k(\cdot,\cdot) := w_h(s_k, \cdot, \cdot)$, for $k = 0, ..., N$. The discrete auxiliary value function $(W_k)_{k=0}^N$ verifies the following properties.

Proposition 5.4.1. (i) *For any $k = 0, ..., N$, W_k is a Lipschitz continuous function.*

(ii) $(W_k)_{k=0}^N$ *is the unique solution of the following discrete dynamic programming equation:*

$$\begin{cases} W_N(x,z) = \Big(\Phi(x) - z\Big) \bigvee g(x), \\ W_k(x,z) = \min\limits_{a\in A}\Big\{W_{k+1}(\hat{F}_h(\hat{x},a)) \bigvee g(x)\Big\}, \quad \textit{with } \hat{x} := (x,z) \quad \textit{for } k = N-1,...,0 \end{cases} \tag{5.13}$$

where $\hat{F}_h := I_d + h\hat{f}$ *with* $\hat{f}$ *is the augmented dynamics given by* $\hat{f}(x,a) := \begin{pmatrix} f(x,a) \\ -\ell(x,a) \end{pmatrix}$ *(other approximations, such as the Heun scheme, can be considered to define* $\hat{F}_h$*).*

The proof of Proposition 5.4.1 can be done by using some classical arguments, see for instance [11, Chapter III].

This first subsection is devoted to present a deep learning algorithm to approximate the discrete auxiliary value function $(W_k)_{k=0}^N$ by training neural networks. Here, we adapt the *Hyprid-Now* Algorithm presented in [78, 10, 77] for stochastic control problems to the deterministic case with state constraints. In [78, 10, 77], the optimal policy is first estimated by neural networks and dynamic programming. Then, the estimated control policy is injected in a backward process in order to approximate the value function by means of neural networks. In our case, the optimal control is not always regular enough therefore its approximation by neural networks may not be a good idea. To this end, we propose to approximate only the discrete auxiliary value function by training neural networks and by exploiting the dynamic programming principle (5.13).

The constraints set $\mathcal{K}$ may be unbounded. Nevertheless in practice, we will compute v on a restricted and bounded subset of $\mathcal{K}$. Moreover, starting from bounded initial states and by studying the evolution of the dynamics f, one can get bounds on the cost functions and hence on the auxiliary variable z in order to determinate v through the relation (5.11). For this reason, we will need to evaluate the auxiliary value function w only on some compact set $\Omega \subset \mathcal{K} \times \mathbb{R}$, that will be made precise later for each numerical example.

Recall that $\mathcal{W}_{d+1}$ denotes the set of neural networks which is the set of all the parametric functions $\pi(\cdot;\theta) : \mathbb{R}^{d+1} \to \mathbb{R}$ corresponding to some given architecture where $\theta \in \mathbb{R}^p$ is the parameters vector and its size $p \geq 1$ can be calculated as in section 5.2.

Finally, at each time step $k = 0,...,N-1$ and in order to train the neural networks to fit W_k, we will consider a random variable $\hat{X}_k$, on the compact set Ω, described by some given probability distribution μ_k, called the training distribution. For sake of simplicity, we consider that μ_k does not depend on the time step k and let $\mu_k = \mu$, $\forall k \in \{0,...,N-1\}$, where μ is a given training distribution.

Algorithm 5.1: Deep learning algorithm to approximate the value function

1: Initialize $\widehat{W}_N(x,z) = (\Phi(x) - z) \bigvee g(x)$ for any $(x,z) \in \Omega$.
2: **for** $k = N-1$,...,0 **do**
3: Knowing $\widehat{W}_{k+1}(\cdot)$, compute the neural network $\widehat{W}_k(\cdot)$ as follows:

$$\widehat{W}_k \in \underset{\pi(\cdot;\theta)\in\mathcal{W}_{d+1}}{\text{argmin}} \ \mathbb{E}\Big[\Big(\min_{a\in A}\Big\{\widehat{W}_{k+1}(\hat{F}_h(\hat{X}_k,a)) \bigvee g(X_k)\Big\} - \pi(\hat{X}_k;\theta)\Big)^2\Big], \tag{5.14}$$

where $\hat{X}_k := (X_k, Z_k) \sim \mu$ is a random variable lying in Ω.
4: **end for**
5: **return** $\widehat{W}_k(\cdot)$, for $k = 0,...,N$.

Algorithm 5.2 iterates in time, in a backward way, and tries at each time step k, for $k = 0,...,N-1$, to find the best neural network $\pi(\cdot;\theta) \in \mathcal{W}_{d+1}$ that approximates $W_k(\cdot)$ by use of the neural network $\widehat{W}_{k+1}(\cdot)$ already obtained in the previous time step $k+1$.

Remark 5.4.2. *1. The approximated value function $\widehat{W}_k(\cdot)$ is computed in (5.14) by considering a training samples $\hat{X}_k^m := (X_k^m, Z_k^m)$, $m = 1, ..., M$, of the random variable $\hat{X}_k$, drawn from the training distribution μ on the computational domain Ω where $M \geq 1$ is the number of training points. The corresponding optimization problem to be solved, by means of the Mini Batch Gradient Descent method presented in section 5.2, has the following form:*

$$\inf_{\pi(\cdot;\theta)\in\mathcal{W}_{d+1}} \frac{1}{M}\sum_{m=1}^{M}\left(\min_{a\in A}\Big\{\widehat{W}_{k+1}(\hat{F}_h(\hat{X}_k^m, a)) \bigvee g(X_k^m)\Big\} - \pi(\hat{X}_k^m;\theta)\right)^2.$$

2. Furthermore, the minimum over A in (5.14) will be approximated by the minimum over a discretized set from A.

3. Algorithm 5.2 can be extended for the case of two-person zero-sum differential games. To this end, one shall exploit the discrete dynamic programming principle corresponding to games, see chapter 3, in the training step (5.14) that becomes:

$$\widehat{W}_k \in \underset{\pi(\cdot;\theta)\in\mathcal{W}_{d+1}}{\textit{argmin}}\ \mathbb{E}\Big[\Big(\max_{b\in B}\min_{a\in A}\Big\{\widehat{W}_{k+1}(\hat{F}_h(\hat{X}_k, a, b)) \bigvee g(X_k)\Big\} - \pi(\hat{X}_k;\theta)\Big)^2\Big],$$

where B is a compact set in which controls of the second player take values.

5.4.2 Neural networks for partial derivatives equations (PDE)

We already know that the auxiliary value function w is the unique viscosity solution of the following Hamilton-Jacobi equation, see [5, 6]:

$$\begin{cases}\min\Big(-\partial_t w(t,\hat{x}) + H(x, D_{\hat{x}}w(t,\hat{x})), w(t,\hat{x}) - g(x)\Big) = 0, & \text{on } [0,T[\times\mathbb{R}^{d+1},\\ w(T,\hat{x}) = (\Phi(x) - z) \bigvee g(x), & \text{on } \mathbb{R}^{d+1},\end{cases}$$

where $\hat{x} := (x, z) \in \mathbb{R}^d \times \mathbb{R}$ and the Hamiltonian H is given by:

$$H(x,p) = \max_{a\in A} -\langle \hat{f}(x,a), p\rangle, \quad \text{for} \quad (x,p) \in \mathbb{R}^d \times \mathbb{R}^{d+1}.$$

The PDE approach consists in approximating the solution of the above HJ equation through training spatio-temporal function approximators to fit the terminal condition, at time T, and to satisfy the equation law. To this end, let $\mathcal{W}_{d+2}$ denote the set of all the parametric functions (neural networks) $\pi(\cdot,\cdot;\theta) : \mathbb{R}^{d+2} \to \mathbb{R}$ corresponding to some given architecture where $\theta \in \mathbb{R}^p$ is the parameters vector. Moreover, consider a given training distribution μ on the computational domain $[0,T] \times \Omega$ where Ω is already defined in the previous section.

This approximated value function is computed in (5.15) by considering a training samples $(\tau^m, \hat{X}^m)$, $m = 1, ..., M$, of the random variable $(\tau, \hat{X})$, drawn from the training distribution μ on the computational domain $[0,T] \times \Omega$ where $M \geq 1$ is the number of training points for the equation law. Moreover, to fit w at the final time T, one shall consider training samples $\hat{X}^m := (X^m, Z^m)$, $m = 1, ..., M_0$, of the random variable $\hat{X}$ drawn on Ω with $M_0 \geq 1$. The corresponding optimization problem to be solved has the following form:

$$\min_{\pi(\cdot,\cdot;\theta)\in\mathcal{W}_{d+2}} \frac{1}{M}\sum_{m=1}^{M}\left(\min\Big(-\partial_t\pi(\tau^m,\hat{X}^m;\theta) + H(X^m, D_{\hat{x}}\pi(\tau^m,\hat{X}^m;\theta)), \pi(\tau^m,\hat{X}^m;\theta) - g(X^m)\Big)\right)^2$$
$$+ \frac{1}{M_0}\sum_{m=1}^{M_0}\Big(\pi(T,\hat{X}^m;\theta) - w_T(\hat{X}^m)\Big)^2$$

Algorithm 5.2: Deep learning algorithm to approximate the solution of an HJ equation

1: Let $(\tau, \hat{X}) := (\tau, X, Z)$ be a random variable, described by μ, that lies in $[0,T] \times \Omega$.

2: The returned neural network, is the one that minimizes, over $\mathcal{W}_{d+2}$, the sum of the following expected quadratic loss functions:

$$\widehat{W} \in \underset{\pi(\cdot,\cdot;\theta)\in\mathcal{W}_{d+2}}{\text{argmin}} \mathbb{E}\Big[\Big(\min\Big(-\partial_t\pi(\tau,\hat{X};\theta) + H(X, D_{\hat{x}}\pi(\tau,\hat{X};\theta)), \pi(\tau,\hat{X};\theta) - g(X)\Big)\Big)^2\Big] + \mathbb{E}\Big[\Big(\pi(T,\hat{X};\theta) - w_T(\hat{X})\Big)^2\Big] \tag{5.15}$$

where $w_T(\hat{x}) := (\Phi(x) - z) \bigvee g(x)$ for $\hat{x} = (x,z) \in \mathbb{R}^d \times \mathbb{R}$.

3: **return** $\widehat{W}$.

Remark 5.4.3. *1. The partial derivatives of the neural networks here are computed by means of automatic differentiation [17]. One can also estimate those derivatives by considering other neural networks, see [79, 80]. Nevertheless, such method will increase the complexity in time of the training process since further variables (parameters of the additional neural networks) will be involved in the optimization problem (5.15).*

2. The PDE approach can also handle two-person zero-sum differential games where the hamiltonian function becomes given by:

$$H(x,p) = \min_{b\in B}\max_{a\in A} -\langle \hat{f}(x,a,b), p\rangle, \quad \textit{for} \quad (x,p) \in \mathbb{R}^d \times \mathbb{R}^{d+1}.$$

5.5 Numerical examples

In all the following numerical examples:

- The training distribution μ is the uniform probability distribution on the computational domain Ω that will be made precise for each example.
- The hidden layers of the neural networks architectures considered are composed by the same number of neurons, $i.e.$ $d_i = d_{i+1}$, $\forall i \in \{1, ..., I-2\}$. Notice that d_0 is equal to the dimension of the system state for the DP approach and it is increased by one, for the PDE method, since the time variable is involved in the training process. Moreover, $d_I = 1$ because we approximate real-valued functions. On the other hand, we take as an activation function ELU, $i.e.$ $\sigma_i =$ ELU, $\forall i \in \{1, ..., I\}$.
- The stochastic optimization algorithm that will be used to train neural networks, $i.e.$ to solve problems (5.14) and (5.15), is ADAM with mini batch, see [88].
- The neural networks will be trained on a training grid of size $M \geq 1$ and tested on a test grid $\mathcal{G}$ of size $N_{\mathcal{G}}$. The training grid is built on the computational domain Ω, from the training distribution μ, while $\mathcal{G}$ corresponds to a uniform grid on Ω.
- The minimum over the control set A in (5.14) is approximated by taking the minimum over a uniform grid A_g on A with size $n_A \in \mathbb{N}^*$.
- Because of the stochastic optimization used in (5.14) and (5.15), we notice that the numerical results (errors when knowing the exact solution or cost functional values for the control of the heat equation (example 2)) corresponding to two different executions may be very different. For this reason, all

the numerical results presented in the following tables correspond to the mean over 5 different measures. However, the illustrative figures (approximated value functions and optimal trajectories and controls) correspond to the best measure among the 5 effectuated measures.

- Our numerical methods are implemented in PYTHON with the Tensorflow library and all the computations are done with a computer that uses an Intel Core i5 at 1,8 GHz with 8 Go RAM.

Example 1: 1D problem without state constraints

Let $T = 1$ and $A = [0, 1]$. The dynamics, the distributed and the final cost functions are given by:

$$f(x, a) = -xa, \quad \ell(x, a) = x \quad \text{and} \quad \Phi(x) = 0.$$

The exact value function is given, for $(t, x) \in [0, T] \times \mathbb{R}$, by:

$$v(t, x) = \begin{cases} Tx, & \text{for} \quad x \leq 0, \\ (e^{-t} - e^{-T})x, & \text{for} \quad x > 0. \end{cases}$$

Notice that this example is without state constraints. Henceforth, the DP and PDE approaches will be directly applied to approximate the unconstrained value function v. The computational domain for this example is $\Omega = [-1, 1]$. For the DP approach, we used $N = 20$ time steps for the time-discretization of $[0, T]$ and $n_A = 11$ points to generate the uniform control grid A_g from the control set A.

The relative errors presented in the following tables are computed with respect to the exact value function, at the initial time instant $t = 0$, as follow:

$$\mathcal{L}_1 \text{ error} := \frac{\sum_{i=1}^{N_\mathcal{G}} |\widehat{V}_0(x_i) - v(0, x_i)|}{\sum_{i=1}^{N_\mathcal{G}} |v(0, x_i)|},$$

$$\mathcal{L}_2 \text{ error} := \sqrt{\frac{\sum_{i=1}^{N_\mathcal{G}} (\widehat{V}_0(x_i) - v(0, x_i))^2}{\sum_{i=1}^{N_\mathcal{G}} v(0, x_i)^2}},$$

$$\mathcal{L}_\infty \text{ error} := \frac{\max\limits_{1 \leq i \leq N_\mathcal{G}} |\widehat{V}_0(x_i) - v(0, x_i)|}{\max\limits_{1 \leq i \leq N_\mathcal{G}} |v(0, x_i)|},$$

where $N_\mathcal{G} = 5000$ is the size of the uniform grid $\mathcal{G}$ on the computational domain Ω and $\widehat{V}_0(\cdot)$ is an approximation of the value function at the initial time instant, $v(0, \cdot)$.

Training parameters			$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M				
2	10	50	3.417 e-02	3.762 e-02	4.511 e-02	15.22
4	20	100	1.445 e-02	1.493 e-02	1.480 e-02	23.03
6	40	200	1.236 e-02	1.247 e-02	1.341 e-02	53.48
8	60	400	1.194 e-02	1.218 e-02	1.325 e-02	93.56

Table 5.1: (Example 1): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solution obtained by Algorithm 5.2 with different network architectures and values of M.

First, we mention that the training samples size M for the DP approach is less than the one used for the PDE approach since for the latter the training involves the time and the space variables. In tables 5.1 and 5.2, we remark an amelioration, in general, of the error estimates when increasing simultaneously the

Training parameters				$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M	M_0				
2	10	125	10	1.828 e-02	1.861 e-02	2.748 e-02	2.31
4	20	250	20	1.457 e-02	1.501 e-02	2.063 e-02	7.40
6	40	5×10^2	40	1.050 e-02	1.152 e-02	1.178 e-02	49.72
8	60	10^3	80	1.097 e-02	1.168 e-02	1.194 e-02	85.41

Table 5.2: (Example 1): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solution obtained by the PDE approach with different network architectures and values of M_0 and M.

training samples size, the depth of the neural networks and the number of neurons per layer. Moreover, we observe that the approximation obtained by the PDE approach is in general more accurate than the DP approximation. This can be explained by the error accumulation through the time steps for the DP approach.

We represent in figure 5.2 the exact and the approximated value functions, obtained by the PDE approach.

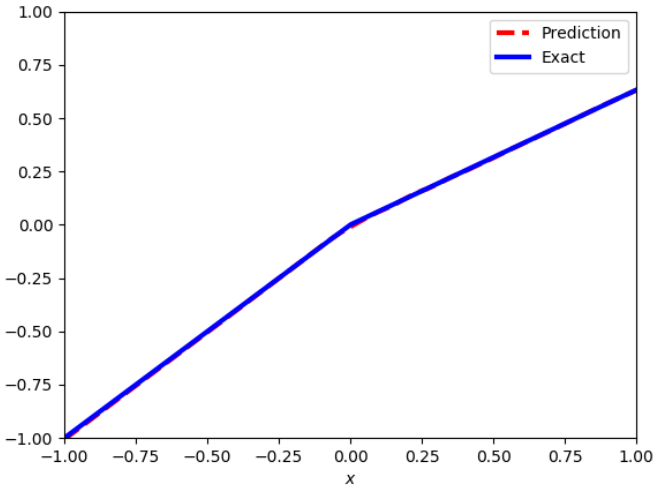

Figure 5.2: (Example 1): Exact and predicted value functions at $t = 0$, $v(0, \cdot)$ and $\hat{V}_0(\cdot)$ obtained by the PDE approach with 6 layers, 40 neurons per layer, $M = 5 \times 10^2$ and $M_0 = 40$.

Example 2: Optimal control of the heat equation

Secondly, we consider the control problem of the heat equation taken from [3]. Recall that this problem was already studied and solved in chapter 4 and its discrete formulation leads to a control problem in a high dimensional state space. The corresponding partial derivatives equation is given by:

$$\begin{cases} \frac{\partial y}{\partial t}(s,x) = \sigma \frac{\partial^2 y}{\partial x^2}(s,x) + y_0(x)a(s), \quad \text{for} \quad (s,x) \in [0,T] \times [0,1], \\ y(s,x) = 0, \quad \text{for} \quad (s,x) \in [0,T] \times \{0,1\}, \\ y(0,x) = y_0(x), \quad \text{for} \quad x \in [0,1], \end{cases} \tag{5.16}$$

where $\sigma = 0.1$, the control $a(\cdot)$ takes values in $A = [-1, 1]$, $T = 1$ and $y_0(x) = -x^2 + x$, for $x \in [0, 1]$. In order to transform (5.16) into a dynamical system similar to (5.8), we first consider a space grid with $d \in \mathbb{N}^*$ points on $]0, 1[$, $x_j = j\Delta x$ for $j = 1, ..., d$ and where the space step is defined as $\Delta x = \frac{1}{d+1}$. Then

we use the centered finite difference scheme to obtain:

$$\begin{cases} \dot{Y}(s) = D \times Y(s) + a(s) \times Y_0, \quad s \in [0,T], \\ Y(0) = Y_0, \end{cases} \tag{5.17}$$

where $Y(s) \in \mathbb{R}^d$ is an approximation of the solution at time s and over the space grid, the matrix $D \in \mathbb{R}^{d\times d}$ is given by $D = \frac{-\sigma}{\Delta x^2} P$, with $P \in \mathbb{R}^{d\times d}$ is already defined in chapter 4, and the vector $Y_0 \in \mathbb{R}^d$ is given by $Y_0 = (y_0(x_j))_{j=1}^d$.

The aim is to minimize, by using the control input $a(\cdot)$, the temperature $Y^a(\cdot)$ which is the solution of (5.17). For this reason, we consider the following cost functional of type Bolza:

$$J(y_0, a) = \int_0^T \Big(\Delta x \|Y^a(s)\|^2 + \gamma a^2(s)\Big) ds + \Delta x \|Y^a(T)\|^2, \tag{5.18}$$

where $\gamma > 0$. Therefore, the Hamiltonian function that will be considered for the PDE approach is given by:

$$H(Y,p) = -\langle D \times Y, p\rangle - \Delta x \|Y\|^2 + \max_{a\in A}\Big\{ -\langle B, p\rangle a - \gamma a^2 \Big\}, \quad \text{for } Y, p \in \mathbb{R}^d.$$

As for the DP approach, we will use a time-discretization of $[0,T]$ with $N = 20$ time steps, $t_k = kh$ for $k = 0, ..., N$, where $h = \frac{T}{N}$. Applying the implicit scheme for (5.17) leads to the following discrete dynamics

$$Y_{k+1}^a = F_h(Y_k^a, a_k) := \Big(I_d - hD\Big)^{-1}\Big(Y_k^a + h \times a_k \times Y_0\Big), \tag{5.19}$$

where $Y_k^a \in \mathbb{R}^d$ is an approximation of the solution of (5.17) at time t_k, see chapter 4 for more details. On the other hand, the cost functional J, defined in (5.18), can be approximated by:

$$\mathcal{J}(Y_0, a) = \sum_{k=0}^{N-1} \rho_h(Y_k^a, a_k) + \Phi(Y_N^a),$$

where the instantaneous cost ρ_h and the final cost Φ are given, for $Y \in \mathbb{R}^d$ and $a \in A$, by:

$$\rho_h(Y,a) = \frac{h}{2}\Big(\Delta x \|Y\|^2 + \Delta x \|F_h(Y,a)\|^2 + 2\gamma a^2\Big) \quad \text{and} \quad \Phi(Y) = \Delta x \|Y\|^2.$$

The computational domain is given by $\Omega =]0,1[^d$ and the training grid $(y_i)_{i=1}^M$, with $y_i \in]0,1[^d$, is generated as follow:

- Let $(x_j)_{j=1}^d$ be the uniform partition of $]0,1[$, defined above, and $(\alpha_i)_{i=1}^M$ be a uniform distribution in $[0,1]$.
- For any $i = 1, ..., M$, $y_i := \alpha_i Y_0$, with $Y_0 = (y_0(x_j))_{j=1}^d \in]0,1[^d$.

Moreover, for this example the test grid coincides with the initial position Y_0. The approximated optimal controls and trajectories, $(a_k^*)_{k=0}^{N-1}$ and $(Y_k^*)_{k=0}^N$ with $Y_k^* \in]0,1[^d$, are computed in feedback form by using the approximated value function $(\widehat{V}_k)_{k=0}^N$ as follows:

$$\begin{cases} Y_0^* = Y_0 \in]0,1[^d, \\ a_k^* = \underset{a\in A_g}{\text{argmin}}\Big\{\rho_h(Y_k^*, a) + \widehat{V}_{k+1}(F_h(Y_k^*, a))\Big\}, \quad k = 0, ..., N-1, \\ Y_{k+1}^* = F_h(Y_k^*, a_k^*), \quad k = 0, ..., N-1, \end{cases}$$

where A_g is the control grid obtained by the uniform discretization of A with a number of points $n_A = 11$. Henceforth, the cost functional $\mathcal{J}$, corresponding to the solution $((a_k^*)_{k=0}^{N-1}, (Y_k^*)_{k=0}^{N})$, is computed as:

$$\mathcal{J}(Y_0, a^*) = \sum_{k=0}^{N-1} \rho_h(Y_k^*, a_k^*) + \Phi(Y_N^*).$$

In tables 5.3 and 5.5, we present the values of the controlled cost functional $\mathcal{J}(Y_0, a^*)$ for different values of γ corresponding to the dimensions $d = 10^2$ and $d = 10^3$ respectively and obtained by the DP while tables 5.4 and 5.6 corresponds to the PDE approach. We observe that the controlled values obtained by the DP approach are less than those corresponding to the PDE approach which are grater even than the uncontrolled cost $\mathcal{J}(Y_0, 0)$ (for instance $\mathcal{J}(Y_0, 0) = 2.038$ e-02 for $d = 10^2$). This observation shows the bad performance of the PDE approach and its high complexity, compared to the DP method, for large values of the dimension d. We observe also, in the different tables, that we ameliorate, in general, the quality of the solution when increasing simultaneously the number of hidden layers, the number of neurons and the size of the training set. However, this results in increasing the CPU time needed to train the neural networks.

Training parameters			$\gamma = 10^{-2}$		$\gamma = 10^{-4}$	
Layers	Neurons	M	$\mathcal{J}(Y_0, a^*)$	CPU(s)	$\mathcal{J}(Y_0, a^*)$	CPU(s)
2	10	50	1.774 e-02	17.40	9.557 e-03	17.47
4	20	100	1.606 e-02	25.71	9.242 e-03	21.36
6	40	200	1.585 e-02	39.87	9.213 e-03	30.53
8	60	400	1.531 e-02	51.70	9.208 e-03	63.24

Table 5.3: (Example 2): Values of the cost $\mathcal{J}$ corresponding to the controlled solutions obtained by Algorithm 5.2 for different neural network architectures and different values of the training set size M and for $d = 10^2$.

Training parameters				$\gamma = 10^{-2}$		$\gamma = 10^{-4}$	
Layers	Neurons	M	M_0	$\mathcal{J}(Y_0, a^*)$	CPU(s)	$\mathcal{J}(Y_0, a^*)$	CPU(s)
2	10	5×10^2	50	2.938 e-02	16.69	2.415 e-02	17.30
4	20	10^3	10^2	2.801 e-02	32.74	2.298 e-02	34.61
6	40	2×10^3	2×10^2	2.835 e-02	75.48	2.316 e-02	78.38
8	60	4×10^3	4×10^2	2.642 e-02	210.70	2.114 e-02	221.77

Table 5.4: (Example 2): Values of the cost $\mathcal{J}$ corresponding to the controlled solutions obtained by the PDE approach for different neural network architectures and different values of the training sets sizes M, M_0 and for $d = 10^2$.

The uncontrolled solution corresponds to a numerical solution of (5.16) when taking $a(\cdot) = 0$. As expected, we observe in figure 5.3 that the controlled solution, obtained by Algorithm 5.2 for $d = 10^3$, is below the uncontrolled solution. Moreover, figure 5.4 represents the evolution of the loss function versus the number of iterations of the stochastic gradient algorithm used for the training of the neural networks in (5.14), for $\gamma = 10^{-2}$ (a similar evolution of the loss, not represented here, is obtained for $\gamma = 10^{-4}$). We remark that the loss reaches some threshold, for high number of iterations, and does not decrease enough (the behaviour of the loss function becomes of the form $C\alpha^n$, where n is the number of iterations, $C > 0$ is a small real constant and $\alpha < 1$ is close to 1).

Furthermore, the controlled solution corresponding to $\gamma = 10^{-4}$ is below the controlled solution corresponding to $\gamma = 10^{-2}$, see figure 5.3. This observation can be explained by the control differences

Training parameters			$\gamma = 10^{-2}$		$\gamma = 10^{-4}$	
Layers	Neurons	M	$\mathcal{J}(Y_0, a^*)$	CPU(s)	$\mathcal{J}(Y_0, a^*)$	CPU(s)
2	10	5×10^2	1.892 e-02	30.10	9.373 e-03	29.23
4	20	10^3	1.687 e-02	55.49	9.215 e-03	48.13
6	40	2×10^3	1.495 e-02	99.22	9.033 e-03	110.26
8	60	4×10^3	1.513 e-02	249.09	9.095 e-03	238.02

Table 5.5: (Example 2): Values of the cost $\mathcal{J}$ corresponding to the controlled solutions obtained by Algorithm 5.2 for different neural network architectures and different values of the training set size M and for $d = 10^3$.

Training parameters				$\gamma = 10^{-2}$		$\gamma = 10^{-4}$	
Layers	Neurons	M	M_0	$\mathcal{J}(Y_0, a^*)$	CPU(s)	$\mathcal{J}(Y_0, a^*)$	CPU(s)
2	10	5×10^3	5×10^2	2.625 e-02	257.86	1.750 e-02	256.28
4	20	10^4	10^3	2.491 e-02	526.33	1.626 e-02	561.35
6	40	2×10^4	2×10^3	2.376 e-02	1194.07	1.603 e-02	1236.19
8	60	4×10^4	4×10^3	2.305 e-02	2735.58	1.590 e-02	2610.62

Table 5.6: (Example 2): Values of the cost $\mathcal{J}$ corresponding to the controlled solutions obtained by the PDE approach for different neural network architectures and different values of the training sets sizes M, M_0 and for $d = 10^3$.

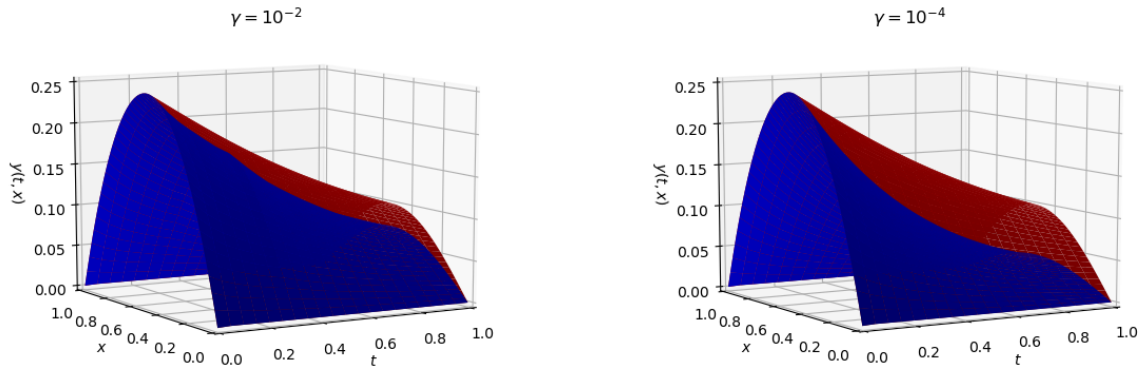

Figure 5.3: (Example 2): Uncontrolled (in red) and controlled solutions (in blue) when $d = 10^3$, corresponding to $\gamma = 10^{-2}$ (left) and $\gamma = 10^{-4}$ (right) obtained by Algorithm 5.2 with 6 layers, 40 neurons per layer and $M = 2000$.

between the two cases, see figure 5.5. Indeed, when γ is equal to 10^{-4}, we allow values of the control with larger norms as it is shown in figure 5.5. In addition to that, due to its important weight in the distributed cost function, the control corresponding to $\gamma = 10^{-2}$ is more regular than the control obtained with $\gamma = 10^{-4}$.

On the other hand, we remark that the DP method requires approximately the same execution time used to solve this example by the *Optimistic Planning* approach (see example 3 of section 4.6). Because of the stochastic optimization process used for the regression (5.14) in Algorithm 5.2, the numerical results

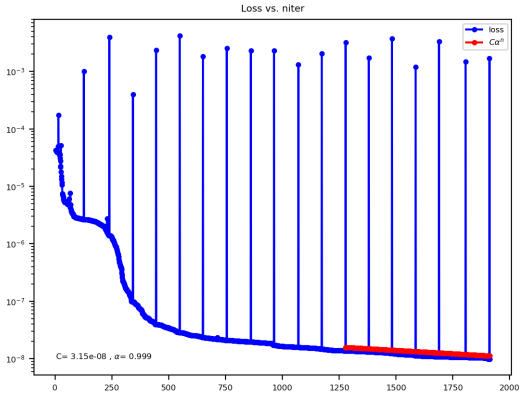

Figure 5.4: (Example 2): Evolution of the loss function versus the number of iterations of the stochastic gradient algorithm for $d = 10^3$ and $\gamma = 10^{-2}$ (6 layers, 40 neurons per layer and $M = 2000$).

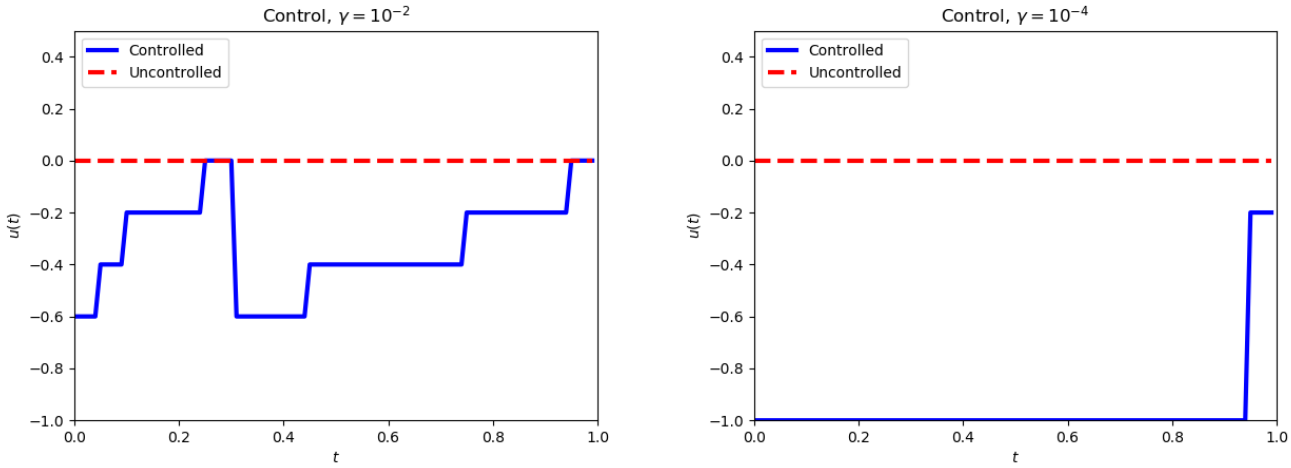

Figure 5.5: (Example 2): Controls for $\gamma = 10^{-2}$ (left) and $\gamma = 10^{-4}$ (right) with dimension $d = 10^3$ obtained by Algorithm 5.2 with 6 layers, 40 neurons per layer and $M = 2000$.

obtained here are worst and less stable than those obtained in chapter 4. Nevertheless, the advantage of the DP method here is the ability of solving this problem for different initial conditions y_0 in contrary to the *Optimistic Planning* approach that computes the solution for only one fixed initial condition. On the other hand, the DP and the PDE approaches are unable to solve this example after adding some constraints on the system state. Indeed, we have considered a constraint of the form $y^a(t, x) \geq \theta y_0(x)$, $\forall x \in [0, 1]$, for $\theta \in]0, 1[$ as we did in section 4.6, and we have obtained a non-negative approximation of the corresponding auxiliary value function w for any value of the auxiliary variable z which means that both approaches cannot compute admissible trajectories for this case.

Front propagation

Consider a first example of front propagation (Example 3), in a dimension $d \in \mathbb{N}^*$, where the dynamics f is given by $f(x, a) := a$ with $a \in A := \partial\mathbb{B}_{\mathbb{R}^d}$ (recall that $\mathbb{B}_{\mathbb{R}^d}$ denotes the closed unit ball of $\mathbb{R}^d$ and $\partial\mathbb{B}_{\mathbb{R}^d}$ denotes its boundary) and the initial condition is given by:

$$\Phi(x) = \min(\epsilon, \|x - A_0\| - r_0, \|x - B_0\| - r_0),$$

with $\epsilon = 1$, $r_0 = 0.5$, $A_0 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ and $B_0 = -A_0$. The corresponding PDE equation (Eikonal equation), for $T > 0$, is defined as:

$$\begin{cases} -\partial_t v(t,x) + \|D_x v(t,x)\| = 0, & t \in [0,T],\ x \in \mathbb{R}^d \\ v(T,x) = \Phi(x), & x \in \mathbb{R}^d \end{cases}$$

and its solution is given by:

$$v(t,x) := \min(\epsilon, \max(0, \|x - A_0\| - T + t) - r_0, \max(0, \|x - B_0\| - T + t) - r_0).$$

The computational domain here is $\Omega = [-2,2]^d$. For the DP approach, we used $N = 10$ time steps for the time-discretization of $[0,T] = [0,0.8]$, $n_A = 10^2$ points when $d = 2$ and $n_A = 10^4$ when $d = 6$, for the generation of the control grid A_g from the control set A.

Again, we observe, in the different tables, that the precision of the approximation can be improved, in general, by increasing simultaneously the number of hidden layers, the number of neurons and the size of the training set. Furthermore, we remark that the PDE approach is more accurate than the DP approach for $d = 2$ and even for $d = 6$. This result can be justified by the error that comes from the discretization of the control set A, used in (5.14), which is of dimension equal to d. On the other hand, both approaches failed to solve this problem for higher dimensions ($d \geq 8$).

Training parameters			$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M				
2	10	500	7.583 e-02	8.644 e-02	1.916 e-01	5.40
4	20	10^3	4.273 e-02	4.968 e-02	1.275 e-01	22.31
6	40	2×10^3	4.511 e-02	5.184 e-02	1.307 e-01	68.05

Table 5.7: (Example 3): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solutions obtained by Algorithm 5.2 corresponding to the dimension $d = 2$ with different neural network architectures and different values of M.

Training parameters				$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M	M_0				
2	10	10^3	100	4.415 e-02	5.838 e-02	1.190 e-01	4.55
4	20	2×10^3	200	1.526 e-02	1.752 e-02	9.865 e-01	16.92
6	40	4×10^3	400	1.472 e-02	1.705 e-02	9.388 e-01	73.18

Table 5.8: (Example 3): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solutions obtained by the PDE approach corresponding to the dimension $d = 2$ with different neural network architectures and different values of M and M_0.

Training parameters			$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M				
2	10	10^4	7.593 e-01	9.289 e-01	1.944 e00	483.61
4	20	2×10^4	2.617 e-01	4.009 e-01	1.366 e00	754.20
6	40	4×10^4	2.801 e-01	4.173 e-01	1.391 e00	1203.51

Table 5.9: (Example 3): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solutions obtained by Algorithm 5.2 for the dimension $d = 6$ with different neural network architectures and different values of M.

Training parameters				$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M	M_0				
2	10	2×10^4	2×10^3	3.754 e-01	4.130 e-01	1.500 e00	136.53
4	20	4×10^4	4×10^3	1.099 e-01	1.859 e-01	1.105 e00	329.08
6	40	8×10^4	8×10^3	9.840 e-02	1.245 e-01	9.954 e-01	918.28

Table 5.10: (Example 3): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solutions obtained by the PDE approach for the dimension $d = 6$ with different neural network architectures and different values of M and M_0.

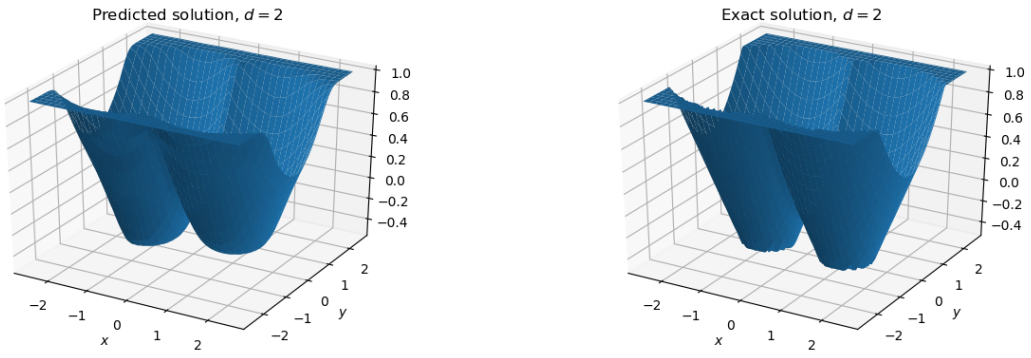

Figure 5.6: (Example 3): Exact and predicted value functions obtained by the PDE approach for $d = 2$ (6 layers, 40 neurons per layer, $M = 4 \times 10^3$ and $M_0 = 400$).

Now consider a second front propagation example (Example 4), in a dimension $d \in \mathbb{N}^*$, where the dynamics f is given by $f(x, a) = a \times \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^d$, with $a \in A = [0, 1]$, and the initial condition is given by $\Phi(x) = \min(\|x - A_0\| - r_0, \epsilon)$, where $\epsilon = 0.2$, $r_0 = 0.5$ and $A_0 = -\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^d$. The corresponding PDE equation, for $T > 0$, is defined as:

$$\begin{cases} -\partial_t v(t,x) + \max(0, \sum_{i=1}^d \partial_{x_i} v(t,x)) = 0, \quad t \in [0,T],\ x \in \mathbb{R}^d, \\ v(T,x) = \Phi(x), \quad x \in \mathbb{R}^d. \end{cases}$$

The computational domain here is $\Omega = [-2, 2]^d$. We used $N = 10$ time steps for the time-discretization

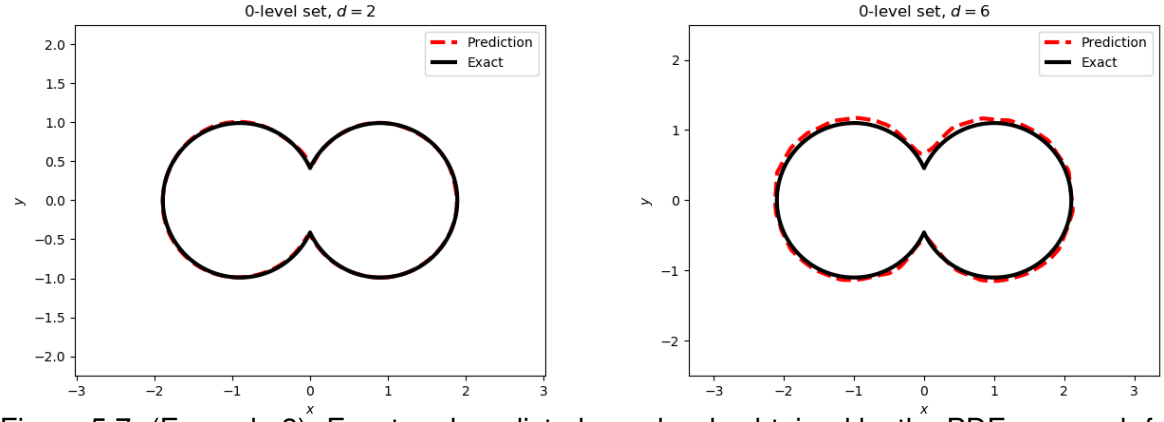

Figure 5.7: (Example 3): Exact and predicted zero levels obtained by the PDE approach for $d = 2$ (left, 6 layers, 40 neurons per layer, $M = 4 \times 10^3$ and $M_0 = 400$) and for $d = 6$ (right, 6 layers, 40 neurons per layer, $M = 8 \times 10^4$ and $M_0 = 8 \times 10^3$).

of $[0, T]$ and $n_A = 11$ points to generate the control grid A_g from the control set A for the DP approach. The relative errors presented in the following tables are computed with respect to the exact value function, which can be determined analytically[2], in a similar way to the first example on a uniform grid $\mathcal{G}$ on the computational domain Ω with size $N_{\mathcal{G}} = 10^{2d}$.

In tables 5.11, 5.13 and 5.15, we present the errors corresponding to different dimensions d obtained by the DP approach while tables 5.12, 5.14 and 5.16 corresponds to the PDE approach. We remark that increasing the size of the training set simultaneously with the number of hidden layers and the number of neurons per layer helps to ameliorate, in general, the accuracy of the approximated solution. Nevertheless, this leads to increasing the CPU time needed to train the neural networks.

Furthermore, for a low dimension ($d = 2$ for instance), we cannot decide which approach is better since their performances are comparable. This is not the case for higher dimensions ($d = 4$ and $d = 6$) where it is clear that the DP approach gives more accurate approximations. On the other hand, both approaches failed to solve this problem for higher dimensions ($d \geq 8$).

We represent in figure 5.8 the exact and the approximated solutions and the zero levels obtained by Algorithm 5.2 and corresponding to the dimension $d = 2$ while figure 5.9 contains the exact and approximated zero levels corresponding to dimensions $d = 4$ and $d = 6$.

[2]The exact solution for this example can be computed as follows. Define

$$A_t := A_0 + (T - t) \times \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \quad u := \frac{\overrightarrow{A_0 A_t}}{\|\overrightarrow{A_0 A_t}\|}, \quad h := \langle \overrightarrow{A_0 x}, u \rangle$$

$$h_P := \min\left(\max(h, 0), \|\overrightarrow{A_0 A_t}\|\right), \quad \text{and} \quad x_P := A_0 + h_P u,$$

where x_P is the projection of x on the segment $[A_0; A_t]$. Then the exact solution is given by

$$v(t, x) = \min(r_{0,t} - r_0, \epsilon), \quad \text{where} \quad r_{0,t} = \|x - x_P\|. \tag{5.20}$$

Training parameters			$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M				
2	10	500	4.31 e-01	4.96 e-01	1.43 e00	8.57
4	20	1000	7.34 e-02	9.38 e-02	3.94 e-01	13.25
6	40	2000	4.95 e-02	6.51 e-02	3.09 e-01	25.88
8	60	4000	3.72 e-02	6.42 e-02	2.93 e-01	60.52

Table 5.11: (Example 4): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solutions obtained by Algorithm 5.2 corresponding to the dimension $d = 2$ with different neural network architectures and different values of M.

Training parameters				$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M	M_0				
2	10	1250	100	4.07 e-02	4.42 e-02	1.32 e00	6.87
4	20	2500	200	6.82 e-02	8.91 e-02	3.76 e-02	10.45
6	40	5×10^3	400	5.12 e-02	6.89 e-02	3.27 e-01	48.04
8	60	10^4	800	5.58 e-02	7.07 e-02	3.49 e-01	175.41

Table 5.12: (Example 4): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solutions obtained by the PDE approach corresponding to the dimension $d = 2$ with different neural network architectures and different values of M and M_0.

Training parameters			$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M				
2	10	125×10^2	4.369 e-01	9.238 e-01	1.495 e00	21.66
4	20	25×10^3	3.045 e-01	6.389 e-01	9.763 e-01	90.79
6	40	5×10^4	6.871 e-02	8.174 e-02	3.501 e-01	367.40
8	60	10^5	6.919 e-02	8.359 e-02	3.690 e-01	935.66

Table 5.13: (Example 4): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solutions obtained by Algorithm 5.2 for the dimension $d = 4$ with different neural network architectures and different values of M.

Training parameters				$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M	M_0				
2	10	2×10^4	10^3	9.519 e-01	1.628 e00	1.897 e00	56.74
4	20	4×10^4	2×10^3	6.740 e-01	8.869 e-01	1.277 e00	216.79
6	40	8×10^4	4×10^3	9.461 e-02	1.195 e-01	7.221 e-01	894.56
8	60	16×10^4	8×10^3	9.284 e-02	1.032 e-01	7.019 e-01	2719.41

Table 5.14: (Example 4): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solutions obtained by the PDE approach for the dimension $d = 4$ with different neural network architectures and different values of M and M_0.

Example 5: Front propagation with state constraints

Now, consider a front propagation problem with an obstacle. Let $T = 1$, the dimension $d \in \mathbb{N}^*$, the dynamics f is given by:

$$f(x,a) = a \times \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^d, \quad \text{with} \quad a \in A = [0,1],$$

Training parameters			$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M				
2	10	375×10^2	4.762 e-01	9.617 e-01	1.305 e00	35.71
4	20	75×10^3	4.877 e-01	9.784 e-01	1.399 e00	105.44
6	40	15×10^4	8.110 e-02	8.926 e-02	5.773 e-01	506.39
8	60	3×10^5	7.651 e-02	8.737 e-02	5.642 e-01	3216.11

Table 5.15: (Example 4): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solutions obtained by Algorithm 5.2 for the dimension $d = 6$ with different neural network architectures and different values of M.

Training parameters				$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M	M_0				
2	10	5×10^4	25×10^2	1.014 e00	1.433 e00	1.605 e00	67.81
4	20	10^5	5×10^3	9.829 e-01	1.305 e00	1.553 e00	231.41
6	40	2×10^5	10^4	1.176 e-01	1.292 e-01	8.351 e-01	944.78
8	60	4×10^5	2×10^4	1.240 e-01	1.403 e-01	8.622 e-01	4117.73

Table 5.16: (Example 4): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solutions obtained by the PDE approach for the dimension $d = 6$ with different neural network architectures and different values of M and M_0.

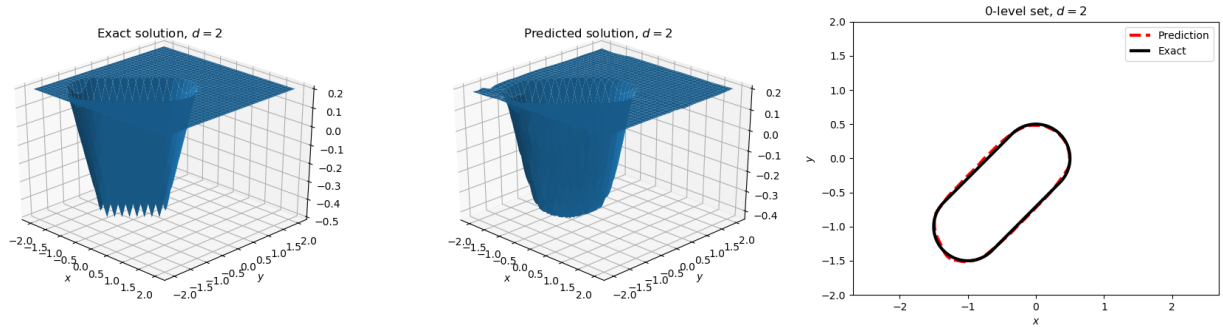

Figure 5.8: (Example 4): Exact solution (left), predicted solution (middle) and exact and predicted zero levels (right) obtained by Algorithm 5.2 corresponding to the dimension $d = 2$ with 8 layers, 60 neurons per layer and $M = 4000$.

and the initial condition is given by:

$$\Phi(x) = \min(\|x - A_0\| - r_0, \epsilon_0),$$

where $\epsilon_0 = 0.2$, $r_0 = 0.5$ and $A_0 = -\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^d$. The obstacle function, $g : \mathbb{R}^d \to \mathbb{R}$, is defined as follows:

$$g(x) = \max(-\epsilon_1, r_1 - \|x - B_0\|),$$

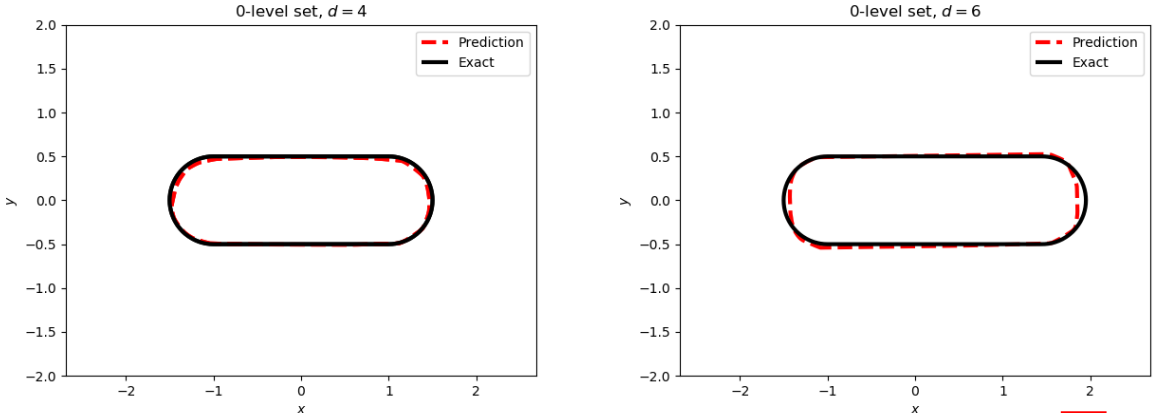

Figure 5.9: (Example 4): Exact and predicted zero levels obtained by Algorithm 5.2 for $d = 4$ (left, 6 layers and 40 neurons per layer, $M = 5 \times 10^4$) and for $d = 6$ (right, 8 layers and 60 neurons per layer, $M = 3 \times 10^5$).

where $\epsilon_1 = 0.2$, $r_1 = 0.5$ and $B_0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^d$. The corresponding PDE equation, for $T > 0$, is defined as:

$$\begin{cases} \min\Big(-\partial_t w(t,x) + \max(0, \sum_{i=1}^d \partial_{x_i} w(t,x)), w(t,x) - g(x)\Big) = 0, \quad t \in [0,T],\ x \in \mathbb{R}^d, \\ w(T,x) = \Phi(x) \bigvee g(x), \quad x \in \mathbb{R}^d. \end{cases}$$

The computational domain here is the same as in the previous example, $\Omega = [-2,2]^d$. We used $N = 10$ time steps for the time-discretization of $[0,T]$ and $n_A = 11$ points to generate the control grid A_g from the control set A. The relative errors presented in the following tables are computed with respect to the exact value function, which can be determined analytically[3], in a similar way to the first example on a uniform grid $\mathcal{G}$ on the computational domain Ω with size $N_{\mathcal{G}} = 10^{2d}$.

In tables 5.17, 5.19 and 5.21, we present the errors obtained by the DP approach and corresponding to different values of d while tables 5.18 and 5.20 contains the errors obtained by the PDE approach for dimensions $d = 2$ and $d = 4$ respectively. We represent in figure 5.10 the exact and the approximated solutions and the zero levels corresponding to the dimension $d = 2$ and in figure 5.11 the exact and approximated zero levels obtained by Algorithm 5.2 and corresponding to dimensions $d = 4$ and $d = 6$

[3]The exact solution is given by the following expression:

$$w(t,x) = \max(v(t,x), g_t(x)),$$

where $v(t,x)$ is the exact solution of the unconstrained case, defined in (5.20), and where

$$g_t(x) := \max_{\theta \in [0,t]} g(x + \theta \times A_0).$$

Furthermore $g_t(x)$ can be computed as follows:

$$B_t := B_0 - (T-t) \times A_0, \quad u := \frac{\overrightarrow{B_0 B_t}}{\|\overrightarrow{B_0 B_t}\|}, \quad h' := \langle \overrightarrow{B_0 x}, u \rangle$$
$$h'_P := \min(\max(h, 0), \|\overrightarrow{B_0 B_t}\|), \quad \text{and} \quad y_P := B_0 + h'_P u,$$

where y_P is the projection of x on the segment $[B_0; B_t]$. Henceforth, g_t can be expressed differently:

$$g_t(x) = \max(-\epsilon_1, r_1 - r_{1,t}), \quad \text{where} \quad r_{1,t} := \|x - y_P\|.$$

while figure 5.12 represents the exact and approximated zero levels and the evolution of the loss function, versus the number of iterations of the stochastic gradient algorithm used for the training of the neural networks, obtained by the PDE approach for $d = 4$.

Similarly to the previous examples, we remark that increasing the size of the training samples simultaneously with the number of hidden layers and the number of neurons per layer ameliorate, in general, the accuracy of the approximated solution and increases the CPU time needed to train the neural networks. Furthermore, we observe, in tables 5.17 and 5.18, the near performances of the different approaches for a small dimension ($d = 2$).

Training parameters			$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M				
2	10	500	3.741 e-01	4.408 e-01	1.035 e00	7.72
4	20	1000	7.822 e-02	1.082 e-01	4.352 e-01	16.32
6	40	2000	3.105 e-02	5.621 e-02	2.278 e-01	30.50
8	60	4000	2.948 e-02	4.370 e-02	1.967 e-01	62.32

Table 5.17: (Example 5): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solution obtained by Algorithm 5.2 corresponding to the dimension $d = 2$ with different neural network architectures and different values of M.

Training parameters				$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M	M_0				
2	10	1250	100	1.956 e-01	2.260 e-01	9.871 e-01	5.33
4	20	2500	200	7.045 e-02	8.493 e-02	3.972 e-01	14.06
6	40	5000	400	3.428 e-02	6.070 e-02	2.877 e-01	36.10
8	60	10^4	800	3.607 e-02	6.499 e-02	3.088 e-01	102.28

Table 5.18: (Example 5): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solution obtained by the PDE approach corresponding to the dimension $d = 2$ with different neural network architectures and different values of M and M_0.

Training parameters			$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M				
2	10	25×10^3	2.290 e-01	3.218 e-01	1.774 e00	36.87
4	20	5×10^4	7.743 e-02	1.416 e-01	6.978 e-01	111.02
6	40	10^5	4.846 e-02	5.098 e-02	1.018 e-01	667.14
8	60	2×10^5	4.520 e-02	4.701 e-02	9.521 e-02	2698.00

Table 5.19: (Example 5): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solution obtained by Algorithm 5.2 for the dimension $d = 4$ with different neural network architectures and different values of M.

Now, by comparing tables 5.19 and 5.20, we deduce again that the DP approach becomes more accurate than the PDE approach for higher dimensions ($d = 4$). Moreover, we observe in figure 5.11 that the DP approach is able to capture the front and the obstacle form for $d = 4$ and $d = 6$ which seems more difficult to handle with the PDE approach, see table 5.20 and figure 5.12. We remark also that the behaviour of the loss function, in figure 5.12, becomes of the form $C\alpha^n$, where n is the number of

Training parameters				$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M	M_0				
2	10	75×10^3	5×10^3	9.561 e-01	7.460 e-01	1.771 e00	369.30
4	20	15×10^4	10^4	9.394 e-01	7.393 e-01	1.649 e00	1180.41
6	40	3×10^5	2×10^4	9.683 e-01	7.711 e-01	1.854 e00	4319.05

Table 5.20: (Example 5): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solution obtained by the PDE approach for the dimension $d = 4$ with different neural network architectures and different values of M and M_0.

Training parameters			$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M				
2	10	375×10^2	1.580 e-01	2.209 e-01	8.166 e-01	109.19
4	20	75×10^3	1.737 e-01	2.385 e-01	8.406 e-01	351.43
6	40	15×10^4	8.097 e-02	1.497 e-01	7.101 e-01	900.71
8	60	3×10^5	5.206 e-02	6.161 e-02	1.919 e-01	2937.42

Table 5.21: (Example 5): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solution obtained by Algorithm 5.2 for the dimension $d = 6$ with different neural network architectures and different values of M.

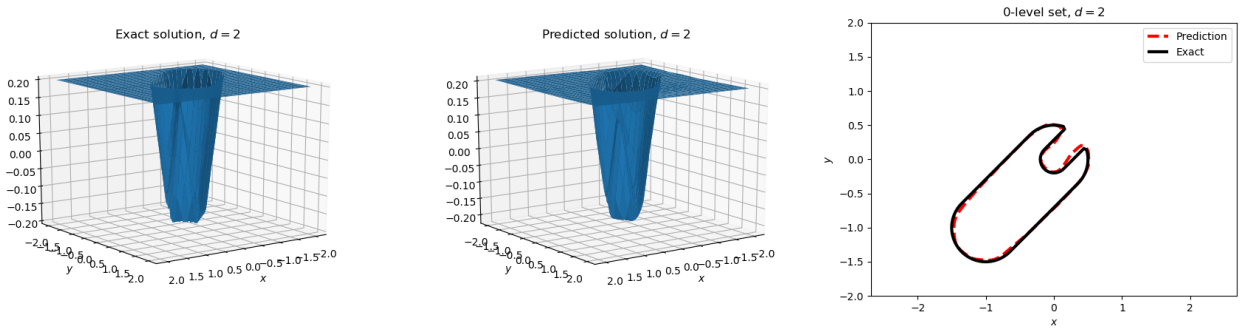

Figure 5.10: (Example 5): Exact solution (left), predicted solution (middle) and exact and predicted zero levels (right) obtained by Algorithm 5.2 corresponding to the dimension $d = 2$ with 8 layers, 60 neurons per layer and $M = 4000$.

iterations, $C > 0$ is a small real constant and $\alpha < 1$ is close to 1. Hence, the loss does not decrease enough, for high number of iterations, and reaches some threshold.

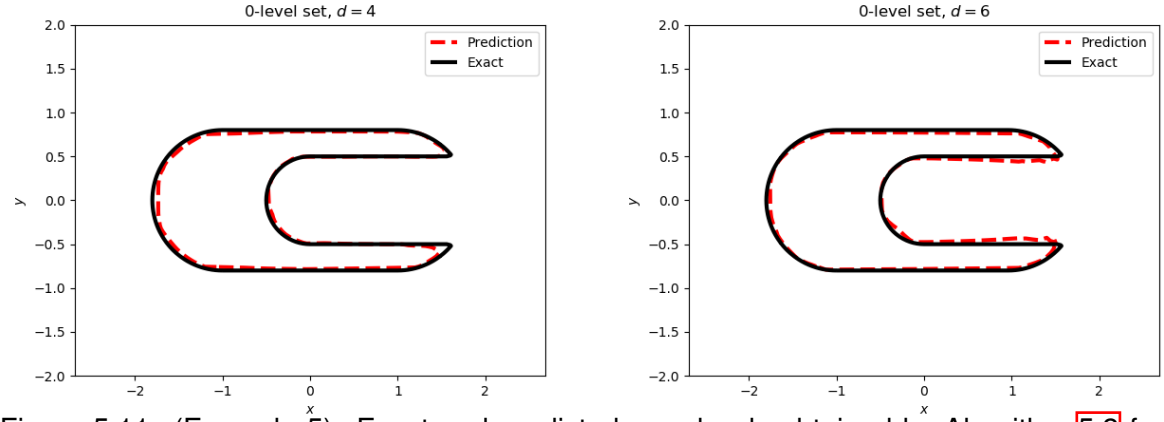

Figure 5.11: (Example 5): Exact and predicted zero levels obtained by Algorithm 5.2 for $d = 4$ (left, 6 layers, 40 neurons per layer, $M = 10^5$) and for $d = 6$ (right, 8 layers, 60 neurons per layer, $M = 3 \times 10^5$).

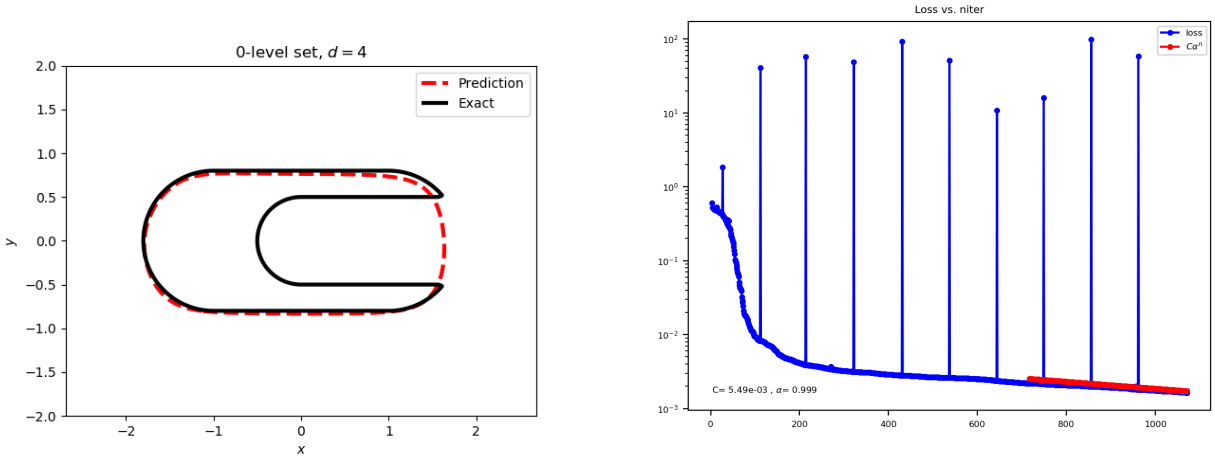

Figure 5.12: (Example 5): Exact and predicted zero levels obtained by the PDE approach (left) and the loss function evolution versus the number of iterations of the stochastic gradient algorithm (right) for $d = 4$ (6 layers, 40 neurons per layer, $M = 10^5$ and $M_0 = 5 \times 10^3$).

Example 6: An unconstrained game

Consider now a zero-sum differential game with finite time horizon $T = 1$ and state dimension $d = 1$. The dynamical system is given by

$$\dot{y}(s) = f(y(s), a(s), b(s)) = |a(s) - b(s)|, \quad s \in [0, T],$$

where the first and the second player controls $a(\cdot)$ and $b(\cdot)$ take values respectively in the control sets A and B with $A = B = [-1, 1]$. The distributed and the final cost functions are given by $\ell(x, a, b) = e^x$ and $\Phi(\cdot) = 0$. The value function, corresponding to the case where the first player tries to minimize the cost functional by using nonanticipative strategies $\alpha[\cdot] \in \Gamma$, is defined by (see chapters 2 and 3 for more definitions about differential games):

$$v(t, x) = \inf_{\alpha[\cdot] \in \Gamma} \sup_{b(\cdot) \in \mathcal{B}} \int_t^T e^{y_{t,x}^{\alpha[b],b}(s)} ds = (T - t)e^x.$$

This value function is the unique viscosity solution of the following HJI equation:

$$\begin{cases} -\partial_t v(t,x) + H(x, \partial_x v(t,x)) = 0, \quad t \in [0,T],\ x \in \mathbb{R}, \\ v(T,x) = \Phi(x), \quad x \in \mathbb{R}, \end{cases}$$

where the hamiltonian function is given, for any $(x,p) \in \mathbb{R} \times \mathbb{R}$, by:

$$H(x,p) = \min_{b \in B} \max_{a \in A} \Big\{ -pf(x,a,b) - \ell(x,a,b) \Big\} = \begin{cases} -p - e^x, & \text{if} \quad p \leq 0, \\ -e^x, & \text{else.} \end{cases}$$

The computational domain for this example is $\Omega = [-1,1]$. For the DP approach, we used $N = 20$ time steps for the time-discretization of $[0,T]$ and $n_A = n_B = 11$ points to generate the uniform control grids A_g and B_g from the control sets A and B respectively. The errors, presented in tables 5.22 and 5.23, are calculated at time $t = 0$ over a uniform grid $\mathcal{G}$ on the computational domain Ω with $N_{\mathcal{G}} = 5000$ points.

Similarly to the previous examples, increasing simultaneously the training samples size, the depth of the neural networks and the number of neurons per layer helps to ameliorate the precision of the approximations obtained by the DP and the PDE approaches, see tables 5.22 and 5.23. Moreover, we observe that the approximation obtained by the PDE approach is more accurate than the DP approximation which can be explained again by the error accumulation through the time steps for the DP approach.

Training parameters			$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M				
2	5	50	1.096 e-01	1.794 e-01	3.538 e-01	62.41
4	10	100	6.314 e-02	6.501 e-02	1.104 e-01	76.84
6	20	200	2.668 e-02	3.150 e-02	6.358 e-02	125.20
8	40	400	2.611 e-02	3.025 e-02	5.145 e-02	148.92

Table 5.22: (Example 6): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solution obtained by the DP approach with different network architectures and values of M.

Training parameters				$\mathcal{L}_1$ error	$\mathcal{L}_2$ error	$\mathcal{L}_\infty$ error	CPU(s)
Layers	Neurons	M	M_0				
2	5	125	10	3.963 e-02	4.056 e-02	1.357 e-01	22.80
4	10	250	20	1.197 e-02	2.072 e-02	9.408 e-02	36.94
6	20	5×10^2	40	9.159 e-03	9.846 e-03	7.408 e-02	71.07
8	40	10^3	80	7.923 e-03	8.025 e-03	7.010 e-02	109.47

Table 5.23: (Example 6): Relative $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ errors between the exact and the predicted solution obtained by the PDE approach with different network architectures and values of M_0 and M.

We represent in figure 5.13 the exact and the approximated value functions obtained by the DP and PDE approaches.

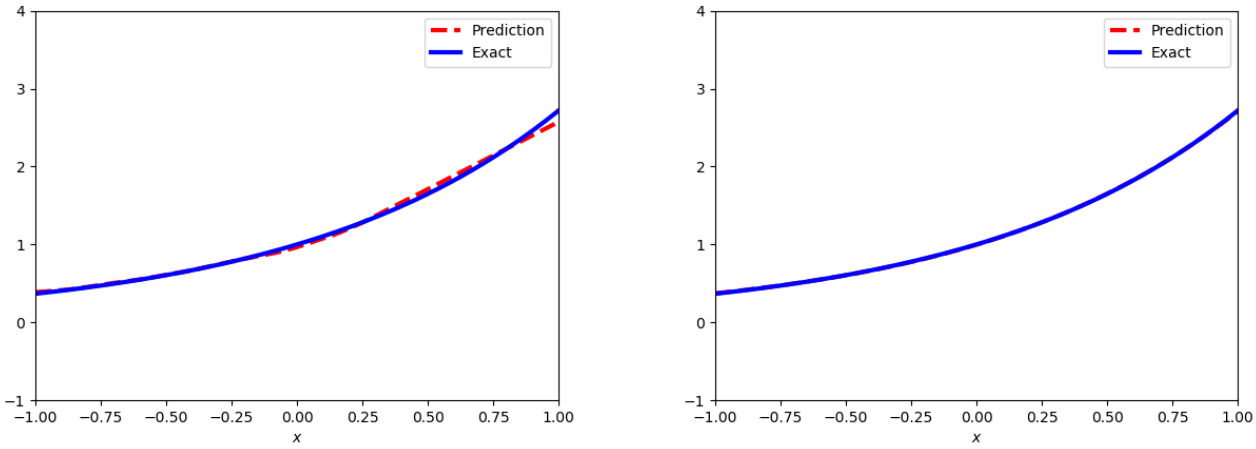

Figure 5.13: (Example 6): Exact and predicted value functions at $t = 0$, $v(0, \cdot)$ and $\hat{V}_0(\cdot)$, obtained by the DP approach (left, $M = 400$) and the PDE approach (right, $M = 10^3$ and $M_0 = 80$) with 8 layers and 40 neurons per layer.

Comparison between the DP and the PDE approaches

Throughout the numerical examples tested in this chapter, we have obtained the following observations:

- First, the numerical performances of both approaches, DP and PDE, can be improved, in general, by increasing simultaneously the number of hidden layers, the number of neurons and the size of the training set.
- The PDE approach is more accurate than the DP approach when either the state dimension is low or the control set dimension is high (see examples 1, 3 and 6). Nevertheless, the PDE approach seems unable to handle problems with obstacle terms in high state dimension (example 5 with $d = 4$).
- The DP approach becomes more accurate than the PDE approach for higher values of the state dimension d (see examples 2, 4 and 5).
- Both approaches can be extended to deal with zero-sum differential games, see example 6.
- For the moment and with the available tools in the Tensorflow library (tensors, neural networks architectures, algorithms for machine learning optimization...), our approaches are unable to solve other state-constrained problems, such as Zermelo, optimal control of the heat equation and windshear problems. Indeed, we have obtained non-negative approximations of the auxiliary value function w, for any value of the auxiliary variable z, which means that both approaches cannot compute admissible trajectories. Recall that those problems are solved, in chapter 4, by use of the *Optimistic Planning* approach. Furthermore, the DP and the PDE approaches failed to solve the different front propagation problems (examples 3, 4 and 5) for higher dimensions ($d \geq 8$).

Chapter 6

Conclusion and perspectives

In this work, we apply the Hamilton-Jacobi approach to study state-constrained two-person zero-sum differential games. Moreover, we provide optimistic planning and deep learning algorithms to solve optimal control problems involving state constraints.

In chapter 3, we study a two-person zero-sum differential game under state constraints and where the controls of the two players are coupled within the dynamics, the cost functions and the state constraints. In particular, we show that the original state-constrained problem can be characterized by means of an auxiliary differential game free of state constraints. Furthermore, we propose a reconstruction procedure to approximate optimal strategies and controls of both players for the auxiliary game and hence for the constrained problem.

In chapter 4, we propose optimistic planning algorithms to solve state-constrained finite-horizon nonlinear optimal control problems. Thanks to the level set approach from [5], it is known that the value function of such problem can be characterized by use of an unconstrained auxiliary problem. In order to compute an approximation of the auxiliary value function, we have adapted optimistic planning methods proposed in [39, 38, 35, 75, 76] to the finite time horizon and extended them to our case with maximum cost functional. Furthermore, we have established theoretical convergence results of those algorithms. Finally, we have designed another algorithm with better performance by exploiting some ideas from model predictive control theory.

A possible future direction of research, from chapters 3 and 4, will be to adapt the optimistic planning approach to solve state-constrained differential games in finite horizon. We refer to [37] where a game with two players having opposite interests and taking decisions in turn was studied without constraints on the system state and with infinite time horizon. As we have seen in section 3.3, the constrained problem can be characterized through a differential game free of state constraints having the following form (for simplicity, let $t = 0$):

$$w(0,x,z) := \inf_{\alpha[\cdot]\in\Gamma} \sup_{b(\cdot)\in\mathcal{B}} \left\{ \left(\max_{s\in[0,T]} \Phi(\hat{y}_{\hat{x}}^{\alpha[b],b}(s)) \right) \bigvee \Psi(\hat{y}_{\hat{x}}^{\alpha[b],b}(T)) \right\}$$

where $T > 0$, $\hat{x} = (x,z) \in \mathbb{R}^d \times \mathbb{R}$, Φ and Ψ are the cost functions and $\hat{y}_{\hat{x}}^{\alpha[b],b}(\cdot)$ is the trajectory associated to the augmented dynamical system. From section 3.4, w can be approximated by the following discrete time differential game:

$$w_h(x,z) := \inf_{\alpha_h[\cdot]\in\Gamma_h} \sup_{(\mathsf{b}_i)_i\in B^N} \left(\max_{i=0,\dots,N} \Phi(\hat{\mathsf{y}}_i^{\alpha_h[\mathsf{b}],\mathsf{b}}) \right) \bigvee \Psi(\hat{\mathsf{y}}_N^{\alpha_h[\mathsf{b}],\mathsf{b}}),$$

where $h = \frac{T}{N}$ with $N \in \mathbb{N}^*$, $\alpha_h[\cdot] \in \Gamma_h$ is a discrete nonanticipative strategy of the first player, $(\mathsf{b}_i)_i \in B^N$ is a control sequence of the second player and $(\hat{\mathsf{y}}_i^{\alpha_h[\mathsf{b}],\mathsf{b}})_{i=0}^N$ is the discrete trajectory approximating

the continuous time trajectory $\hat{y}_{\hat{x}}^{\alpha[b],b}(\cdot)$. Moreover, from the definition of nonanticipative discrete time strategies, the advantage of information attributed to the first player means that the latter makes his choice $\mathsf{a}_i \in A$, for $i = 0, ..., N-1$, after observing his opponent action $\mathsf{b}_i \in B$. Therefore, w_h can be rewritten as follows:

$$w_h(\hat{x}) = \max_{\mathsf{b}_0} \min_{\mathsf{a}_0} \cdots \max_{\mathsf{b}_{N-1}} \min_{\mathsf{a}_{N-1}} \left(\max_{i=0,...,N} \Phi(\hat{\mathsf{y}}_i^{\mathsf{a},\mathsf{b}}) \right) \bigvee \Psi(\hat{\mathsf{y}}_N^{\mathsf{a},\mathsf{b}}), \tag{6.1}$$

with $\mathsf{a} := (\mathsf{a}_0, ..., \mathsf{a}_{N-1}) \in A^N$ and $\mathsf{b} := (\mathsf{b}_0, ..., \mathsf{b}_{N-1}) \in B^N$. The Formulation (6.1) is similar to the minimax problem considered in [37] for the case of an infinite horizon sum with a positive discount factor and under a boundedness assumption on the instantaneous reward. In our context, the cost functional, that the first player wants to minimize while the second player tries to maximize, is defined in finite horizon by taking the maximum of several terms depending on the actions of the two players.

The main question now is how to adapt the optimistic search algorithm for minimax sequential decisions, introduced in [37], and to propose, in a similar way to chapter 4, optimistic planning algorithms to solve problem (6.1). Furthermore, one can provide theoretical convergence results that relate the near-optimality of the returned solution with the computational budget allowed. Recall that the advantage of the optimistic planning approach is to remove the direct dependence between the resolution complexity and the dimension of the system state. Nevertheless, we have seen, in chapter 4, that the complexity of such numerical methods depends on the resolution horizon N (the number of time steps or the number of actions to be applied) and also on the control dimension $q \in \mathbb{N}^*$. For this reason, it will be very interesting to propose improvements of the existing algorithms in such a way to alleviate the complexity of those methods with respect to N and q.

Another practical contribution in the future can be by parallelizing the C++ code of optimistic planning algorithms with respect to the auxiliary variable z. In such a way, one can execute computations to approximate the auxiliary value function for several values of z in the same time which reduces the global computational time.

Finally, in chapter 5 we investigate numerical methods based on deep learning for constrained deterministic optimal control problems. In particular, we propose two different approaches. The first algorithm exploits the dynamic programming principle while the aim of the second approach is to approximate the solutions of Hamilton-Jacobi equations. Those approaches can be extended to handle two-person zero-sum differential games.

In this subject, there are two broad directions of research. First, it will be very interesting to prove theoretical convergence results of the proposed approaches at least when using neural networks with a single hidden layer. We mention that such results has been proven in [78, 77] for deep learning algorithms, based on the dynamic programming principle, when studying stochastic optimal control problems free of state constraints. The second direction concerns the amelioration of the precision and the complexity of our approaches. This can be done essentially by either modifying the architecture design of the parametric functions (the neural networks) or ameliorating the precision of the stochastic optimization process (the training).

Bibliography

[1] M. Akian, S. Gaubert, and A. Lakhoua. The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis. *SIAM Journal on Control and Optimization*, 47(2):817–848, 2008.

[2] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

[3] A. Alla, M. Falcone, and L. Saluzzi. An efficient DP algorithm on a tree-structure for finite horizon optimal control problems. *SIAM Journal on Scientific Computing*, 41(4):A2384–A2406, 2019.

[4] A. Alla, M. Falcone, and L. Saluzzi. A tree structure algorithm for optimal control problems with state constraints. *arXiv preprint arXiv:2009.12384*, 2020.

[5] A. Altarovici, O. Bokanowski, and H. Zidani. A general Hamilton-Jacobi framework for non-linear state-constrained control problems. *ESAIM: Control, Optimisation and Calculus of Variations*, 19(2):337–357, 2013.

[6] M. Assellaou, O. Bokanowski, A. Desilles, and H. Zidani. Value function and optimal trajectories for a maximum running cost control problem with state constraints. application to an abort landing problem. *ESAIM: Mathematical Modelling and Numerical Analysis*, 52(1):305–335, 2018.

[7] J.-P. Aubin and A. Cellina. *Differential inclusions: set-valued maps and viability theory*, volume 264. Springer Science & Business Media, 2012.

[8] J.-P. Aubin and H. Frankowska. *Set-valued analysis*. Springer Science & Business Media, 2009.

[9] R. J. Aumann, M. Maschler, and R. E. Stearns. *Repeated games with incomplete information*. MIT press, 1995.

[10] A. Bachouch, C. Huré, N. Langrené, and H. Pham. Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications. *arXiv preprint arXiv:1812.05916*, 2018.

[11] M. Bardi and I. Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer Science & Business Media, 2008.

[12] M. Bardi, S. Koike, and P. Soravia. Pursuit-evasion games with state constraints: dynamic programming and discrete-time approximations. *Discrete & Continuous Dynamical Systems-A*, 6(2):361, 2000.

[13] M. Bardi and P. Soravia. A PDE framework for games of pursuit-evasion type. In *Differential games and applications*, pages 62–71. Springer, 1989.

[14] G. Barles and P. E. Souganidis. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptotic analysis*, 4(3):271–283, 1991.

[15] E. Barron. Differential games with maximum cost. *Nonlinear analysis: Theory, methods & applications*, 14(11):971–989, 1990.

[16] T. Başar and P. Bernhard. *H_∞ optimal control and related minimax design problems: a dynamic game approach*. Springer Science & Business Media, 2008.

[17] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, 18(1):5595–5637, 2017.

[18] R. Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716, 1952.

[19] R. Bellman and R. E. Kalaba. *Dynamic programming and modern control theory*, volume 81. Citeseer, 1965.

[20] R. E. Bellman and S. E. Dreyfus. *Applied dynamic programming*. Princeton university press, 2015.

[21] D. P. Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific Belmont, MA, 2019.

[22] D. P. Bertsekas and J. N. Tsitsiklis. Neuro-dynamic programming: an overview. In *Proceedings of 1995 34th IEEE Conference on Decision and Control*, volume 1, pages 560–564. IEEE, 1995.

[23] P. Bettiol, P. Cardaliaguet, and M. Quincampoix. Zero-sum state constrained differential games: existence of value for Bolza problem. *International Journal of Game Theory*, 34(4):495–527, 2006.

[24] P. Bettiol, M. Quincampoix, and R. Vinter. Existence and characterization of the values of two player differential games with state constraints. *Applied Mathematics & Optimization*, 80(3):765–799, 2019.

[25] O. Bokanowski, N. Forcadel, and H. Zidani. Reachability and minimal times for state constrained nonlinear problems without any controllability assumption. *SIAM Journal on Control and Optimization*, 48(7):4292–4316, 2010.

[26] O. Bokanowski, N. Forcadel, and H. Zidani. Deterministic state-constrained optimal control problems without controllability assumptions. *ESAIM: Control, Optimisation and Calculus of Variations*, 17(4):995–1015, 2011.

[27] O. Bokanowski, J. Garcke, M. Griebel, and I. Klompmaker. An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton-Jacobi Bellman equations. *Journal of Scientific Computing*, 55(3):575–605, 2013.

[28] V. Boltyanskiy, R. V. Gamkrelidze, Y. MISHCHENKO, and L. Pontryagin. Mathematical theory of optimal processes. 1962.

[29] N. Botkin and V. Turova. Dynamic programming approach to aircraft control in a windshear. In *Advances in dynamic games*, pages 53–69. Springer, 2013.

[30] L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

[31] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[32] T. Bqar and G. J. Olsder. Dynamic noncooperative game theory, 1982.

[33] R. Bulirsch, F. Montrone, and H. J. Pesch. Abort landing in the presence of windshear as a minimax optimal control problem, part 1: Necessary conditions. *Journal of Optimization Theory and Applications*, 70(1):1–23, 1991.

[34] R. Bulirsch, F. Montrone, and H. J. Pesch. Abort landing in the presence of windshear as a minimax optimal control problem, part 2: Multiple shooting and homotopy. *Journal of Optimization Theory and Applications*, 70(2):223–254, 1991.

[35] L. Buşoniu, A. Daniels, R. Munos, and R. Babuška. Optimistic planning for continuous-action deterministic systems. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 69–76. IEEE, 2013.

[36] L. Busoniu and R. Munos. Optimistic planning for markov decision processes. In *Artificial Intelligence and Statistics*, pages 182–189, 2012.

[37] L. Buşoniu, R. Munos, and E. Páll. An analysis of optimistic, best-first search for minimax sequential decision making. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 1–8. IEEE, 2014.

[38] L. Buşoniu, E. Páll, and R. Munos. Discounted near-optimal control of general continuous-action nonlinear systems using optimistic planning. In *2016 American Control Conference (ACC)*, pages 203–208. IEEE, 2016.

[39] L. Buşoniu, E. Páll, and R. Munos. Continuous-action planning for discounted infinite-horizon nonlinear optimal control with lipschitz values. *Automatica*, 92:100–108, 2018.

[40] P. Cardaliaguet. A differential game with two players and one target: The continuous case. 1994.

[41] P. Cardaliaguet. A differential game with two players and one target. *SIAM Journal on Control and Optimization*, 34(4):1441–1460, 1996.

[42] P. Cardaliaguet. Nonsmooth semipermeable barriers, Isaacs' equation, and application to a differential game with one target and two players. *Applied Mathematics and Optimization*, 36(2):125–146, 1997.

[43] P. Cardaliaguet. Differential games with asymmetric information. *SIAM journal on Control and Optimization*, 46(3):816–838, 2007.

[44] P. Cardaliaguet. Representations formulas for some differential games with asymmetric information. *Journal of optimization theory and applications*, 138(1):1, 2008.

[45] P. Cardaliaguet. *Introduction to differential games*, volume 126. Université de Brest, 2010.

[46] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Pursuit differential games with state constraints. *SIAM Journal on Control and Optimization*, 39(5):1615–1632, 2000.

[47] Q. Chan-Wai-Nam, J. Mikael, and X. Warin. Machine learning for semi linear PDEs. *Journal of Scientific Computing*, 79(3):1667–1712, 2019.

[48] X. Chen, S. Liu, R. Sun, and M. Hong. On the convergence of a class of ADAM-type algorithms for non-convex optimization. *arXiv preprint arXiv:1808.02941*, 2018.

[49] J. Clark and R. Vinter. On the interpretation of non-anticipative control strategies in differential games and applications to flow control. In *Optimal Control, Stabilization and Nonsmooth Analysis*, pages 29–47. Springer, 2004.

[50] M. G. Crandall and P.-L. Lions. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American mathematical society*, 277(1):1–42, 1983.

[51] M. G. Crandall and P.-L. Lions. Two approximations of solutions of Hamilton-Jacobi equations. *Mathematics of computation*, 43(167):1–19, 1984.

[52] G. Cybenko. Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:183–192, 1989.

[53] J. Darbon, G. P. Langlois, and T. Meng. Overcoming the curse of dimensionality for some Hamilton–Jacobi partial differential equations via neural network architectures. *Research in the Mathematical Sciences*, 7(3):1–50, 2020.

[54] S. De, A. Mukherjee, and E. Ullah. Convergence guarantees for RMSProp and ADAM in non-convex optimization and an empirical comparison to Nesterov acceleration. *arXiv preprint arXiv:1807.06766*, 2018.

[55] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[56] R. Elliott and N. Kalton. *The existence of value in differential games*, volume 126. American Mathematical Soc., 1972.

[57] R. Elliott and N. Kalton. Values in differential games. *Bulletin of the American Mathematical Society*, 78(3):427–431, 1972.

[58] R. J. Elliott. Feedback strategies in deterministic differential games. In *Differential Games and Applications*, pages 136–142. Springer, 1977.

[59] R. J. Elliott and N. J. Kalton. Cauchy problems for certain Isaacs-Bellman equations and games of survival. *Transactions of the American Mathematical Society*, 198:45–72, 1974.

[60] L. C. Evans and P. E. Souganidis. Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations. *Indiana University mathematics journal*, 33(5):773–797, 1984.

[61] M. Falcone. Numerical solution of dynamic programming equations. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman equations. Birkhäuser*, 3:2, 1997.

[62] M. Falcone. Numerical methods for differential games based on partial differential equations. *International Game Theory Review*, 8(02):231–272, 2006.

[63] M. Falcone and R. Ferretti. *Semi-Lagrangian approximation schemes for linear and Hamilton—Jacobi equations*. SIAM, 2013.

[64] M. Falcone and T. Giorgi. An approximation scheme for evolutive Hamilton-Jacobi equations. In *Stochastic analysis, control, optimization and applications*, pages 289–303. Springer, 1999.

[65] M. Falcone, P. Lanucara, and A. Seghini. A splitting algorithm for Hamilton-Jacobi-Bellman equations. *Applied Numerical Mathematics*, 15(2):207–218, 1994.

[66] W. H. Fleming and W. M. McEneaney. Risk sensitive optimal control and differential games. In *Stochastic theory and adaptive control*, pages 185–197. Springer, 1992.

[67] W. H. Fleming and P. E. Souganidis. On the existence of value functions of two-player, zero-sum stochastic differential games. *Indiana University Mathematics Journal*, 38(2):293–314, 1989.

[68] H. Frankowska and S. Plaskacz. Semicontinuous solutions of Hamilton–Jacobi–Bellman equations with degenerate state constraints. *Journal of mathematical analysis and applications*, 251(2):818–838, 2000.

[69] H. Frankowska, S. Plaskacz, and T. Rzezuchowski. Measurable viability theorems and the Hamilton-Jacobi-Bellman equation. *Journal of Differential Equations*, 116(2):265–305, 1995.

[70] H. Frankowska and R. Vinter. Existence of neighboring feasible trajectories: applications to dynamic programming for state-constrained optimal control problems. *Journal of Optimization Theory and Applications*, 104(1):20–40, 2000.

[71] Y. Gu, H. Yang, and C. Zhou. Selectnet: Self-paced learning for high-dimensional partial differential equations. *arXiv preprint arXiv:2001.04860*, 2020.

[72] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[73] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560, 1990.

[74] K. Hornik, M. Stinchcombe, H. White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[75] J.-F. Hren. *Planification optimiste pour systemes déterministes*. PhD thesis, 2012.

[76] J.-F. Hren and R. Munos. Optimistic planning of deterministic systems. In *European Workshop on Reinforcement Learning*, pages 151–164. Springer, 2008.

[77] C. Huré. *Numerical methods and deep learning for stochastic control problems and partial differential equations*. PhD thesis, 2019.

[78] C. Huré, H. Pham, A. Bachouch, and N. Langrené. Deep neural networks algorithms for stochastic control problems on finite horizon, part i: convergence analysis. *arXiv preprint arXiv:1812.04300*, 2018.

[79] C. Huré, H. Pham, and X. Warin. Some machine learning schemes for high-dimensional nonlinear PDEs. *arXiv preprint arXiv:1902.01599*, 2019.

[80] C. Huré, H. Pham, and X. Warin. Deep backward schemes for high-dimensional nonlinear PDEs. *Mathematics of Computation*, 89(324):1547–1579, 2020.

[81] R. Isaacs. *Differential Games*. 1965.

[82] R. Isaacs. *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1999.

[83] H. Ishii. A boundary value problem of the dirichlet type for Hamilton-Jacobi equations. *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze*, 16(1):105–135, 1989.

[84] H. Ishii. On uniqueness and existence of viscosity solutions of fully nonlinear second-order elliptic PDE's. *Communications on pure and applied mathematics*, 42(1):15–45, 1989.

[85] H. Ishii and S. Koike. A new formulation of state constraint problems for first-order PDEs. *SIAM Journal on Control and Optimization*, 34(2):554–571, 1996.

[86] H. Ishii and P.-L. Lions. Viscosity solutions of fully nonlinear second-order elliptic partial differential equations. *Journal of Differential equations*, 83(1):26–78, 1990.

[87] M. R. James. Asymptotic analysis of nonlinear stochastic risk-sensitive control and differential games. *Mathematics of Control, Signals and Systems*, 5(4):401–417, 1992.

[88] D. P. K. JLB. Adam: A method for stochastic optimization. In *3rd international conference for learning representations, San Diego*, 2015.

[89] A. Kleimenov and N. P. D. Igry. Nonantagonistic positional differential games. *Science, Ekaterinburg*, 1993.

[90] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[91] N. V. Krylov. On the rate of convergence of finite-difference approximations for Bellman's equations. , 9(3):245–256, 1997.

[92] H. J. Kushner and P. Dupuis. *Numerical methods for stochastic control problems in continuous time*, volume 24. Springer Science & Business Media, 2001.

[93] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[94] G. Lars and P. Jürgen. *Nonlinear model predictive control theory and algorithms*. Springer, 2011.

[95] J. B. Lasserre, D. Henrion, C. Prieur, and E. Trélat. Nonlinear optimal control via occupation measures and LMI-relaxations. *SIAM journal on control and optimization*, 47(4):1643–1666, 2008.

[96] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[97] X. Li and F. Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 983–992. PMLR, 2019.

[98] S. Liang and R. Srikant. Why deep neural networks for function approximation ? *arXiv preprint arXiv:1610.04161*, 2016.

[99] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pages 6231–6239, 2017.

[100] C. R. Mansley, A. Weinstein, and M. L. Littman. Sample-Based planning for continuous action Markov decision processes. In *ICAPS*, 2011.

[101] K. Máthé, L. Buçoniu, R. Munos, and B. De Schutter. Optimistic planning with a limited number of action switches for near-optimal nonlinear control. In *53rd IEEE Conference on Decision and Control*, pages 3518–3523. IEEE, 2014.

[102] R. C. Merton. Optimum consumption and portfolio rules in a continuous-time model. In *Stochastic Optimization Models in Finance*, pages 621–661. Elsevier, 1975.

[103] A. Miele, T. Wang, and W. Melvin. Quasi-steady flight to quasi-steady flight transition for abort landing in a windshear: trajectory optimization and guidance. *Journal of Optimization Theory and Applications*, 58(2):165–207, 1988.

[104] A. Miele, T. Wang, C. Tzeng, and W. Melvin. Optimal abort landing trajectories in the presence of windshear. *Journal of Optimization Theory and Applications*, 55(2):165–202, 1987.

[105] R. Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in neural information processing systems*, pages 783–791, 2011.

[106] R. Munos. From bandits to Monte-Carlo Tree Search: The optimistic principle applied to optimization and planning. 2014.

[107] T. Nakamura-Zimmerer, Q. Gong, and W. Kang. Adaptive deep learning for high dimensional Hamilton-Jacobi-Bellman equations. *arXiv preprint arXiv:1907.05317*, 2019.

[108] S. Osher and C.-W. Shu. High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations. *SIAM Journal on numerical analysis*, 28(4):907–922, 1991.

[109] L. S. Pontryagin. On the theory of differential games. *RuMaS*, 21(4):193–246, 1966.

[110] A. Quarteroni and A. Valli. *Domain decomposition methods for partial differential equations*. Oxford University Press, 1999.

[111] M. Quincampoix and O. S. Serea. A viability approach for optimal control with infimum cost. *Annals. Stiint. Univ. Al. I. Cuza Iasi, sI a, Mat*, 48:113–132, 2002.

[112] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[113] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics informed deep learning (part II): Data-driven discovery of nonlinear partial differential equations, arxiv preprint (2017). *arXiv preprint arXiv:1711.10566*.

[114] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part I): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[115] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[116] E. A. Rapaport. Characterization of barriers of differential games. *Journal of optimization theory and applications*, 97(1):151–179, 1998.

[117] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[118] J. Rowland and R. Vinter. Construction of optimal feedback controls. *Systems & control letters*, 16(5):357–367, 1991.

[119] E. Roxin. Axiomatic approach in differential games. *Journal of Optimization Theory and Applications*, 3(3):153–163, 1969.

[120] L. Saluzzi, A. Alla, and M. Falcone. Error estimates for a tree structure algorithm solving finite horizon control problems. *arXiv preprint arXiv:1812.11194*, 2018.

[121] O. S. Serea. Discontinuous differential games and control systems with supremum cost. *Journal of mathematical analysis and applications*, 270(2):519–542, 2002.

[122] J. A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms. *SIAM Journal on Numerical Analysis*, 41(1):325–363, 2003.

[123] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[124] H. M. Soner. Optimal control with state-space constraint I. *SIAM Journal on Control and Optimization*, 24(3):552–561, 1986.

[125] H. M. Soner. Optimal control with state-space constraint. II. *SIAM journal on control and optimization*, 24(6):1110–1122, 1986.

[126] P. Soravia. H_∞ control of nonlinear systems: Differential games and viscosity solutions. *SIAM Journal on control and optimization*, 34(3):1071–1097, 1996.

[127] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[128] R. van der Meer, C. Oosterlee, and A. Borovykh. Optimally weighted loss functions for solving PDEs with neural networks. *arXiv preprint arXiv:2002.06269*, 2020.

[129] P. P. Varaiya. On the existence of solutions to a differential game. *SIAM Journal on Control*, 5(1):153–162, 1967.

[130] R. Ward, X. Wu, and L. Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pages 6677–6686, 2019.

[131] A. Weinstein and M. L. Littman. Bandit-based planning and learning in continuous-action markov decision processes. In *Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling*, pages 306–314, 2012.

[132] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in neural information processing systems*, pages 4148–4158, 2017.

[133] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar. Adaptive methods for nonconvex optimization. In *Advances in neural information processing systems*, pages 9793–9803, 2018.

[134] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[135] D. Zhou, Y. Tang, Z. Yang, Y. Cao, and Q. Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv preprint arXiv:1808.05671*, 2018.

ECOLE
DOCTORALE
DE MATHEMATIQUES
HADAMARD

Titre: Approche Hamilton-Jacobi pour les jeux différentiels avec des contraintes d'état et méthodes numériques d'apprentissage pour des problèmes de commande optimale

Mots clés: Commande optimale, Jeux différentiels, Contraintes d'état, Planification optimiste, Apprentissage profond

Résumé: L'objectif de cette thèse est d'étudier des jeux différentiels avec contraintes d'état par l'approche Hamilton-Jacobi et de développer des méthodes numériques d'apprentissage pour résoudre des problèmes de commande optimale. La première partie considère un jeu différentiel à somme nulle et à deux joueurs où nous n'imposons aucune hypothèse de contrôlabilité et où les contrôles des deux joueurs peuvent être couplés dans la dynamique, les fonctions de coût et les contraintes d'état. En particulier, nous caractérisons la fonction valeur de ce problème à travers un jeu différentiel auxiliaire, sans contraintes d'état, et nous proposons une procédure générale permettant d'approcher les contrôles optimaux des deux joueurs pour le problème initial. La seconde partie de cette thèse présente des méthodes numériques permettant de résoudre des problèmes de commande optimale avec une grande dimension d'état. Notre première contribution dans cette partie consiste à étendre la méthode de planification optimiste pour traiter des problèmes de commande optimale à horizon fini et en présence de contraintes d'état. En outre, nous établissons des résultats de convergence de ces algorithmes qui dépendent d'un budget de calcul donné. Une analyse numérique de ces méthodes est effectuée sur plusieurs exemples dans un espace d'état de grande dimension. Finalement, nous étudions des méthodes numériques, basées sur l'apprentissage profond et la programmation dynamique, pour des problèmes de commande optimale déterministe avec contraintes d'état ou pour des jeux différentiels à somme nulle et à deux joueurs.

Title: Hamilton-Jacobi Approach for State-Constrained Differential Games and Numerical Learning Methods for Optimal Control Problems

Keywords: Optimal control, Differential games, State constraints, Optimistic planning, Deep learning

Abstract: The aim of this work is to study state-constrained differential games by the Hamilton-Jacobi approach and to develop numerical learning methods to solve optimal control problems. The first part considers a two-person zero-sum differential game where we do not assume any controllability assumption and the controls of the two players are allowed to be coupled within the dynamics, the cost functions and the state constraints. In particular, we characterize the value function of such a problem through an auxiliary differential game free of state constraints and we propose a general approach allowing to construct approximated optimal feebacks of the constrained differential game for both players. The second part of this thesis presents some numerical methods allowing to solve optimal control problems in high dimensional state space. Our first contribution in this part consists in extending optimistic planning methods to deal with finite horizon problems in the presence of state constraints. Moreover, we provide convergence results of those algorithms that depend on given computing resources. A numerical analysis of these methods is carried out on several examples in high dimensional state space. Finally, we investigate numerical methods, based on deep learning and dynamic programming, for state-constrained deterministic optimal control problems or for controlled two-person zero-sum differential games.

Institut Polytechnique de Paris
91120 Palaiseau, France

INSTITUT
POLYTECHNIQUE
DE PARIS