Gran Sasso Science Institute

PHD PROGRAMME IN MATHEMATICS IN NATURAL, SOCIAL AND LIFE SCIENCES

10/02/2020

A Tree-Structure Algorithm for Optimal Control Problems via Dynamic Programming

PHD CANDIDATE
Luca Saluzzi

ADVISOR
Prof. Maurizio Falcone
Sapienza, University of Rome

GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

Thesis Jury Members
Elisabetta Carlini (Sapienza Università di Roma)
Raffaele D'Ambrosio (Università degli studi dell'Aquila)
Lars Grüne (University of Bayreuth)
Dante Kalise (University of Nottingham)
Michele Palladino (Gran Sasso Science Institute, L'Aquila)

Thesis Referees
Lars Grüne (University of Bayreuth)
Dante Kalise (University of Nottingham)

Acknowledgement

First of all, I would like to express my gratitude to my advisor Prof. Maurizio Falcone for its support during these three years. He has been not only a guide for my academic growth, but also a fellow traveler always present in time of troubles.

A special thank to my co-advisor Prof. Alessandro Alla for having dedicated his time in supporting and helping me. Thank you for believing in me and in my ideas.

I want also to thank the referees of this thesis, Prof. Lars Grüne and Prof. Dante Kalise, for their useful remarks which opened my eyes on new perspectives and possible future directions.

I would like to thank all the people who made these three years the most beautiful period of my life: my colleagues in GSSI (in particular Roberto and Roberta ;)), my friends in Sapienza and i potentini (especially Marco and Laura).

It is important for me to thank the city of L'Aquila, my second hometown, a safe harbour which is slowly rising from the ashes.

I cannot omit to thank Maurizione, who helped me to survive in this last period, this thesis is also yours!

Infine un grazie alla mia famiglia, che c'è e ci sarà sempre.

Contents

Chapter 1

Introduction

1.1 General Background

Optimal control theory deals with the computation of control strategies for complex dynamical systems in order to optimize their performance according to a fixed cost functional. Nowadays, this research area has been growing in importance since new numerical techniques have become available to solve a wide class of industrial and financial problems.
In this thesis we will focus on the finite horizon optimal control problems, the so-called *Bolza problems* that consists in minimizing the following cost functional:

$$J_{x,t}(u) = \int_t^T L(y(s), u(s), s) e^{-\lambda(s-t)}\, ds + g(y(T)) e^{-\lambda(T-t)} \tag{1.1}$$

over all the possible solutions of the following controlled dynamical system:

$$\begin{cases} \dot{y}(s) = f(y(s), u(s), s), \quad s \in (t, T], \\ y(t) = x, \end{cases} \tag{1.2}$$

where $u : [t, T] \to U \subset \mathbb{R}^m$ is taken in set of admissible controls $\mathcal{U}$. The problem can be formulated in the following way:

$$\inf_{u \in \mathcal{U}} J_{x,t}(u) \quad s.t. \quad y(s)\, satisfies\, (1.2). \tag{1.3}$$

The optimal control problem (1.3) can be solved with two main approaches: the Pontryagin Maximum Principle (PMP) and the Dynamic Programming Principle (DPP). The PMP was formulated in 1956 by the Russian mathematician Lev Pontryagin ([49]) and provides necessary conditions for the optimality which lead to a two-point boundary value problem. This approach produces an open-loop control which is rather unstable in presence of perturbations. For the purposes of this thesis we will concentrate on the Dynamic Programming approach. The DDP was introduced by Richard Bellmann in the 1950s ([9]) and it provides a synthesis of the feedback control law via the solution of a nonlinear partial differential equation (the Hamilton-Jacobi-Bellman (HJB) equation) that gives a characterization of the value function, Once the value function is known (or approximated), the synthesis of a feedback control law can be computed. This approach has been revitalized in the 80's by the development of a theory of weak solutions for the HJB equation, the so-called viscosity solutions, introduced by Crandall and Lions in the middle of the 80s (see, e.g., the monographs by Bardi and Capuzzo-Dolcetta ([7]) on deterministic control problems and by Fleming and Soner ([34]) on stochastic control problems). The theory related to this approach is now rather complete and established giving a complete characterization of the value function as the unique viscosity solution of the HJB equation for many classical control problems. It is important to note that even in low-dimension the numerical solution of HJB equations is a challenging problem since the value function associated to our control problem is known to be only Lipschitz-continuous also when the dynamics and the running costs are regular functions. Because of its lack of regularity, we cannot consider numerical schemes exploiting the regularity of the solution. Moreover, it is well-known that the DPP approach suffers from the "curse of dimensionality", expression coined by Bellman himself:

> *"..what casts the pall over our victory celebration? It is the curse of dimensionality, a malediction that has plagued the scientist from the earliest days."*

The HJB equation is set in a domain with the same dimension of the dynamical system. This dimension could be very high since if is often greater than 10 in many industrial applications, it can be even greater (order of thousands) when the dynamical system comes from a semi-discretization of an evolutive Partial Differential Equation (PDE). These difficulties have reduced the impact of the general theory over real industrial applications.

However, there is a complete literature for low dimensional numerical methods to solve the HJB equation (see e.g. the monographies by Sethian [54], Osher and Fedkiw [29] and Falcone and Ferretti [26]). In general all the PDE methods require a discretization in space, leading to problems in memory allocations for high dimensional systems.

The starting point of this thesis relies in particular on semi-Lagrangian schemes. In this context Falcone and Giorgi in [27] presented an approximation scheme for the evolutive HJB equation arising from optimal control problems, together with some error estimates. The proposed method works on a triangulation of a fixed domain and this becomes unfeasible for high dimension. Several efforts have been made to mitigate the curse of dimensionality. A possible way to address this problem stands in the splitting of the solution HJB equation into blocks of reasonable size. In this context we mention [28] for a domain decomposition method with overlapping between the subdomains and [14] for similar results without overlapping. This kind of approach deals with subdomains with simple geometry, but it introduces computational expensive boundary conditions. Another way, based on Al'brekht method ([1]), has been proposed in [47] and it consists in a patchy decomposition. Later in [12] the patchy idea has been extended creating subdomains which are almost invariant with respect to the optimal dynamics, avoiding the transmission conditions at the internal boundaries. More recently other decomposition techniques for optimal control problems and games have been proposed in [31] where the parallel algorithm is based on the construction of independent sub-domains and in [32] where a parallel version of the Howard's algorithm is proposed and analyzed. In general, domain decomposition methods enables to reduce a huge problem into a series of smaller subproblems which can be solved on different CPU via a parallel method, but the approximation schemes used in every subdomain are rather standard. An improvement can be obtained using efficient acceleration methods for the computation of the value function in every subdomain, e.g. by fast-marching or fast-sweeping techniques [53, 56]. In the framework of optimal control problems an efficient acceleration technique based on the coupling between value and policy iterations has been recently proposed and studied in [2]. Although domain decomposition coupled with acceleration techniques can help to solve problems up to dimension 10, we can not solve problems beyond this limit with a direct approach.

A reasonable solution to attack high-dimensional problems is to apply first model order reduction techniques, e.g. Proper Orthogonal Decomposition (POD,[57]). POD technique allows to consider a low dimensional problem which preserves the most important features of the original system. Considering a reduced space of dimension $d < 5$, it is possible to apply the classical method. The interested reader will find more information on this coupling in [43] and [5]. In the last chapter we will consider this technique, extending it in our framework. For the sake of completeness we mention other techniques present in literature to attack the curse of dimensionality. A possible approach in this context relies on the theory of max-plus algebras ([45, 46]). This technique avoids the space discretization, considering a particular basis, the max-plus basis, to express the value function. It is based on the computation of the coefficients of the basis, whose number increases exponentially in the number of time steps, restricting the possible applications of the method. Another technique to solve the optimal control problem is the so-called Model Predictive Control method. This approach does not solve directly the HJB equation, but it splits the problem in short horizon sub-problems computing the optimal trajectory in each subinterval and restarting the procedure. This method depends on the prediction horizon, which must be chosen properly in order to get accurate results. We refer the interested reader to the monograph [44] and to the recent introduction to MPC [36]. Finally, we should also mention that in [21, 22] has been proposed to apply a discrete version of Hopf-Lax representation formulas for Hamilton-Jacobi equations avoiding the global approximation on a grid. The advantage of this method is that it can be applied at every point in the space and that it can be easily parallelized.

However, this method can not be used for general nonlinear control problems since the Hopf-Lax representation formula is valid only for hamiltonians of the form $H(Du)$, whereas the hamiltonian related to optimal control problems is typically $H(x, u, Du)$.

1.2 Contributions

The aim of this thesis is to introduce a new algorithm to attack the curse of dimensionality: the Tree Structure Algorithm (TSA, [3]). Our proposed algorithm does not require the space discretization and the construction of a grid and this allows to reduce the memory allocations extending the possibility to apply the DP approach. For the finite horizon problem this can be done via the construction of a tree-structure that will account for the controlled dynamics. Considering a discretization of the control set and the application of a Euler scheme for the dynamics, we can construct a tree which mimic the possible trajectories and we will solve the HJB equation on the constructed tree.

The algorithm applies a Euler scheme to solve the equation for the dynamical system, but the extension to high-order scheme is straightforward. Indeed, we provide a Discrete Dynamical Programming Principle for high-order schemes, as suggested in [25] for the infinite horizon case. It is important to notice that the tree structure depends on the dynamics, the number of steps used for the time discretization and the cardinality of the control set. This can produce a huge number of branches in the tree, however not all these branches must be considered to get an accurate approximation of the value and a pruning criteria has been introduced to reduce the complexity of the algorithm. Moreover, we consider a post-processing procedure to get a more accurate feedback reconstruction starting from the creation of a tree with few controls. Working on the tree has several advantages:

(i) we do not need to define a priori a numerical domain Ω where we want to solve the problem; the original tree is constructed by the controlled dynamics;

(ii) we do not need to build a space grid and to make a space interpolation on the grid nodes, and therefore we do not introduce an interpolation error in the approximation of the value function;

(iii) the pruned tree allows us to deal with high-dimensional problems.

In conclusion, with respect to the standard space discretization we can drop the interpolation step that is rather expensive in high-dimension and we do not need the classical assumptions at the boundary of Ω which classically requires one to have an invariant dynamics or to impose boundary conditions (Dirichlet, Neumann, or state constraint). Via the tree structure algorithm we eliminate these difficulties at least for the finite horizon problem and we can directly solve the discrete time HJB equation for high dimension without any particular assumption on the structure of the problem as in a model reduction context.

Moreover, we develop an error analysis of the TSA giving precise error estimates. In particular, as proposed in [16] for the infinite horizon case, under the assumption of semiconcavity, we improve the order of convergence provided in [27] for the finite horizon optimal control problem and we extend the error analysis to the pruned TSA. We refer to [15] for more details on the property of semiconcavity and its connection with the HJB equation. Therefore, it is clear that the method is expensive when we deal with PDEs since it requires to solve many equations for several control inputs. It is then natural to couple the TSA with POD in order to speed up the method. With the approach we have four major advantages:

(i) we build the snapshots set upon all the trajectories that appear in the tree, avoiding the selection of a forecast for the control inputs which is always not trivial for model reduction,

(ii) the application of POD also allows an efficient pruning since it reduces the dimension of the problem and provides information on the most variable components,

(iii) the theory of DPP is valid on the whole state space $\mathbb{R}^d$ but, in general, for numerical reasons we need to restrict our equation to a bounded domain. Our method avoids to define the numerical domain for the projected problem, which is a difficult task since we lose the physical meaning of the reduced coordinates,

(iv) we are not restricted to consider a reduced space dimension smaller than 5 as in e.g. [5], [43], which was a limitation of the method since many classes of PDEs require more basis functions to capture the essential features.

In presence of nonlinearities the application of POD is still computationally expensive since the nonlinear part depends on the dimension of the original problem. In this case we use the Discrete Empirical Interpolation Method (DEIM, [17]). Thanks to a computation of the POD basis functions for the nonlinear part, we obtain a low-dimensional problem completely independent from the high-dimensional problem. Finally, we provide a-priori error estimate for the coupling between TSA and model order reduction to validate our approach.

1.3 Organization of the thesis

The thesis is divided in five chapters.

- **Chapter 1** recalls the Dynamical Programming Principle and the evolutive Hamilton-Jacobi-Bellman equation. We discuss some properties of the solution, focusing in particular on the notion of semiconcavity (see [15] for more details) which constitutes the key point for our error estimates. Afterwards, we recall some numerical methods present in literature to solve this problem. First, we present the semi-Lagrangian scheme proposed in [27] to solve the finite horizon optimal control problem. This method represents the starting point of our proposed algorithm. Then we pass to consider some techniques which try to mitigate the curse of dimensionality in particular cases: the Linear Quadratic Regulator, Darbon-Osher method and the max-plus based algorithm.

- **Chapter 2** is the core of the thesis, it introduces the Tree-Strucure Algorithm (TSA). First, we explain how the algorithm works, explaining step by step the construction of the tree and the resolution of the HJB equation on the constructed domain. Then we pass to technical improvements to speed-up the method: the tree structure increases exponentially in the number of discrete controls and time steps and for this reason a *pruning criteria* has been introduced. Based on a neighbour search, the pruning criteria allows to cut the nodes which are not relevant in the resolution of the HJB equation. Since the neighbour search is an expensive tool, the pruning procedure has been speeded up thanks to the application of Principal Component Analysis techniques ([48, 39]). After the computation of the value function, one can consider some feedback reconstruction techniques and we present two possible methods. Finally, we discuss the extension of the algorithm to high-order schemes, presenting a new Discrete Dynamical Programming Principle and showing some examples.

- **Chapter 3** provides the theoretical framework for the proposed algorithm. We study a-priori error estimate for the TSA, obtaining a first order convergence. The assumption of the semiconcavity is a key ingredient for the proposed proof. We present also an error estimate which takes into account the error in the minimization by comparison. Then, we study the pruning case, getting again a first order convergence under suitable assumptions on the choice of the pruning threshold. Finally, we extend the assumptions on the pruning criteria to high-order schemes. Finally, we present some numerical tests which confirm the theoretical findings.

- **Chapter 4** is dedicated to optimal control problems for PDEs. Since the problem presents some difficulties for the high dimensionality, we study the coupling of the TSA with model reduction techniques. First, we recall the Proper Orthogonal Decomposition and the Discrete Emprical Interpolation Method and then we show how to apply them in our framework. Finally, we study an error estimate for the coupling of the two methods. In the section of numerical tests we study the order of convergence of the TSA-POD algorithm and we apply it in the framework of 2D nonlinear PDEs.

- **Chapter 5** provides our conclusions and some future directions.

1.4 Original material

In this section we mention the original contributions behind this thesis.

Chapter 3 is based on the paper [3], published in SIAM J. of Scientific Computing, and on the published IFAC CPDE conference article [4].

Chapter 4 is based on the paper [51] submitted to SIAM J. of Numerical Analysis.

Chapter 5 is based on the paper [6] which will appear for a special issue in Applied Numerical Mathematics.

Chapter 2

Background

In this chapter we will sketch the essential features of the dynamic programming approach and its numerical approximation which will be useful in the following chapters. First of all, we introduce the Dynamic Programming Principle, which leads in our framework to a first order time-dependent Hamilton-Jacobi-Bellman equation. We will discuss some properties of the solution of this equation, focusing in particular on the property of semiconcavity. We will pass to some numerical approximations present in literature to approach this kind of problem.

2.1 Programming Principle and Hamilton-Jacobi-Bellman equation

Let us consider the classical *finite horizon optimal control problem.* Let the system be driven by

$$\begin{cases} \dot{y}(s) = f(y(s), u(s), s), \quad s \in (t, T], \\ y(t) = x \in \mathbb{R}^d. \end{cases} \tag{2.1}$$

We will denote by $y : [t, T] \to \mathbb{R}^d$ the solution, by $u : [t, T] \to \mathbb{R}^m$ the control, by $f : \mathbb{R}^d \times \mathbb{R}^m \times [t, T] \to \mathbb{R}^d$ the dynamics and by

$$\mathcal{U} = \{u : [t, T] \to U, \text{measurable}\}$$

the set of admissible controls, where $U \subset \mathbb{R}^m$ is a compact set.

Under the assumptions of boundedness and Lipschitz-continuity of the vector field f, there exists a unique solution for (2.1) for each $u \in \mathcal{U}$. We refer to e.g. [7] for a precise statement.

The cost functional for the finite horizon optimal control problem will be given by

$$J_{x,t}(u) := \int_t^T L(y(s, u), u(s), s) e^{-\lambda(s-t)}\, ds + g(y(T)) e^{-\lambda(T-t)}, \tag{2.2}$$

where $L : \mathbb{R}^d \times \mathbb{R}^m \times [t,T] \to \mathbb{R}$ is the running cost and $\lambda \geq 0$ is the discount factor. In what follows we will assume that the functions f, L, g are bounded:

$$\begin{gathered}|f(x,u,s)| \leq M_f, \quad |L(x,u,s)| \leq M_L, \quad |g(x)| \leq M_g, \\ \forall\, x \in \mathbb{R}^d,\ u \in U \subset \mathbb{R}^m,\ s \in [t,T],\end{gathered} \tag{2.3}$$

the functions f, L are Lipschitz-continuous with respect to the first variable

$$\begin{gathered}|f(x,u,s) - f(y,u,s)| \leq L_f|x-y|, \quad |L(x,u,s) - L(y,u,s)| \leq L_L|x-y|, \\ \forall\, x, y \in \mathbb{R}^d, u \in U \subset \mathbb{R}^m, s \in [t,T],\end{gathered} \tag{2.4}$$

and finally the cost g is also Lipschitz-continuous:

$$|g(x) - g(y)| \leq L_g|x-y|, \quad \forall x, y \in \mathbb{R}^d. \tag{2.5}$$

The goal is to find a state-feedback control law $u(t) = \Phi(y(t), t)$, in terms of the state equation $y(t)$, where Φ is the feedback map. To derive optimality conditions we use the well-known Dynamic Programming Principle (DPP) due to Bellman. We first define the value function for an initial condition $(x,t) \in \mathbb{R}^d \times [t,T]$:

$$v(x,t) := \inf_{u \in \mathcal{U}} J_{x,t}(u) \tag{2.6}$$

which satisfies the DPP, i.e. for every $\tau \in [t,T]$:

$$v(x,t) = \inf_{u \in \mathcal{U}} \left\{ \int_t^\tau L(y(s), u(s), s) e^{-\lambda(s-t)} ds + v(y(\tau), \tau) e^{-\lambda(\tau - t)} \right\}. \tag{2.7}$$

Due to (2.7) we can derive the HJB for every $x \in \mathbb{R}^d$, $s \in [t,T)$:

$$\begin{cases} -\dfrac{\partial v}{\partial s}(x,s) + \lambda v(x,s) + \max\limits_{u \in U} \{-L(x,u,s) - \nabla v(x,s) \cdot f(x,u,s)\} = 0, \\ v(x,T) = g(x). \end{cases} \tag{2.8}$$

Defining the Hamiltonian

$$H(s,x,p) = \max_{u \in U} \{-L(x,u,s) - p \cdot f(x,u,s)\},$$

we can rewrite (2.8) as

$$\begin{cases} -\dfrac{\partial v}{\partial s}(x,s) + \lambda v(x,s) + H(s,x,\nabla v) = 0, \\ v(x,T) = g(x). \end{cases} \tag{2.9}$$

It is well known that equation (2.9) is in general not well-posed. For this reason the theory of viscosity solutions was introduced. This theory, initiated in the early 80's by the papers of M.G. Crandall and P.L. Lions [19], provides a convenient framework for dealing with the lack of smoothness of the value functions arising in dynamic optimization problems. We pass to introduce one of the possible definitions of viscosity solution.

Definition 2.1.1. *A continuous function $u : [0,T] \times \mathbb{R}^d \to \mathbb{R}$ is a viscosity subsolution of* (2.9) *if, for every C^1 function $\Phi = \Phi(t,x)$ such that $u - \Phi$ has a local maximum at (t,x), one has*

$$-\partial_s \Phi(x,s) + \lambda v(x,s) + H(t,x,\nabla\Phi) \leq 0.$$

A continuous function $u : [0,T] \times \mathbb{R}^d \to \mathbb{R}$ is a viscosity supersolution of (2.9) *if, for every C^1 function $\Phi = \Phi(t,x)$ such that $u - \Phi$ has a local minimum at (t, x),one has*

$$-\partial_s \Phi(x,s) + \lambda v(x,s) + H(t,x,\nabla\Phi) \geq 0.$$

A continuous function $u : [0,T] \times \mathbb{R}^d \to \mathbb{R}$ is a viscosity solution of (2.9) *if it is a viscosity subsolution and a viscosity supersolution of* (2.9).

In this setting it is possible to prove that there exists a unique viscosity solution of (2.9) under suitable assumptions.

Theorem 2.1.1. *Assuming* (2.3), (2.4) *and* (2.5), *the value function $v(x,t)$ is the unique viscosity solution of* (2.9).

The analytical solution of (2.9) is known just in particular cases, for this reason we need efficient numerical methods to solve it. Suppose that the value function is known, analytically or by numerical methods, then it is possible to compute the optimal feedback control as:

$$u^*(t) := \arg\max_{u \in U} \left\{ -L(x,u,t) - \nabla v(x,t) \cdot f(x,u,t) \right\}. \tag{2.10}$$

We will discuss more about the synthesis of the feedback control in the next chapters.

2.1.1 The infinite horizon case

In the infinite horizon case the cost functional is given in the following form

$$J_x(u) := \int_0^\infty L(y(s,u),u(s)) e^{-\lambda s} \, ds \ , \tag{2.11}$$

where the discount coefficient λ ensures that the integral is finite, under the assumption of boundedness for the running cost L. In this case we define the set of admissible controls as:

$$\mathcal{U} = \{u : [0,+\infty) \to U, \text{measurable}\}$$

and the value function as:

$$v(x) = \inf_{u \in \mathcal{U}} J_x(u) \ , \tag{2.12}$$

which satisfies now the following stationary Hamilton-Jacobi-Bellmann equation:

$$\lambda v(x) + \max_{u \in U} \left\{ -L(x,u) - \nabla v(x) \cdot f(x,u) \right\} = 0. \tag{2.13}$$

The Dynamic Programming Principle in this case reads as follows

$$v(x) = \inf_{u \in \mathcal{U}} \left\{ \int_0^\tau L(y(s), u(s)) e^{-\lambda s} ds + v(y(\tau)) e^{-\lambda \tau} \right\}, \quad \forall \tau \geq 0. \tag{2.14}$$

The DPP (2.14) can be discretized and the numerical solution $V(x)$ can be found solving the following fixed point problem:

$$V(x) = \min_{u \in U} \{ e^{-\lambda \Delta t} V(x + \Delta t f(x, u)) + \Delta t L(x, u) \}. \tag{2.15}$$

This problem can be solved numerically via the value iteration method, the policy iteration method or a coupling of the two techniques. We address the interested reader to [2] for a complete description of these methods.

2.1.2 The minimum time problem

Another class of optimal control problems is given by the *minimum time problem*. In this case the aim is to steer the solution to a target $\mathcal{T}$ in the shortest time. It is possible to define the first time of arrival on the target $\mathcal{T}$ as

$$t_x(u) = \begin{cases} \min\{t : y(t, u) \in \mathcal{T}\} & \text{if } y(t, u) \in \mathcal{T} \text{ for some } t, \\ +\infty & \text{else.} \end{cases} \tag{2.16}$$

In this case the value function is also called minimum time function and it is defined by

$$T(x) = \inf_{u \in \mathcal{U}} t_x(u) \ .$$

In this context we need to provide the definition of *reachable set*, *i.e.* the domain in which the value function is finite, and then we can introduce the DPP satisfied by the value function.

Definition 2.1.2. *The reachable set is defined as $\mathcal{R} = \{x \in \mathbb{R}^d : T(x) < +\infty\}$, i.e. the set of points from which the dynamics can reach the target $\mathcal{T}$.*

Proposition 2.1.1 (Dynamic Programming Principle)**.** *For all $x \in \mathcal{R}$, $0 \leq t \leq T(x)$ (so that $x \notin \mathcal{T}$), the value function satisfies*

$$T(x) = \inf_{u \in \mathcal{U}} \{ t + T(y(t, u; x)) \}. \tag{2.17}$$

From (2.17) we can derive the following HJB equation

$$\max_{u \in U} \{ -\nabla T(x) \cdot f(x, u) \} - 1 = 0, \qquad x \in \mathcal{R} \setminus \mathcal{T}, \tag{2.18}$$

coupled with the following natural boundary conditions

$$\begin{cases} T(x) = 0 & x \in \partial \mathcal{T}, \\ \lim_{x \to \partial \mathcal{R}} T(x) = +\infty. \end{cases} \tag{2.19}$$

For more details and results for the minimum time problem we refer to [26].

2.2 Properties of the value function

In this subsection we want to present the main results for the solution of (2.8) which will be useful to prove error estimates for the algorithm introduced later on. We will begin with a classical result on the continuity and Lipschitz-continuity of the value function.

Proposition 2.2.1. *Assuming* (2.3), (2.4) *and* (2.5), *then*

- *v is bounded and continuous in $\mathbb{R}^d \times [0, T]$, for all $T > 0$;*
- *if $\lambda > 0$, then $v \in BC(\mathbb{R}^d \times [0, +\infty])$,*
- *v is Lipschitz continuous on $K \times [0, T]$ for all compact sets $K \subset \mathbb{R}^d$ and $T > 0$.*

A proof of this proposition can be found in [7]. An essential ingredient for the first order error estimates stands on the definition of *semiconcavity.*

Definition 2.2.1. *Given $A \subset \mathbb{R}^d$ an open set, we say that a function $u : A \to \mathbb{R}$ is semiconcave with linear modulus if it is continuous in A and there exists $C \geq 0$ such that*

$$u(x+h) + u(x-h) - 2u(x) \leq C|h|^2, \tag{2.20}$$

for all $x, h \in \mathbb{R}^d$ such that $[x-h, x+h] \subset A$. The constant C above is called semiconcavity constant for u.

We denote by $SCL(A)$ the functions which are semiconcave in A with a linear modulus and by $SCL_{loc}(A)$ for the functions which are semiconcave with a linear modulus locally in A, *i.e.*, on every compact subset of A. The interested reader will find all the definitions and results regarding semiconcave functions in [15]. It is possible to give different characterizations for semiconcave functions, as stated in the following result.

Proposition 2.2.2. *Given $u : A \to \mathbb{R}$, with $A \subset \mathbb{R}^d$ open convex, and given $C \geq 0$, the following properties are equivalent:*

- *u is semiconcave with a linear modulus in A with semiconcavity constant C;*
- *u satisfies*

  $$\lambda u(x) + (1-\lambda)u(y) - u(\lambda x + (1-\lambda)y) \leq C\frac{\lambda(1-\lambda)}{2}|x-y|^2,$$

  for all x, y such that $[x, y] \subset A$ and for all $\lambda \in [0, 1]$;
- *the function $x \to u(x) - \frac{C}{2}|x|^2$ is concave in A;*

- *there exist two functions $u_1, u_2 : A \to \mathbb{R}$ such that $u = u_1 + u_2$, with u_1 concave and $u_2 \in C^2(A)$ satisfying $\|D^2 u_2\|_\infty \leq C$;*
- *for any $\eta \in \mathbb{R}^d$ such that $|\eta| = 1$ we have $\partial^2_\eta u \leq C$ in A in the sense of distributions;*
- *u can be represented as $u(x) = \inf_{i\in I} u_i(x)$, where $\{u_i\}_{i\in I}$ is a family of functions of $C^2(A)$ such that $\|D^2 u_i\|_\infty \leq C$, for all $i \in I$.*

Under suitable assumptions one can prove that the value function is semiconcave, as stated in the next theorem.

Theorem 2.2.1. *Let us assume $f(\cdot, u)$ and $L(\cdot, u)$ are Lipschitz-continuous and the control set U is compact. Moreover let us suppose $f_x(\cdot, u)$ is Lipschitz-continuous, $g \in SCL_{loc}(\mathbb{R}^n)$ and the following assumption on L:*

$$L(x,u) + L(y,u) - 2L\left(\frac{x+y}{2}, u\right) \leq \lambda_R |x-y|^2, \quad x, y \in B_R, u \in U.$$

Then $v \in SCL_{loc}([0,T] \times \mathbb{R}^n)$.

We address the interested reader to [15] for a proof of the theorem and a complete overview on this topic. The property of semiconcavity will be essential in Chapter 4 since it will allow us to prove a first order convergence for our proposed algorithm.

2.3 Numerical approximation of HJB equation

In this section we present the principal methods to solve numerically Equation (2.8). First of all we introduce the Semi-Lagrangian scheme, which is the main building block of our algorithm. For the sake of completeness, we mention other techniques for the approximation of the first order time-dependent Hamilton-Jacobi-Bellman equation.

2.3.1 Semi-Lagrangian scheme

Equaton (2.8) is a nonlinear PDE of the first order which is hard to solve analitically although a general theory of weak solutions is available in e.g. [7]. Rather, we can solve equation (2.8) numerically by means of finite difference or semi-Lagrangian methods. In this section we recall the semi-Lagrangian method. One usually starts the numerical method by discretizing in time the underlying control problem with a time step $\Delta t := [(T-t)/\overline{N}]$ where $\overline{N}$ is the number of temporal time steps and then projects the semi-discrete scheme on a grid obtaining the fully discrete scheme:

$$\begin{cases} V_i^n = \min_{u\in U}[\Delta t\, L(x_i,u,t_n) + e^{-\lambda\Delta t} I[V^{n+1}](x_i+\Delta t f(x_i,u,t_n))], & \\ & n=\overline{N}-1,\dots,0, \\ V_i^{\overline{N}} = g(x_i) & x_i\in\Omega, \end{cases} \tag{2.21}$$

where $t_n = t+n\Delta t$, $t_{\overline{N}} = T$, Ω is the numerical domain and x_i is an element of its discretization, $V_i^n := V(x_i,t_n)$ and $I[\cdot]$ is an interpolation operator which is necessary to compute the value of V^n at the point $x_i+\Delta t\, f(x_i,u,t_n)$ (in general, this point will not be a node of the grid). The interested reader will find in [26] a detailed presentation of the scheme. The result of Lipschitz-continuity of the continuous value function $v(x,t)$ can be extended to its numerical approximation $V(x,t)$ as explained in the following proposition. The proof follows closely from the continuous version in [7, Prop. 3.1].

Proposition 2.3.1. *Let us suppose the functions $f(\cdot,u,t), L(\cdot,u,t)$ and $g(\cdot)$ are Lipschitz continuous uniformly with respect to the other variables. Then, the numerical value function $V^n(x)$ is Lipschitz in x*

$$|V^n(x)-V^n(y)| \le \begin{cases} |x-y|\left(\frac{L_L}{L_f-\lambda}(e^{(T-t_n)(L_f-\lambda)}-1)+L_g e^{(T-t_n)(L_f-\lambda)}\right), & \\ & \textit{for } L_f>\lambda, \\ |x-y|\left(L_L(T-t_n)+L_g e^{(T-t_n)(L_f-\lambda)}\right), & \\ & \textit{for } L_f\le\lambda, \end{cases} \tag{2.22}$$

$\forall\, x,y\in\mathbb{R}^d$ and $n=0,\dots,\overline{N}$.

Proof. In the case $n=\overline{N}$, we have that $V^{\overline{N}}(x)=g(x)$, then the estimate follows directly from the hypothesis on g.
In the case $n<\overline{N}$, we fix $\overline{x},\overline{y}\in\mathbb{R}^d$ and consider the following quantity $V^n(\overline{x})-V^n(\overline{y})$:

$$\begin{aligned} V^n(\overline{x})-V^n(\overline{y}) &\le e^{-\lambda\Delta t}V^{n+1}(\overline{x}+\Delta t f(\overline{x},u_*^n,t_n)) + \Delta t L(\overline{x},u_*^n,t_n) \\ &\quad - e^{-\lambda\Delta t}V^{n+1}(\overline{y}+\Delta t f(\overline{y},u_*^n,t_n)) - \Delta t\, L(\overline{y},u_*^n,t_n) \\ &\le e^{-\lambda\Delta t}(V^{n+1}(\overline{x}+\Delta t f(\overline{x},u_*^n,t_n)) - V^{n+1}(\overline{y}+\Delta t f(\overline{y},u_*^n,t_n))) \\ &\quad + \Delta t\, L_L|\overline{x}-\overline{y}|, \end{aligned} \tag{2.23}$$

provided that

$$u_*^n = \arg\min_{u\in U}\left\{e^{-\lambda\Delta t}V^{n+1}\left(\overline{y}+\Delta t f(\overline{y},u,t_n)\right)+\Delta t L(\overline{y},u,t_n)\right\}.$$

To achieve the desired estimate (2.22), we need to iterate (2.23) starting from $\overline{x}$ and $\overline{y}$ at time t_n. Let us first define the whole tree paths $\{x^m\}_m$ and $\{y^m\}_m$ as

$$x^m := x^n + \Delta t\sum_{j=n}^{m-1} f(x^j,u_*^j,t_j), \qquad y^m := y^n + \Delta t\sum_{j=n}^{m-1} f(y^j,u_*^j,t_j),$$

where

$$u_*^j = \arg\min_{u\in U} \left\{ e^{-\lambda\Delta t} V^{j+1}\left(y^j + \Delta t f(y^j, u, t_j)\right) + \Delta t L(y^j, u, t_j) \right\}, \quad j = n, \dots, m-1.$$

By the discrete Grönwall's lemma, it is easy to prove the following estimate for Euler schemes starting from $x^n = \overline{x}$ and $y^n = \overline{y}$

$$|x^{n+k} - y^{n+k}| \le |x^n - y^n| e^{k\Delta t L_f} = |\overline{x} - \overline{y}| e^{k\Delta t L_f}, \quad k = 0, \dots, \overline{N} - n. \tag{2.24}$$

Then, iterating (2.23) we obtain

$$\begin{aligned} V^n(\overline{x}) - V^n(\overline{y}) &\le \Delta t\, L_L \sum_{k=0}^{\overline{N}-n-1} e^{-\lambda k \Delta t} |x^{n+k} - y^{n+k}| + e^{-\lambda(T-t_n)} |g(x^{\overline{N}}) - g(y^{\overline{N}})| \\ &\le \Delta t\, L_L \sum_{k=0}^{\overline{N}-n-1} e^{-\lambda k \Delta t} |x^{n+k} - y^{n+k}| + L_g e^{-\lambda(T-t_n)} |x^{\overline{N}} - y^{\overline{N}}| \\ &\le |\overline{x} - \overline{y}| \left(\Delta t L_L \sum_{k=0}^{\overline{N}-n-1} e^{k\Delta t(L_f - \lambda)} + L_g e^{(T-t_n)(L_f-\lambda)} \right), \end{aligned} \tag{2.25}$$

where we used (2.24) and the Lipschitz continuity of g.
If $L_f > \lambda$, then by (2.25) and the equality $(\overline{N} - n)\Delta t = T - t_n$, we get

$$\begin{aligned} V^n(\overline{x}) - V^n(\overline{y}) &\le |\overline{x} - \overline{y}| \left(\Delta t L_L \frac{e^{(T-t_n)(L_f-\lambda)} - 1}{e^{\Delta t(L_f-\lambda)} - 1} + L_g e^{(T-t_n)(L_f-\lambda)} \right) \\ &\le |\overline{x} - \overline{y}| \left(\frac{L_L}{L_f - \lambda} (e^{(T-t_n)(L_f-\lambda)} - 1) + L_g e^{(T-t_n)(L_f-\lambda)} \right), \end{aligned} \tag{2.26}$$

whereas if $L_f \le \lambda$, noticing that $e^{k\Delta t(L_f-\lambda)} \le 1$, we directly obtain

$$V^n(\overline{x}) - V^n(\overline{y}) \le |\overline{x} - \overline{y}| \left(L_L(T - t_n) + L_g e^{(T-t_n)(L_f-\lambda)} \right). \tag{2.27}$$

Analogously, it is possible to obtain the same estimate for $V^n(\overline{y}) - V^n(\overline{x})$ which leads to the desired result.
□ □

We will take advantage of the estimate (2.23) to guarantee the feasibility of our proposed method. The numerical approximation of the feedback control follows directly from the SL-scheme (2.21) and reads

$$u_*^n(x) = \arg\min_{u\in U} [\Delta t\, L(x, u, t_n) + e^{-\lambda\Delta t} I[V^{n+1}](x + \Delta t f(x, u, t_n))].$$

2.3.2 Linear Quadratic Regulator

In this subsection we present a particular case of optimal control problem in which we can obtain the solution solving a Riccati differential equation ([50]). We consider a linear dynamical system

$$\begin{cases} \dot{y}(s) = Ay(s) + Bu(s), \quad s \in (t, T], \\ y(t) = x \in \mathbb{R}^d, \end{cases} \tag{2.28}$$

and a quadratic cost functional

$$J_{x,t}(u) := \int_t^T \big(y(s)^T Q y(s) + u(s)^T R u(s) \big)\, ds + y(T)^T F y(T), \tag{2.29}$$

with $A, Q, F \in \mathbb{R}^{d\times d}, B \in \mathbb{R}^{d\times M}$ and $R \in \mathbb{R}^{M\times M}$. In this case the value function is given by $v(x,t) = x^T P(t) x$, where $P(t)$ solves the so-called Riccati equation

$$-\dot{P}(t) = A^T P(t) + P(t)A - P(t)BR^{-1}B^T P(t) + Q, \quad P(T) = F \tag{2.30}$$

Solving (numerically or analytically) (2.30), one obtains the value function in terms of the matrix $P(t)$ and the feedback control is given by $u^*(t,x) = -K(t)x$, where the matrix $K(t) \in \mathbb{R}^{M\times d}$ is computed in the following way

$$K(t) = R^{-1} B^T P(t).$$

In this particular case we can obtain the information on the optimal trajectory "just" solving a differential matrix equation. This will turn out to be useful when we will compute the order of convergence of the proposed algorithm in high-dimension. For recent developments on the treatment of the Riccati equation in high dimension we refer to [41]. We stress on the fact that the Riccati equation holds in a particular setting (linear dynamics and quadratic cost functional). Our algorithm can be used in general frameworks and we will show in Chapter 5 examples of control for non-linear PDEs.

2.3.3 The Hopf formula and Darbon-Osher method

Now let us consider the following Hamilton-Jacobi equation

$$\begin{cases} \dfrac{\partial \varphi}{\partial s}(x,s) + H(\nabla \varphi) = 0, \\ \varphi(x,0) = g(x), \end{cases} \tag{2.31}$$

where the Hamiltonian H depends only on the gradient of the value function. In this case it is possible to obtain an explit solution via the Hopf formula ([38]). First we introduce the definition of Fenchel-Legendre transform, then we present the Hopf formula.

Definition 2.3.1. *Given a convex function $f : X \to \mathbb{R}$ on a convex domain X, we define the Fenchel-Legendre transform f^* as*

$$f^*(v) = \sup_{x\in\mathbb{R}^d} (x \cdot v - f(x)) . \tag{2.32}$$

Proposition 2.3.2. *Assume that H is convex, g is Lipschitz-continuous and*

$$\lim_{|p|\to+\infty} \frac{H(p)}{|p|} = +\infty .$$

Then the solution of equation (2.31) *is given by*

$$\varphi(x,t) = - \inf_{v\in\mathbb{R}^d} \{g^*(v) + tH(v) - x \cdot v\} , \tag{2.33}$$

where $g^*(v)$ is the Fenchel-Legendre transform of g.

Taking into account (2.33), in [?] Darbon and Osher propose a method to solve (2.31) mixing the Hopf formula and the split Bregman iterative approach ([35]). In Algorithm 1 we recall the split Bregman iterative scheme.

Algorithm 1 Split Bregman iterative scheme

```
1: $v^0 = x$, $d^0 = x$ and $b^0 = 0$
2: for $k = 1, 2, \ldots$ do
3:     $v^{k+1} = \arg\min_{v\in\mathbb{R}^d} \left\{g^*(v) - x \cdot v + \frac{\lambda}{2}\|d^k - v - b^k\|_2^2\right\}$
4:     $d^{k+1} = \arg\min_{d\in\mathbb{R}^d} \left\{tH(d) + \frac{\lambda}{2}\|d - v^{k+1} - b^k\|_2^2\right\}$
5:     $b^{k+1} = b^k + v^{k+1} - d^{k+1}$
```

The sequences $\{v^k\}_k$ and $\{d^k\}_k$ are both converging to the minimizer of (2.33). Steps 3 and 4 in Algorithm 1 need the computation of a minimization and they can be reformulated as

$$\arg\min_{w}\{\alpha f(w) + \frac{1}{2}\|w - z\|_2^2\}, \tag{2.34}$$

with f convex function. The unique minimizer $\overline{w}$ can be found computing the proximal map of f, *i.e.*

$$\overline{w} = (I + \alpha\partial f)^{-1}(z),$$

where ∂f stands for the subdifferential of f. More details on the proximal map can be found in [18]. For some particular cases it is possible to derive explicit expressions for the proximal map (see [?] for some examples).

This technique turns out to mitigate the curse of dimensionality, but it deals with a particular kind of Hamilton-Jacobi-Bellman equation, *e.g.* with an Hamiltonian depending only on the gradient of the solution, and this restricts the possible applications for the method.

2.3.4 A max-plus-based algorithm

Another way to solve the HJB equation relies on the theory of the max-plus algebra. We will consider the max-plus-based algorithm introduced by Fleming and McEneaney in [33]. In the max-plus algebra we define the addition $\oplus$ and the multiplication $\otimes$ in $\mathbb{R}$ as

$$a \oplus b = \max\{a, b\}, \quad a \otimes b = a + b.$$

We refer to [20] for more details and properties on this topic. Let us consider a particular dynamics

$$\begin{cases} \dot{x}(s) = f(x) + \sigma(x)w \ , \\ x(0) = x_0 \in \mathbb{R}^d, \end{cases} \tag{2.35}$$

where $w \in L^2_{loc}([0,\infty); \mathbb{R}^m)$ is the disturbance in the dynamics, $x(t) \in \mathbb{R}^d$ and σ is a $d \times m$ matrix-valued function. Introduced the cost criterion of the form

$$J(T, x_T, w) = \phi(x_0) - \frac{1}{2}\int_0^T |w(t)|^2 dt \ ,$$

we want to compute the following value function

$$P(T, x_T) = \sup_{w \in L_2} J(T, x_T, w). \tag{2.36}$$

We know that the value function P verifies the following dynamic programming principle ([33])

$$P(t+\delta, x) = \sup_{w \in L_2} \left\{ P(t, x(t)) - \frac{1}{2}\int_t^{t+\delta} |w(r)|^2 dr \right\}, \quad t \leq t+\delta \leq T, \tag{2.37}$$

where $x(t)$ is given by (2.35) and $x(t+\delta) = x$, The dynamic principle (2.37) can be rewritten as

$$P(t+\delta)(x) = S_\delta[P(t, \cdot)](x), \tag{2.38}$$

where $S_\delta[P(t,\cdot)](x)$ is the solution operator given by the right-hand side of (2.37). Exploiting the dynamic principle (2.37), the algorithm introduced in [33] is based on the following three main results:

1. the solution operator S_δ is linear in the max plus algebra;
2. given a function ψ semiconvex, then $S_\delta[\psi]$ is semiconvex for each $\delta > 0$;
3. a continuous and semiconvex function ψ, with semiconvexity constant equal to c, can be expressed in terms of max-plus basis in a ball of radius R in the following form

$$\psi(x) = \bigoplus_{i=1}^{\infty} [a_i \otimes g_i(x)] \quad \forall x \in B_R,$$

where

$$g_i(x) = -\frac{\hat{c}}{2}|x - x_i|^2, \quad a_i = -\max_{x\in B_R}[g_i(x) - \Psi(x)],$$

with $\{x_i\}_i$ a countable dense set and $\hat{c} \in (c, +\infty)$.

Given all these ingredients, we can introduce Algorithm 2.

Algorithm 2 Max-plus based algorithm

```
1: Fix $\delta$, $n$, $P(0,x) = \Phi(x)$, $t_0 = 0$
2: for $\ell = 0, \dots, N-1$ do
3:     Estimation of the semiconvexity constant $\overline{c}$ for $P(t_{\ell-1}, \cdot)$
4:     Choice of a basis set $B_{j_1} = \{g_{i,j_1}\}_i$ with $\hat{c}_{j_1} > \overline{c}$
5:     $a_i^{\ell-1} = -\max_{x\in B_R}[g_i(x) - P(t_{\ell-1}, x)]$
6:     Choice of a basis set $B_{j_2} = \{g_{i,j_2}\}_i$ with $\hat{c}_{j_2}$ greater than the semiconvexity constants of $\{S_\delta[g_{j_1,i}]\}_{i=1}^n$
7:     $b_{k,i} = -\max_{x\in B_R}[g_{j_2,k}(x) - S_\delta[g_i](x)]$
8:     $P(t_{\ell+1}, x) \approx \bigoplus_{k=1}^n [(\bigoplus_{i=1}^n (b_{k,i} \otimes a_i)) \otimes g_{j_2,k}(x)]$
```

Exploiting its linearity and semiconvexity, the solution operator can be expressed in terms of max-plus basis functions, considering the basis representation up to n basis elements. We refer to [33] for more details on the algorithm and its extensions.

Chapter 3

Tree-Structure Algorithm

In this chapter we will present the core of the thesis: the tree-structure algorithm (TSA, [3]) to solve the HJB equation (2.9). We will present the main features of the algorithm and some improvements in order to reduce dimensionality problems arising from the tree structure. Finally, we present the extension of the algorithm to high-order schemes and some numerical tests to show the efficiency of the algorithm.

3.1 Hamilton-Jabobi-Bellmann on a tree structure

The DP approach for the numerical approximation of viscosity solutions of the HJB equation is typically based on a time discretization which is projected on a fixed state-space grid of the numerical domain. The choice of the numerical domain is already one bottleneck of the method. In fact, although the theory is valid in the whole space $\mathbb{R}^d$ for computational reasons we need to restrict to a compact set in $\mathbb{R}^d$ which should be large enough to include all the possible trajectories. That also yields the selection of some boundary conditions which are not trivial.

In this section we will provide a novel algorithm which does not require a state-space grid and therefore avoids

(i) the choice of the numerical domain,

(ii) the computation of polynomial interpolation,

(iii) the selection of boundary conditions,

and finally it will allow to solve problems for high dimension, such as $d \gg 5$. Note that dimension 5 was the maximum dimension for SL-schemes based on a grid on a standard computer (see e.g. [5]).

3.1.1 Construction of the tree data structure

We build the nodes tree $\mathcal{T}$ starting from a given initial condition x and following directly the dynamics in (2.1) discretized by e.g. Euler method. Since we only discretize in time, we set a temporal step Δt which divides the interval $[t, T]$ into $\overline{N}$ subintervals. We note that $\mathcal{T} := \cup_{j=0}^{\overline{N}} \mathcal{T}^j$, where each $\mathcal{T}^j$ contains the nodes of the tree correspondent to time t_j. The first level $\mathcal{T}^0 = \{x\}$ is simply given by the initial condition x. To compute the other levels we suppose to discretize the control domain U with step-size Δu. The control set U is a subset in $\mathbb{R}^m$ and in particular we will consider U as a hypercube, discretized in all directions with constant step-size Δu, obtaining $U^{\Delta u} = \{u_1, ..., u_M\}$. To ease the notation in the sequel we continue to denote by U the discrete set of controls. Then, starting from the initial condition x, we consider all the nodes obtained following the dynamics (2.1) discretized using e.g. an explicit Euler scheme with different discrete controls $u_j \in U$

$$\zeta_j^1 = x + \Delta t \, f(x, u_j, t_0), \qquad j = 1, \ldots, M.$$

Therefore, we have $\mathcal{T}^1 = \{\zeta_1^1, \ldots, \zeta_M^1\}$. We note that all the nodes can be characterized by their $n-$th *time level*, as in the following definition.

Definition 3.1.1. *The general n-th level of the tree will be composed by M^n nodes denoted by*

$$\mathcal{T}^n = \{\zeta_i^{n-1} + \Delta t f(\zeta_i^{n-1}, u_j, t_{n-1}),\, j = 1, \ldots, M,\, i = 1, \ldots, M^{n-1}\}.$$

We show in the left panel of Figure 3.1 the structure of the whole tree $\mathcal{T}$. All the nodes of the tree can be shortly defined as

$$\mathcal{T} := \{\zeta_j^n,\, j = 1, \ldots, M^n,\, n = 0, \ldots, \overline{N}\},$$

where the nodes ζ_i^n are the result of the dynamics at time t_n with the controls $\{u_{j_k}\}_{k=0}^{n-1}$:

$$\begin{aligned}\zeta_{i_n}^n &= \zeta_{i_{n-1}}^{n-1} + \Delta t f(\zeta_{i_{n-1}}^{n-1}, u_{j_{n-1}}, t_{n-1}) \\ &= x + \Delta t \sum_{k=0}^{n-1} f(\zeta_{i_k}^k, u_{j_k}, t_k),\end{aligned}$$

with $\zeta^0 = x$, $i_k = \left\lceil \frac{i_{k+1}}{M} \right\rceil$ and $j_k \equiv i_{k+1} \text{mod } M$, where $\lceil \cdot \rceil$ is the ceiling function. We note that $\zeta_i^k \in \mathbb{R}^d, i = 1, \ldots, M^k$. On the right panel of Figure 3.1 we show the path to reach for instance ζ_{26}^4 if the control set contains only three elements. We, again, would like to emphasize that the domain is not chosen a priori, but constructed following the dynamics.

In what follows we provide two remarks about the properties of the tree $\mathcal{T}$ under some particular assumptions on the dynamics f.

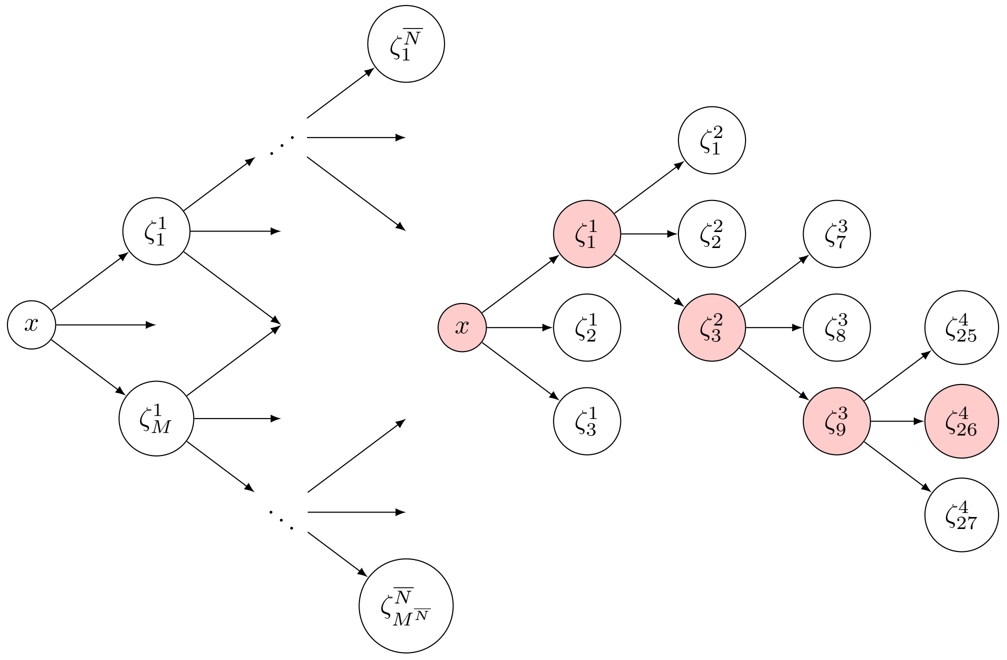


Figure 3.1: Example of the tree $\mathcal{T}$ (left), path to reach ζ_{26}^4 starting from the initial condition x with $U = \{u_1, u_2, u_3\}$ (right).

Remark 3.1.1. *Let us suppose that the dynamics is affine with respect to u and that $u \in [u_{min}, u_{max}] \subset \mathbb{R}$, e.g. the following decomposition holds true*

$$f(x, u, t) = f_1(x, t)u + f_2(x, t).$$

Then, all the nodes in $\mathcal{T}^n$ will lie on the segment with extremal points given by the controls at the boundary $\partial U = \{u_1 = u_{min}, u_M = u_{max}\}$. Specifically,

$$\textit{if } z \in \mathcal{T}^n \textit{ this implies } z \in [\zeta_{i_1}^n, \zeta_{i_M}^n]$$

where $\zeta_{i_1}^n$ and $\zeta_{i_M}^n$ are obtained by using the control u_1 and u_M respectively.

Remark 3.1.2. *Let us suppose that the dynamics is monotone with respect to $u \in [u_{min}, u_{max}] \subset \mathbb{R}$:*

$$\min_{\widetilde{u} \in \{u_{min}, u_{max}\}} f_j(x, \widetilde{u}, t) \leq f_j(x, u, t) \leq \max_{\widetilde{u} \in \{u_{min}, u_{max}\}} f_j(x, \widetilde{u}, t),$$
$$\forall u \in [u_{min}, u_{max}],\ j = 1, \ldots, d.$$

Then the nodes of the tree will belong to a box with vertices given by the coordinates of the nodes obtained with the extremal controls u_{min} and u_{max} as follows:

$$\min_{\bar{i} \in \{i_1, i_M\}} \zeta_{\bar{i}}^n \leq \zeta_i^n \leq \max_{\bar{i} \in \{i_1, i_M\}} \zeta_{\bar{i}}^n, \quad i \in \{i_1, \ldots, i_M\},$$

where the last inequality holds component-wise.

3.1.2 Approximation of the value function

The numerical value function $V(x,t)$ will be computed on the tree nodes in space, whereas in time it will be approximated as a piecewise constant function, *i.e.*

$$V(x,t) = V^n(x) \quad \forall x \in \mathcal{T}, t \in [t_n, t_{n+1}),$$

where $t_n = t + n\Delta t$.

We note that we start to approximate the value function once the tree $\mathcal{T}$ has been already built. Then, we will be able to approximate the value function $V^n(x_i + \Delta t f(x_i, u, t_n))$ in (2.21) without the use of an interpolation operator on a grid. The reason is that we build our domain according to all the possible directions of the dynamics for a discrete set of controls and, as a consequence, all the nodes $x_i + \Delta t f(x_i, u, t_n)$ will belong to the grid. It is now straightforward to evaluate the value function. The TSA defines a grid $\mathcal{T}^n = \{\zeta_j^n,\, j = 1, \ldots, M^n\}$ for $n = 0, \ldots, \overline{N}$, we can approximate (2.8) as follows:

$$\begin{cases} V^n(\zeta_i^n) = \min\limits_{u \in U}\{e^{-\lambda \Delta t} V^{n+1}(\zeta_i^n + \Delta t f(\zeta_i^n, u, t_n)) + \Delta t\, L(\zeta_i^n, u, t_n)\}, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \zeta_i^n \in \mathcal{T}^n\,, n = \overline{N}-1, \ldots, 0, \\ V^{\overline{N}}(\zeta_i^{\overline{N}}) = g(\zeta_i^{\overline{N}}), \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \zeta_i^{\overline{N}} \in \mathcal{T}^{\overline{N}}. \end{cases} \tag{3.1}$$

We note that the minimization is computed by comparison on the discretized set of controls U. We refer to [11, 40] for a more sophisticated approach to compute the minimum in (2.21).

Remark 3.1.3. *If the dynamics* (2.1) *is autonomous, the evolution of the dynamics will not depend explicitly on t_n and the problem can be simplified since the argument of the minimization in* (3.1) *will be*

$$e^{-\lambda \Delta t} V^{n+1}(\zeta + \Delta t f(\zeta, u)) + \Delta t\, L(\zeta, u, t_n).$$

At time t_n we have n levels of the tree on the left and $\overline{N} - n$ levels on the right (till $t_{\overline{N}}$). Since the computation is going backward, to compute the value function at time t_n, we need to do $\overline{N} - n$ steps in time starting from the final condition at time T. Once we know V^n, this information can also be interpreted as a final condition for the sub-tree $\cup_{k=0}^n \mathcal{T}^k$ and, since the dynamics is autonomous, we can proceed backward computing V^{n-1} for the nodes belonging to all the $k-$th time levels, for $k \leq n-1$. Indeed the nodes $\zeta + \Delta t f(\zeta, u)$ do not depend explicitly on the time and they can be involved in the computation of the value function at different time steps (this is not the case for a non-autonomous dynamics). Thus, we will proceed as follows: first we impose the final cost g on the whole tree, then we start computing the value function backward. This procedure leads to a more extensive knowledge of the value function on the tree.

3.2 Hints on the algorithm

In this section we will provide further details on the implementation of the method proposed in Section 3.1. We will explain how to reduce the number of tree nodes to make the problem feasible, compute the feedback control and recall the whole procedure.

3.2.1 Pruning the tree

The proposed method mitigates the curse of dimensionality and it allows to deal with problems in $\mathbb{R}^d$ with $d \gg 5$, which is absolutely not feasible with the classical approach. However, we still have dimensionality problem related to the amount of nodes in the tree $\mathcal{T}$. In fact, given $M > 1$ controls and $\overline{N}$ time steps, the cardinality of the tree is

$$|\mathcal{T}| = \sum_{i=0}^{\overline{N}} M^i = \frac{M^{\overline{N}+1} - 1}{M - 1},$$

which is infeasible due to the huge amount of memory allocations, if M or $\overline{N}$ are too large. Therefore, we suggest to select the nodes of the TSA neglecting those very close to each other, assuming that the value function will not be completely different on those nodes, e.g.

$$\zeta_i^n \approx \zeta_j^n \Longrightarrow V(\zeta_i^n) \approx V(\zeta_j^n).$$

This is a realistic assumption since the numerical value function is Lipschitz continuous as explained in Proposition 2.3.1. We can introduce the *pruning rule.*

Definition 3.2.1 (Pruning rule)**.** *Two given nodes ζ_i^n and ζ_j^n can be merged if*

$$\|\zeta_i^n - \zeta_j^n\| \le \varepsilon_{\mathcal{T}}, \qquad \text{with } n = 0, \dots, \overline{N}, \tag{3.2}$$

for a given threshold $\varepsilon_{\mathcal{T}} > 0$.

Specifically, if during the construction of the tree, a node ζ^{n-1} has as a son a new node ζ_j^n which verifies (3.2) with a certain ζ_i^n, then we will not add the new node to the tree and we will connect the node ζ^{n-1} with ζ_i^n. We cut the node which verifies the criteria before going on with the construction of the tree, in this way we avoid the sub-tree coming out from this node, saving a huge amount of memory.

The cut of the tree works as follows: during the construction of the n-th level, the new node will be compared with the previous nodes already computed at the same level n. If the new node ζ_j^n, whose father is ζ^{n-1}, satisfies the condition (3.2) with a node ζ_i^n, the new node will not be added to the tree and the adjacency list will be uploaded, connecting the node ζ^{n-1} to the node ζ_i^n. Figure 3.2 provides a graphic idea about the application of the

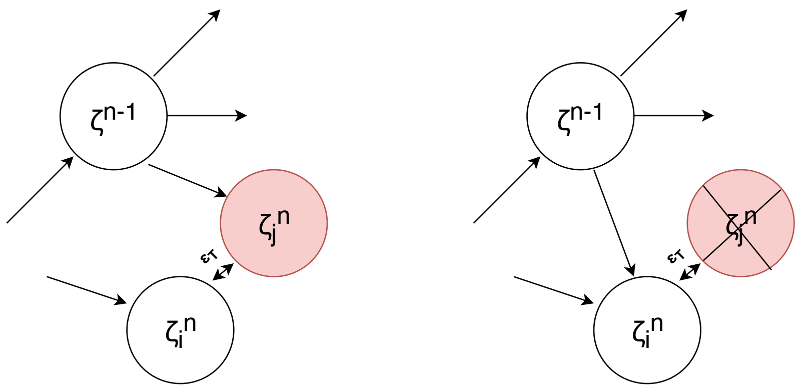


Figure 3.2: Pruning technique throughout the construction of the tree: when two nodes are very close (left), we link those nodes in order to prune the tree.

pruning criteria. The choice of the tolerance plays an important role: if $\varepsilon_\mathcal{T}$ is very small, the algorithm will be very slow, whereas if it is too large, we will not obtain an accurate approximation. A reasonable choice turns out to be $\varepsilon_\mathcal{T} = C\Delta t^2$, as shown in Chapter 4, where the reader will find a rigorous proof of this heuristic statement together with convergence results of the proposed method.

Remark 3.2.1 (Pruning rule in the autonomous case)**.** *If the dynamics is autonomous, as explained in Remark 3.1.3, we can extend the computation of the value function at time t_n even for nodes belonging to the subtree $\cup_{k=0}^n \mathcal{T}^k$. Therefore, we can extend the pruning criteria* (3.2) *as follows. Two given nodes ζ_i^n and ζ_j^m can be merged if*

$$\|\zeta_i^n - \zeta_j^m\| \le \varepsilon_\mathcal{T}, \qquad \text{with } n, m = 0, \ldots, \overline{N}, \tag{3.3}$$

for a given threshold $\varepsilon_\mathcal{T} > 0$.

Remark 3.2.2 (Efficient Pruning)**.** *The computation of the distances among all the nodes would be very expensive, especially for high dimensional problems. Hence, we need an efficient algorithm to compute the distances quickly. One possible strategy is the Principal Analysis Component ([48],[39]). Our aim is to project the data onto a lower dimensional linear space such that the variance of the projected data is maximized. This can be done e.g. computing the Singular Value Decomposition of the data matrix and taking the first basis. Once we project the data, the distances will be computed in a lower dimension space and this turns out to accelerate the algorithm. We will explain better this point in the numerical simulations. It is also possible to consider directly a projected dynamical system and this will be the aim of Chapter 5.*

3.2.2 Feedback reconstruction and closed-loop control

During the computation of the value function, we store the control indices corresponding to the argmin in (3.1). Then starting from $\zeta_*^0 = x$, we follow the path of the tree to build the optimal trajectory $\{\zeta_*^n\}_{n=0}^{\overline{N}}$ in the following way

$$u_n^* := \arg\min_{u\in U} \left\{ e^{-\lambda\Delta t} V^{n+1}(\zeta_*^n + \Delta t f(\zeta_*^n, u, t_n)) + \Delta t\, L(\zeta_*^n, u, t_n) \right\}, \qquad (3.4)$$

$$\zeta_*^{n+1} \in \mathcal{T}^{n+1} \; s.t. \; \zeta_*^n \to^{u_n^*} \zeta_*^{n+1},$$

for $n = 0, \dots, \overline{N}-1$, where the symbol $\to^u$ stands for the connection of two nodes by the control u. We note that this is possible because we assume to consider the same discrete control set U for both HJB equation (3.1) and feedback reconstruction (3.4). In the next section we will present a technique which considers a finer control set for the feedback reconstruction.

3.2.3 Algorithm

In what follows we summarize the whole algorithm including the construction of the tree, the selection of the nodes and, finally, the approximation of the value function.

Algorithm 3 TSA algorithm with pruning

```
 1: $\mathcal{T}^0 \leftarrow x$
 2: for $n = 1, ..., \overline{N}$ do
 3:     for $u_j \in U$, $\zeta^{n-1} \in \mathcal{T}^{n-1}$ do
 4:         $\zeta_{new} = \zeta^{n-1} + \Delta t f(\zeta^{n-1}, u_j, t_{n-1})$
 5:         if $\|\zeta_{new} - \zeta\| > \varepsilon_{\mathcal{T}}, \forall \zeta \in \mathcal{T}$ then
 6:             $\mathcal{T}^n \leftarrow \zeta_{new}$
 7:             $\zeta^{n-1} \to^u \zeta_{new}$
 8:         else
 9:             $\overline{\zeta} = arg\,min_{\zeta\in\mathcal{T}} \|\zeta_{new} - \zeta\|$
10:             $\zeta^{n-1} \to^u \overline{\zeta}$
11: $V^{\overline{N}}(\zeta) = g(\zeta), \forall \zeta \in \mathcal{T}^{\overline{N}}$
12: for $n = \overline{N}-1, ..., 0$ do
13:     $V^n(\zeta^n) = \min_{\zeta^{n+1}: \zeta^n \to^u \zeta^{n+1}} \{e^{-\lambda\Delta t} V^{n+1}(\zeta^{n+1}) + \Delta t\, L(\zeta^n, u, t_n)\}, \quad \zeta^n \in \mathcal{T}^n.$
```

As one can see in Algorithm 3, we first start the construction of the tree $\mathcal{T}$ from 1 to step 10. We note that the pruning criteria is involved in the steps 5-10 of Algorithm 3. Clearly, a very small tolerance will not allow any selection of the nodes and we will work with a full tree. Finally, in step 11-12-13 we compute the approximation of the value function. In the last step, the computation of the value function $V^n(\zeta^n)$ can be extended to the nodes in the tree $\cup_{k=0}^n \mathcal{T}^k$ in the case of autonomous dynamics.

3.3 Post-processing: Feedback reconstruction

The Tree Structure Algorithm allows to get the synthesis of the feedback control directly by the computation of the numerical value function, as explained in Section 3.2.2. The computational cost for the construction of the full tree is exponential in the number of discrete controls, for this reason it is better to consider few controls for the tree construction and for the resolution of the HJB equation. Once obtained the value function on a tree-structure, it is reasonable to consider a post-processing procedure which takes into account a finer control set. This is possible thanks to the formula for the synthesis of the feedback control

$$u_n^* := \arg\min_{u \in \widetilde{U}} \left\{ e^{-\lambda \Delta t} V^{n+1}(x + \Delta t f(x, u, t_n)) + \Delta t\, L(x, u, t_n) \right\}, \qquad (3.5)$$

where the *argmin* is computed on a finer set $\widetilde{U}$ with respect to the initial set U. This minimization can be computed again by comparison, but we need to reintroduce an interpolation step on scattered data. In low dimension (*i.e.* two or three) one can consider a Delaunay triangulation of the data and then perform an interpolation on the triangulation. In high dimension the triangulation becomes unfeasible and one has to proceed in different ways, for example via kernel methods ([52]). In this section we focus on the low dimensional case, while the high dimensional case will be addressed to the next future. In Algorithm 4 we present a method for the feedback reconstruction based on a minimization by comparison on a finer control set.

Algorithm 4 Feedback reconstruction via comparison on a finer control set

```
1: Computation of the tree $\mathcal{T}$ and value function $\{V^k\}_k$ with control set $U$
2: Fix a new control set $\widetilde{U} \supset U$ and $\zeta_*^0 = x$
3: for $n = 0, ..., \overline{N} - 1$ do
4:     for $u_j \in \widetilde{U}$ do
5:         $\zeta_j = \zeta_*^n + \Delta t f(\zeta_*^n, u_j, t_n)$
6:         Compute $V(\zeta_j, t_{n+1})$ via scattered interpolation with $(\mathcal{T}^{n+1}, V^{n+1})$
7:     $u_n^* := \arg\min_{u_j \in \widetilde{U}} \left\{ e^{-\lambda \Delta t} V(\zeta_j, t_{n+1}) + \Delta t\, L(\zeta_*^n, u_j, t_n) \right\}$
8:     $\zeta_*^{n+1} = \zeta_*^n + \Delta t f(\zeta_*^n, u_n^*, t_n)$
```

The interpolation on scattered data can be computed via the MATLAB function scatteredInterpolant. If the dynamics f is autonomous, by Remark 3.1.3 we know that we can compute at time t_n the value function on the subtree $\cup_{k=0}^n \mathcal{T}^k$. In this case, in step 6 we can compute the scattered interpolation with $\cup_{k=0}^n (\mathcal{T}^k, V^k)$, guaranteeing more information for a more efficient interpolation.

Now let us consider a dynamics f affine in the control $u \in \mathbb{R}$. By Remark 3.1.1 we know that all the tree sons of a node lay on a segment. In this case we can apply one dimensional interpolation, for instance quadratic interpolation if we consider three discrete controls for each iteration. The quadratic interpolation is a good choice in the Linear Quadratic Regulator case since we know that the value function is quadratic and then we do not introduce interpolation error in this case. Moreover let us suppose that the running cost L is of the form $L(x,u,t) = g(x,t) + \gamma|u|^2 + \delta u$. In Algorithm 5 we describe this procedure based on a quadratic interpolation, fixing $\lambda = 0$ for simplicity. In step 9 the operator $\mathcal{P}_U$ stands for the projection operator onto the set U. We will use and compare these two techniques in Chapter 5.5.1, where we will consider the optimal control for the heat equation.

Algorithm 5 Feedback reconstruction via quadratic interpolation

```
 1: Computation of the tree $\mathcal{T}$ and value function $\{V^k\}_k$ with control set $U = \{u_1, u_2, u_3\}$.
 2: $\zeta_*^0 = x$
 3: for $n = 0, ..., \overline{N} - 1$ do
 4:     for $u_j \in U$ do
 5:         $\zeta(u_j) = \zeta_*^n + \Delta t f(\zeta_*^n, u_j, t_n)$
 6:         Compute $V(\zeta(u_j), t_{n+1})$ via scattered interpolation with $(\mathcal{T}^{n+1}, V^{n+1})$
 7:     $V(\zeta(u), t_{n+1}) \approx au^2 + bu + c, \quad \forall u \in [u_1, u_3]$
 8:     if $a + \Delta t\, \gamma > 0$ then
 9:         $u_n^* = \mathcal{P}_{[u_1,u_3]}\left(-\frac{b+\Delta t\,\delta}{2(a+\Delta t\,\gamma)}\right)$
10:     else
11:         $u_n^* = \underset{u_i \in \{u_1,u_3\}}{\arg\min} \{V(\zeta(u_i), t_{n+1}) + \Delta t\, L(\zeta_*^n, u_i, t_n)\}$
12:     $\zeta_*^{n+1} = \zeta_*^n + \Delta t f(\zeta_*^n, u_n^*, t_n)$
```

3.4 Extension to high-order schemes

In the previous sections we consider the TSA using a forward Euler scheme which leads to a first order convergence, as we will show later on. In this section we will show how our approach can be easily extended to high-order schemes improving previous results. In what follows, we set $\lambda = 0$ in (2.29), without loss of generality. For more details on the topic, we refer to [4].

Let us consider a high-order approximation scheme for the cost functional (2.29) and for the dynamics (2.1) under the assumptions on L, f and g provided in the previous sections. As already suggested in [25] for the infinite horizon problem, we introduce a one-step approximation for the dynamics (2.1) as follows

$$\begin{cases} y^{n+1} = y^n + \Delta t \Phi(y^n, \mathbf{U}, t_n, \Delta t), \\ y^0 = x, \end{cases} \tag{3.6}$$

where the admissible control matrix $\mathbf{U} \in U_{\Delta t} \subset U \times U \ldots \times U \in \mathbb{R}^{M\times(q+1)}$ with $U \subset \mathbb{R}^M$ the discretized control set and $q+1$ is the number of stages of the numerical method for the ODE (it is also possible to consider a time dependence of U as in [25] but we will avoid this complication here). We denote by u_i^n the $i-$th control of U for the $n-$th column of $\mathbf{U}$.

We further assume that the function Φ in (3.6) is consistent:

$$\lim_{\Delta t\to 0} \Phi(x, \bar{\mathbf{u}}, t, \Delta t) = f(x, \bar{u}, t), \tag{3.7}$$

where $\bar{\mathbf{u}} = (\bar{u}, \ldots, \bar{u}) \in \mathbf{U}$ for $\bar{u} \in U$ and Lipschitz continuous:

$$|\Phi(x, \mathbf{U}, t, \Delta t) - \Phi(y, \mathbf{U}, t, \Delta t)| \leq L_\Phi |x-y|, \tag{3.8}$$

for any admissible set U and $0 < \Delta t < \overline{\Delta t}$. Under these assumptions the scheme (3.6) is convergent. Then, we consider the approximation of the cost functional

$$J_{x,t}^{\Delta t}(\mathbf{U}) = \Delta t \sum_{m=n}^{N-1} \sum_{i=0}^{q} w_i L(y^{m+\tau_i}, u_i^m, t_{m+\tau_i}) + g(y^N), \tag{3.9}$$

where τ_i and w_i are the nodes and weights of the quadrature formula satisfying:

$$0 \leq \tau_i \leq 1, \qquad \omega_i \geq 0, \qquad \sum_{i=0}^{q} w_i = 1.$$

We will suppose that the following "order assumptions" hold:

1. for any initial condition $x \in \mathbb{R}^d$ and any measurable $u : [0, \Delta t) \to U$, there exists an admissible matrix $\mathbf{U} \in U_{\Delta t}$ and two positive constants K_1 and K_2 such that

$$|y(\Delta t, u; x) - x - \Delta t \Phi(x, \mathbf{U}, \Delta t)| \leq K_1 \Delta t^{p+1}, \tag{3.10}$$

$$\left| \int_0^{\Delta t} L(y(s), u(s), s) ds - \Delta t \sum_{i=0}^{q} w_i L(y^{\tau_i}, u_i, t_{\tau_i}) \right| \leq K_2 \Delta t^{p+1}, \tag{3.11}$$

2. for any initial condition $x \in \mathbb{R}^d$ and any admissible matrix $\mathbf{U} \in U_{\Delta t}$, there exists a measurable $u : [0, \Delta t) \to U$ such that (3.10) and (3.11) hold.

In some cases it is possible to define explicitly the set $U_{\Delta t}$ verifying conditions (3.10) and (3.11). For more details we refer to [30] and [37]. In what follows we are going to consider affinely controlled system and we assume C^p regularity for the data and for the optimal control.

Therefore, we define the numerical value function as

$$V(x,t) = \inf_{\mathbf{U}} J^{\Delta t}_{x,t}(\mathbf{U}). \tag{3.12}$$

Following [25], it is possible to prove the extended DPP which reads:

$$V(x,t) = \inf_{\mathbf{U}} \left\{ \Delta t \sum_{i=0}^{q} w_i L(y^{n+\tau_i}, u_i^n, t_{n+\tau_i}) + V(y^{n+1}, t_{n+1}) \right\}. \tag{3.13}$$

Under our assumptions on L and f, it is easy to check that V is Lipschitz-continuous and bounded. This will guarantee the convergence of the numerical scheme.

Note that for $q = 0$ in (3.13) we obtain the standard formulation with Euler method:

$$V(x,t) = \min_{u\in U} \left\{ \Delta t\, L(x,u,t) + V(x + \Delta t f(x,u,t), t + \Delta t) \right\}.$$

For Heun's scheme, e.g. $q = 1$, equation (3.13) becomes

$$\begin{aligned} V(x,t) = \min_{(\bar u_0, \bar u_1)\in U\times U} & \Big\{ \tfrac{\Delta t}{2} (L(x, \bar u_0, t) + \\ & L(x + \Delta t \Phi(x, \{\bar u_0, \bar u_1\}, t, \Delta t), \bar u_1, t + \Delta t)) + \\ & + V(x + \Delta t \Phi(x, \{\bar u_0, \bar u_1\}, t, \Delta t), t + \Delta t) \Big\}, \end{aligned} \tag{3.14}$$

where

$$\Phi(x, \{\bar u_0, \bar u_1\}, t, \Delta t) = \frac{1}{2} \left(f(x, \bar u_0, t) + f(x + \Delta t f(x, \bar u_0, t), \bar u_1, t + \Delta t) \right). \tag{3.15}$$

It is also possible to deal with implicit numerical schemes in equation (3.13) using e.g. the trapezoidal rule obtaining:

$$\begin{aligned} V(x,t) = \min_{(\bar u_0, \bar u_1)\in U\times U} & \Big\{ \tfrac{\Delta t}{2} (L(x, \bar u_0, t) + \\ & + L(y^{n+1}(\bar u_0, \bar u_1), \bar u_1, t + \Delta t)) + V(y^{n+1}(\bar u_0, \bar u_1), t + \Delta t) \Big\}, \end{aligned} \tag{3.16}$$

where $y^{n+1}(\bar u_0, \bar u_1)$ is obtained solving

$$y^{n+1}(\bar u_0, \bar u_1) = x + \frac{\Delta t}{2} \left(f(x, \bar u_0, t) + f(y^{n+1}(\bar u_0, \bar u_1), \bar u_1, t + \Delta t) \right). \tag{3.17}$$

It is clear that the cardinality of the tree $\mathcal{T}$ will significantly increase when dealing with high order schemes. Therefore, a pruning rule (3.2) is essential. We will see in Chapter 4 how to choose the pruning tolerance $\varepsilon_{\mathcal{T}}$ to insurance to the TSA the same order of convergence of the underlying numerical scheme.

3.5 Numerical tests

In this section we are going to apply the proposed algorithm to show the effectiveness of the method.

We will present six test cases. In the first we are able to compute the analytical solution of the HJB equation and, therefore, to compute the error with our method compared to the classical approach, see e.g. [27]. The second test concerns the well-known Van der Pol equation and we will compare our proposed algorithm with Model Predective Control. The third one is about a non-autonomous dynamics and we will explain better the differences with the autonomous case. Then, we present the results for two different linear PDEs which show the power of the method even for large-scale problems. Finally, we will show the efficiency of high order schemes in the case of an advection equation.

The numerical simulations reported in this paper are performed on a laptop with 1CPU Intel Core i5-3,1 GHz and 8GB RAM. The codes are written in C++.

3.5.1 Test 1: Comparison with an exact solution

In the first example we consider the following dynamics in (2.1)

$$f(x,u) = \begin{pmatrix} u \\ x_1^2 \end{pmatrix}, \; u \in U \equiv [-1,1], \tag{3.18}$$

where $x = (x_1, x_2) \in \mathbb{R}^2$. The cost functional in (2.29) is:

$$L(x,u,t) = 0, \qquad g(x) = -x_2, \qquad \lambda = 0, \tag{3.19}$$

where we only consider the terminal cost g. The corresponding HJB equation is

$$\begin{cases} -V_t + |V_{x_1}| - x_1^2 V_{x_2} = 0 & (x,t) \in \mathbb{R}^2 \times [0,T], \\ V(x,T) = g(x) \;, \end{cases} \tag{3.20}$$

where its unique viscosity solution reads

$$V(x,t) = -x_2 - x_1^2(T-t) - \frac{1}{3}(T-t)^3 - |x_1|(T-t)^2. \tag{3.21}$$

Furthermore, we set $T = 1$. Figure 3.3 shows the contour lines of the value function $V(x,t)$ for time instances $t = \{0, 0.5, 1\}$.

In this example, we compare the classical approach with the TSA algorithm proposed in Algorithm 1 using both strategies: (i) no selection of the nodes and (ii) applying criteria (3.2) to select the nodes as explained in Section 3.2. To perform a fair comparison we projected the value function computed with the classical method into the tree nodes. We note that it will not modify the accuracy of the classical approach since the interpolation has to be performed

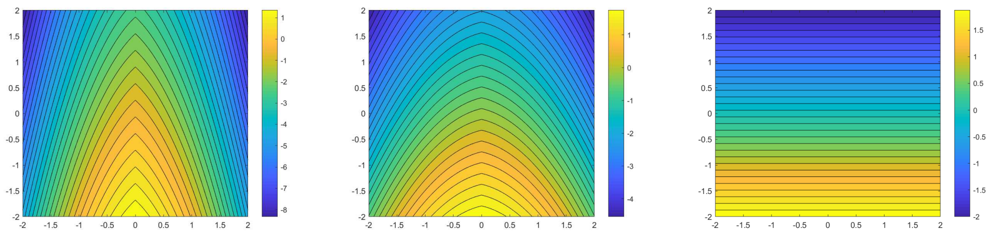


Figure 3.3: Test 1: Contour lines for (4.49) with $t = 0$ (left), $t = 0.5$ (middle) and $t = 1$ (right).

also on a structured grid. We compare the different approximations according to ℓ_2-relative error with the exact solution on the tree nodes

$$\mathcal{E}_2(t_n) = \sqrt{\frac{\sum\limits_{x_i \in \mathcal{T}^n} |v(x_i, t_n) - V^n(x_i)|^2}{\sum\limits_{x_i \in \mathcal{T}^n} |v(x_i, t_n)|^2}},$$

where $v(x_i, t_n)$ represents the analytical solution and $V^n(x_i)$ its numerical approximation.

In Figure 3.4, we show all the nodes of the tree $\mathcal{T}$ for the initial condition $x = (-0.5, 0.5)$, $\Delta t = 0.05$ and different choices of $\varepsilon_{\mathcal{T}} = \{0, \Delta t^2\}$. We note that there is a huge difference between the cardinality of the trees, that is $|\mathcal{T}| = 2097151$ when the tolerance is not applied whereas we have $|\mathcal{T}| = 3151$ for $\varepsilon_{\mathcal{T}} = \Delta t^2$.

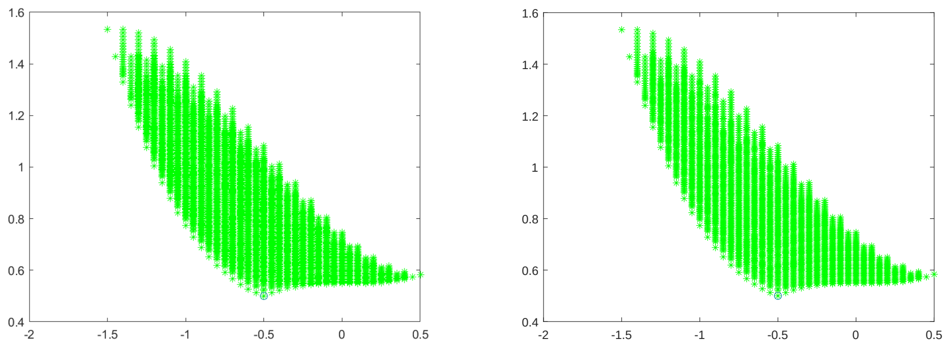


Figure 3.4: Test 1: Tree nodes without tolerance (left) and with tolerance equal to Δt^2 (right) for $x = (-0.5, 0, 5)$.

In Figure 3.5, we show the behaviour of the error $\mathcal{E}_2$ for two different initial conditions x. We note that its behaviour is very similar using both the classical approach and the TSA with or without the pruning criteria (3.2) for the nodes. As already mentioned, we would like to stress that the domain for the solution of the classical approach is chosen as large as possible to avoid that the boundary conditions are active, whereas with TSA we do not have this kind of problem, since the domain of the tree constructed according to the vector field. We note that to compute the value function in the classical approach we use the following step size: $\Delta x = \Delta t = 0.05$.

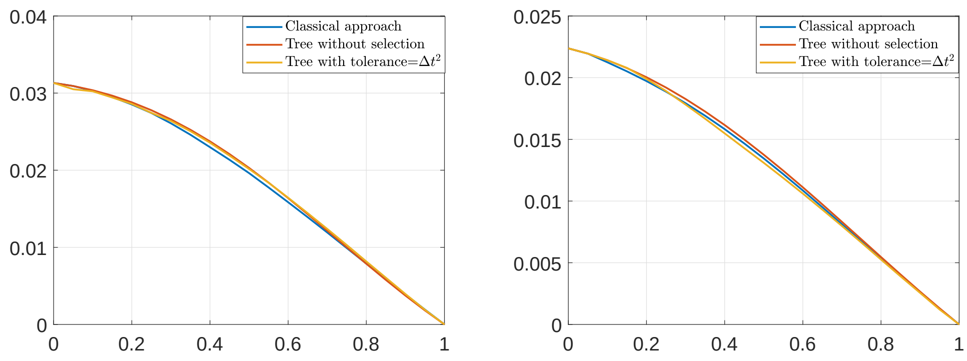


Figure 3.5: Test 1: Comparison of the different methods with initial datum $(-0.5, 0.5)$ (left) and with initial datum $(1, 1)$ (right) for each time instance (x-axis).

We will consider again this example in Chapter 4, where we will compute the order of convergence of the scheme.

3.5.2 Test 2: Van der Pol oscillator

In the second test case we consider the Van der Pol oscillator. The dynamics in (2.1) is given by

$$f(x,u) = \begin{pmatrix} x_2 \\ \omega(1-x_1^2)x_2 - x_1 + u \end{pmatrix} \qquad u \in U \equiv [-1,1]. \tag{3.22}$$

We note that the origin is a repulsive point for the uncontrolled dynamics in (3.22), e.g. $u = 0$, if $\omega \in (0, 2]$. For this example we consider $\omega = 0.15$ in (3.22). It is well-known that Van der Pol oscillator is characterized by its cycle limit as shown in Figure 3.11 with two different initial conditions.

In this example we want to minimize the following cost functional:

$$J_{x,t}(u) = \int_t^T \left(\delta_1 \|y(s)\|_2^2 + \gamma |u(s)|^2\right) \, ds + \delta_2 \|y(T)\|_2^2, \tag{3.23}$$

where $\delta_1, \delta_2, \gamma$ are positive constants.

Case 1 We consider the minimization of the terminal cost in (3.24), e.g. $\delta_1 = \gamma = 0$ and $\delta_2 = 1$. Let us consider $x = (-1, 1)$, $\Delta t = 0.05$ and $T = 1$. The error is computed with respect to the classical approach with a fine grid ($\Delta t = \Delta x = 0.002$).

We will consider Euler scheme with $U = \{-1, 1\}$ and the tolerance is set equal to $\varepsilon_{\mathcal{T}} = \Delta t^2$ with $|\mathcal{T}| = 37030$. In Figure 3.7 we compare the contour

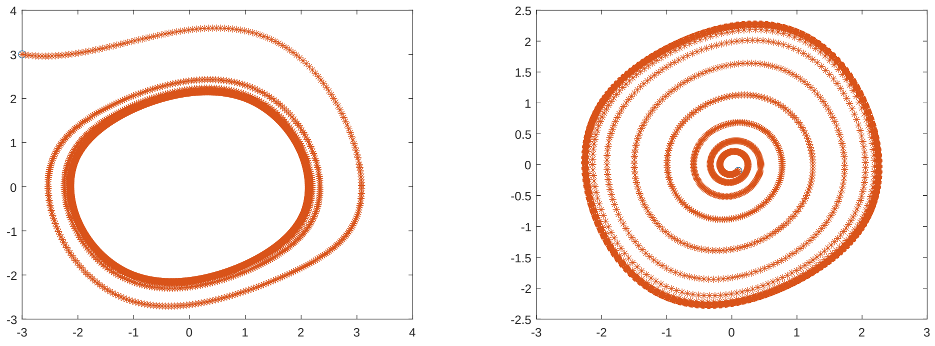


Figure 3.6: Test 2: Cycle limit for Van der Pol oscillator with initial point (-3,3) (left) and with initial point (0.1,-0.1) (right)

lines of the value function computed by the classical approach with a fine grid and the TSA. We note the approximations show the same behaviour. Furthermore, we mention that the contour line of the value functions are obtained by using MATLAB function `tricontour`, based on a Delaunay's triangulation of the scattered data. We remark that we can compute the value function $V^n(\zeta)$ for $\zeta \in \cup_{k=0}^n \mathcal{T}^k$ since the dynamics is autonomous.

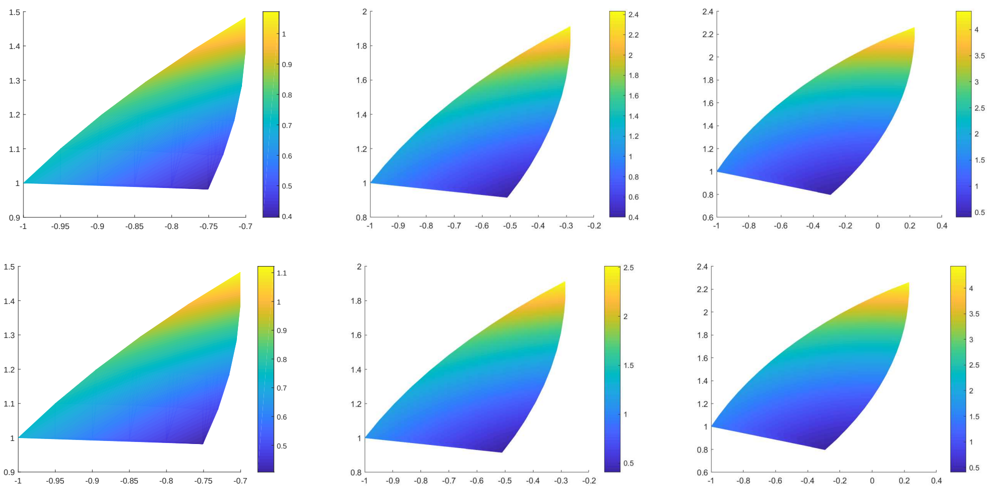


Figure 3.7: Test 2: Value function with the classical approach (top) on tree nodes at time $t = 0.25$ (left), $t = 0.5$ (middle) and $t = 0.75$ (right). Value function with the TSA (bottom) on tree nodes at time $t = 0.25$ (left), $t = 0.5$ (middle) and $t = 0.75$ (right)

The quality of the numerical approximation is confirmed by the error shown in Figure 3.8. As we can see, pruning the nodes does not influence the error. For each time step the error is below to 0.05 which leads to an accurate approximation of the value function.

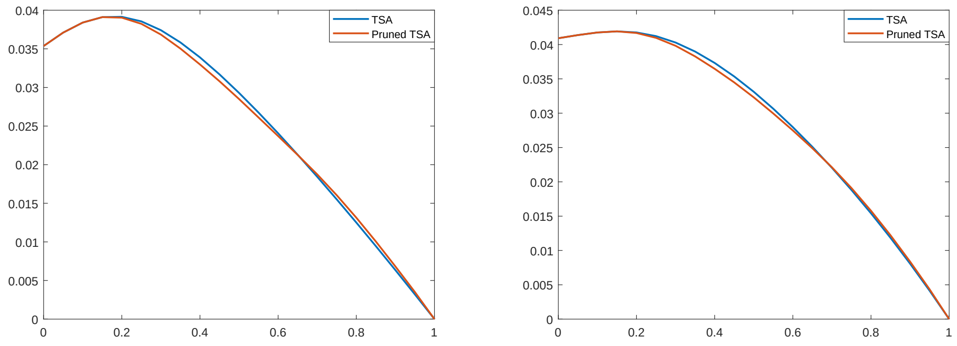


Figure 3.8: Test 2: Error in time with TSA without pruning and with pruning with tolerance $\varepsilon_{\mathcal{T}} = \Delta t^2$ for Case 1 (left) and Case 2 (right) with respect to a value function computed with the classical approach with a very fine grid.

Case 2 We consider the minimization of the cost functional in (3.24) with $\delta_1 = \delta_2 = 1$ and $\gamma = 0.01$. Furthermore we set the same initial condition, discretization step and tolerance as in the previous case. The contour lines of the value function are shown in Figure 3.9.

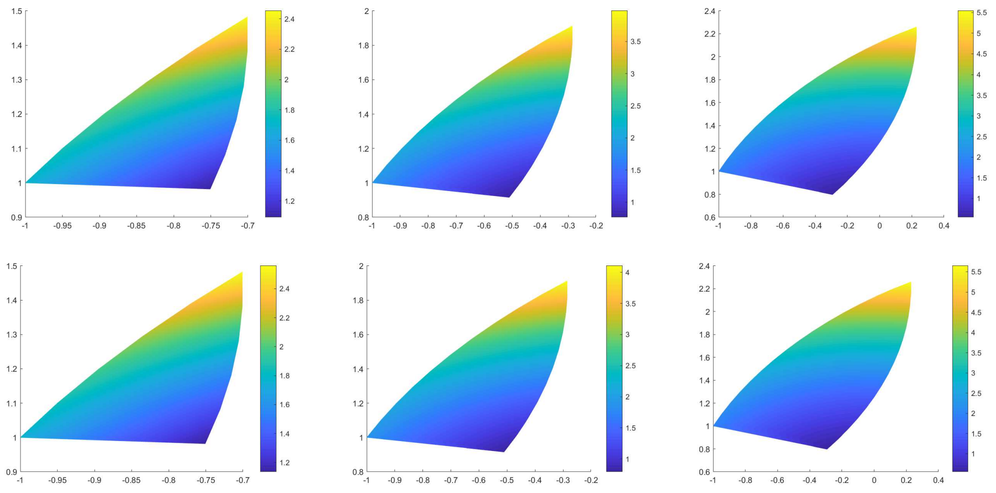


Figure 3.9: Test 2: Value function with the classical approach (top) on tree nodes at time $t = 0.25$ (left), $t = 0.5$ (middle) and $t = 0.75$ (right). Value function with the TSA (bottom) on tree nodes at time $t = 0.25$ (left), $t = 0.5$ (middle) and $t = 0.75$ (right)

We note that the results are very similar to the previous case. Our approach is robust with respect to different cost functionals and initial conditions. The right panel of Figure 3.8 shows the error for each time step considering the tree algorithm with and without nodal selection.

Case 3 The third case deals with a two dimensional control space, considering the parameter ω in (3.22) as a control, e.g. $\omega \in U$. Therefore, we consider as control variables $(\omega, u) \in U \times U$ in (3.22). In the cost functional (3.24) we consider again $\delta_1 = \gamma = 0.1$ and $\delta_2 = 1$, with $x = (-0.5, 0.5)$, $\Delta t = 0.05$ and $T = 1$. We consider two different choices for the control set: $U = [-2, 0]$ and $U = [-1, 1]$. The control set is discretized with step-size $\Delta u = 0.2$, obtaining altogether 100 discrete controls for both examples. In Figure 3.10 we show the results in both situations. We can observe that the tree has a different shape due to the different control space. Here, we have set the pruning criteria with $\varepsilon_{\mathcal{T}} = \Delta t^2$. Finally, we note that in both situations we are able to steer the solution to the origin.

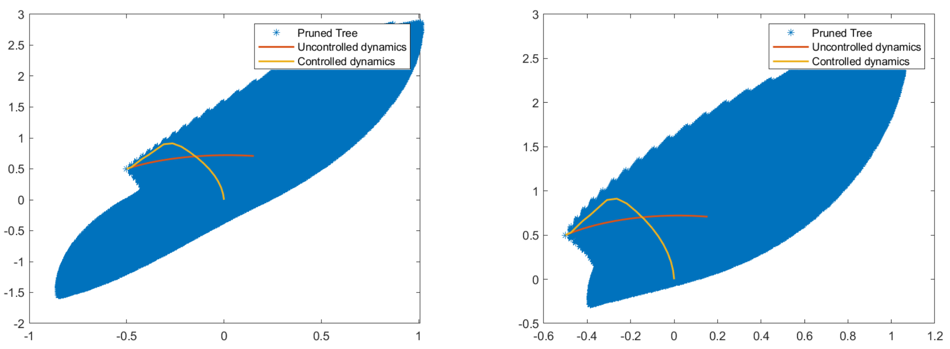


Figure 3.10: Test 2: Pruned tree with the uncontrolled and controlled dynamics with $U = [-2, 0]$ (left) and with $U = [-1, 1]$ (right)

Comparison with Model Predective Control Finally, we present a comparison with the MPC method for the Van Der Pol oscillator. MPC considers a localized version of the DPP since it deals only with a given initial condition and our aim is to show the efficiency of our TSA under the same settings.

For this example we consider $\omega = 0.15$ in (3.22), $T = 2, \Delta t = 0.05$. In this example we want to minimize the following cost functional:

$$J_{x,t}(u) = \int_t^T \left(\|y(s)\|_2^2 + \frac{1}{100}|u(s)|^2 \right) ds + \|y(T)\|_2^2. \tag{3.24}$$

In Figure 3.11 we show the optimal trajectory using the TSA and MPC method. TSA has been computed with 8 discrete controls in $[-1, -0.4]$. This control space has been selected according to the MPC results. As one can see, with TSA we are able to achieve the desired state faster than MPC method. The MPC method here has been computed with a prediction horizon of $N = 30$. We choose this setting for MPC[1] since it is well-known that the

[1]We used the MATLAB implementation provided in `http://numerik.mathematik.uni-bayreuth.de/~lgruene/nmpc-book/` correspondent to [44].

larger horizon the better the algorithm. The evaluation of the cost functional for the optimal trajectories with TSA is 0.0569, whereas 0.0695 with MPC. As expected, the TSA method has a lower value. In the right panel of Figure 3.11 we show the optimal controls with both methods.

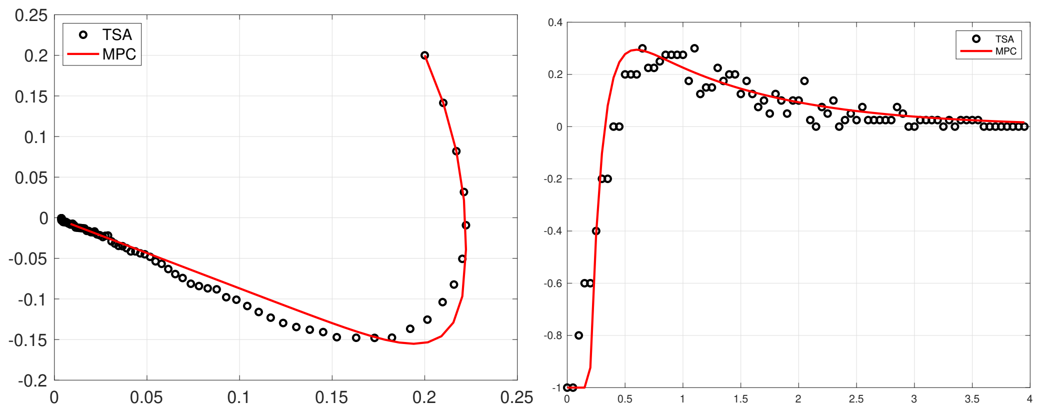


Figure 3.11: Test 2: Comparison of optimal trajectory (left) and optimal control (right).

3.5.3 Test 3: Damped harmonic oscillator

In this third example we consider a non-autonomous dynamical system: a damped oscillator driven by a sinusoidal external force. The dynamics in (2.1) is given by

$$f(x,u,t) = \begin{pmatrix} x_2 \\ -\omega x_2 - \omega^2 x_1 + \sin(\omega t) + u \end{pmatrix} \qquad u \in U \equiv [-1,1], \tag{3.25}$$

for $x = (x_1, x_2) \in \mathbb{R}^2$. In this example, we aim to show that our approach works also with non-autonomous dynamics. In this case we can not compute the value function $V^n(\zeta)$ on the sub-tree $\cup_{k=0}^n \mathcal{T}^k$, but only at the $n-$th time level $\mathcal{T}^n$ and we will apply the pruning rule (3.2). The uncontrolled dynamics ($e.g.\, u = 0$) converges asymptotically to the cycle limit:

$$\overline{x}_1(t) = \frac{1}{\omega^2}\sin(\omega t + \pi/2),\; \overline{x}_2(t) = \frac{1}{\omega}\cos(\omega t + \pi/2)\ .$$

We used the same cost functional of the previous case with $\delta_1 = \gamma = 0.1$, $\delta_2 = 1$. The parameters are set as follows: $\omega = \pi/2$, $x = (-0.5, 0.5)$, $U = \{-1, 0, 1\}, \Delta t = 0.05$, $T = 1$, $\varepsilon_{\mathcal{T}} = \Delta t^2$. The cardinality of tree in this case is 32468. In the left panel of Figure 3.12 we show the tree nodes and the optimal trajectory computed with Algorithm 1 and the uncontrolled solution. To show the quality of the controlled solution we evaluate the cost functional for each time step as shown in the right panel of Figure 3.12. As expected the controlled trajectory is always below the uncontrolled one. In order to further

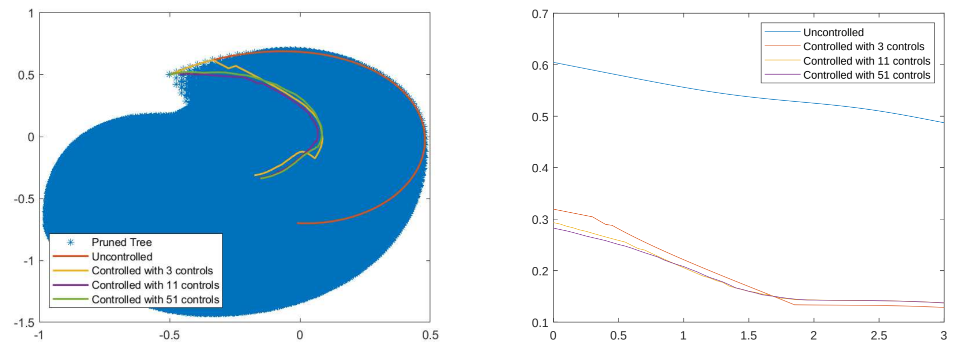


Figure 3.12: Test 3: Pruned tree with the uncontrolled and controlled dynamics (left) and comparison of the cost functional on time varying the number of discrete controls (right)

show the effectiveness of the pruning criteria we have increased the number of controls up to $M = 51$ and the horizon up to $T = 3$. Again, this would not be possible without a pruning criteria due to the dimension of the tree.

3.5.4 Test 4: Heat equation

The fourth example concerns the control of a PDE. In the first three examples we showed the accuracy of our method with respect to existing methods for low-dimensional problems. In what follows we would like to give an idea of how the proposed method can work in higher dimension.

We want to study the following heat equation:

$$\begin{cases} y_t = \sigma y_{xx} + y_0(x)u(t) & (x,t) \in \Omega \times [0,T]\,, \\ y(x,t) = 0 & (x,t) \in \partial\Omega \times [0,T]\,, \\ y(x,0) = y_0(x) & x \in \Omega\,, \end{cases} \tag{3.26}$$

where the state lies in an infinite-dimensional Hilbert space (see e.g. [24]). Here, we consider the term $y_0(x)u(t)$ to provide a spatial dependence to the control input. This is a particular choice, but the algorithm has no restrictions on more general shape functions. To write equation (3.26) in the form (2.1) we use the centered finite difference method which leads to the following ODEs system

$$\dot{y}(t) = Ay(t) + Bu(t), \tag{3.27}$$

where the matrix $A \in \mathbb{R}^{d\times d}$ is the so called *stiffness* matrix whereas the vector $B \in \mathbb{R}^n$ is given by $(B)_i = y_0(x_i)$ for $i = 1, \dots, n$ and x_i is the spatial grid with constant step size Δx. The cost functional we want to minimize reads:

$$J_{y_0,t}(u) = \int_t^T \left(\delta_1 \|y(s)\|_2^2\, dx + \gamma |u(s)|^2\right)\, ds + \|y(T)\|_2^2,$$

where $y(t)$ is the solution of (4.51), $u(t)$ is taken in the admissible set of controls $\mathcal{U} = \{u : [0,T] \to [-1,1]\}$ and $\Omega = [0,1]$. We set $\delta_1 = 1$ and $\gamma = 0.01$.

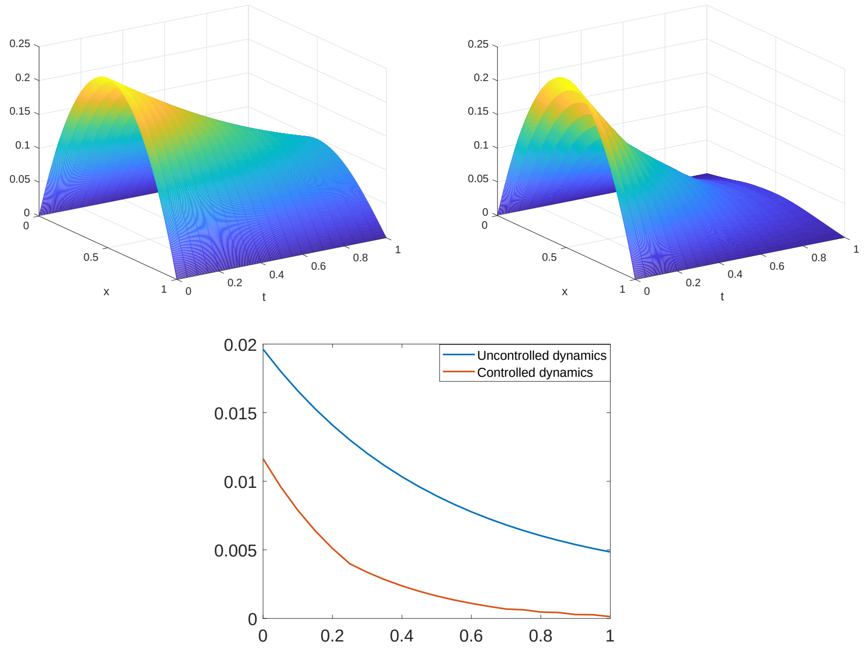


Figure 3.13: Test 4 (smooth initial condition): Uncontrolled solution (top left), optimal control solution (top right), time comparison of the cost functional of the uncontrolled solution and controlled solution (bottom).

Smooth initial condition In the numerical approximation of (3.26) we consider $y_0(x) = -x^2 + x$, $\Delta x = 10^{-3}$, $\Delta t = 0.05$, $T = 1$ and $\sigma = 0.1$. The dimension of the problem is $d = 1000$. We use an implicit Euler scheme to integrate the system (4.51) and guarantee its stability. We note that the use of a one step implicit is straightforward even if we have introduced an explicit scheme in the previous sections.

The solution of the uncontrolled problem (3.26) with $u(t) \equiv 0$ is shown in the top-left panel of Figure 5.4. In the top-right we show the solution of the controlled problem where the value function is computed with Algorithm 1 and the control is computed as explained in (5.6). We note that feedback control was computed with the discrete control set $U = \{-1, 0, 1\}$ as for the value function. It is extremely interesting to show that we are able to compute the value function for (3.26) in dimension 1000. Finally in the bottom panel of Figure 5.4 we show the time behaviour of the cost functional for the uncontrolled and the controlled solution. As expected, the cost functional of the latter is lower.

Non-smooth initial condition In this example we consider the following non-smooth initial $y_0(x) = \chi_{[0.25,0.75]}(x)$, where $\chi_\omega(x)$ is the characteristic function in the domain ω, whereas the other parameters are set as in the previous case.

As one can see from Figure 3.14, we are able to approximate the control

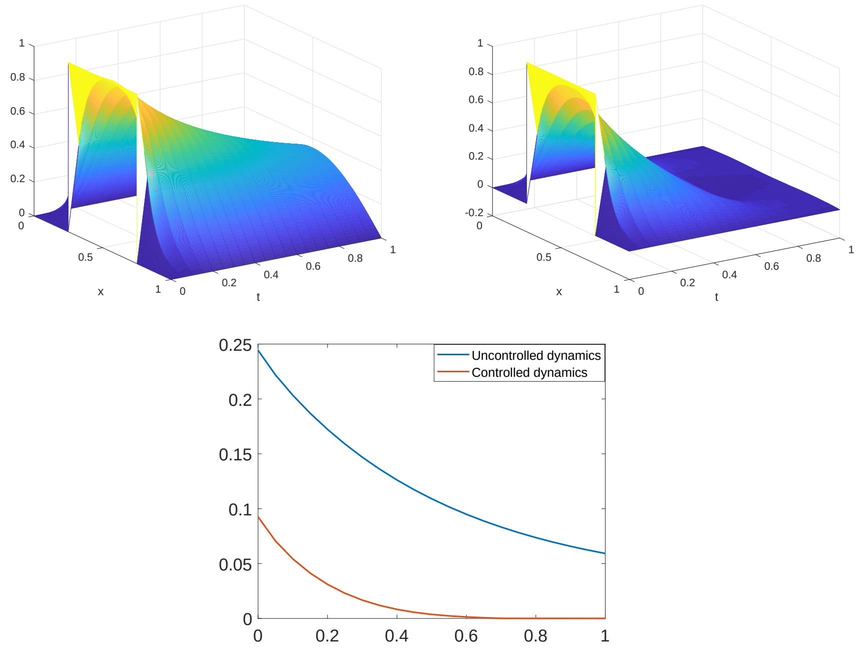


Figure 3.14: Test 4 (non-smooth initial condition): Uncontrolled solution (top-left), optimal control solution (top-right), time comparison of the cost functional of the uncontrolled solution and controlled solution (bottom).

problem even if the initial condition is non-smooth. We note that, although the simple diffusive properties of the problem, a model reduction approach will not be able to reconstruct such initial condition with a few number of basis functions. Therefore it will not be possible to solve this problem with a classical approach. This again shows the effectiveness of the method.

3.5.5 Test 5: Wave equation

For this example we consider a hyperbolic PDE, the wave equation which reads:

$$\begin{cases} w_{tt} = c\, w_{xx} + \chi_\omega(x) u(t) & (x,t) \in \Omega \times [0,T]\ , \\ w(x,t) = 0 & (x,t) \in \partial\Omega \times [0,T]\ , \\ w(x,0) = w_0(x)\ ,\ w_t(x,0) = w_1(x) & x \in \Omega\ , \end{cases} \tag{3.28}$$

where ω is a subset of Ω. For all initial data $(w_0, w_1) \in H_0^1(\Omega) \times L^2(\Omega)$ and every $u(t) \in L^2(0,T)$, there exists a unique solution $w \in C^0(0,T; H_0^1(\Omega)) \cap C^1(0,T; L^2(\Omega)) \cap C^2(0,T; H^{-1}(\Omega))$ of the Cauchy problem (3.28). We refer to [24] for more details about this equation. We can rewrite the wave equation in the following compact form

$$\dot{y}(t) = Ay(t) + Bu(t)\ ,$$

defining

$$y(t) = \begin{pmatrix} w(t) \\ w_t(t) \end{pmatrix}, \quad A = \begin{pmatrix} 0 & I \\ c\,\partial_x^2 & 0 \end{pmatrix}, \quad Bu(t) = \begin{pmatrix} 0 \\ \chi_\omega(x)u(t) \end{pmatrix}. \tag{3.29}$$

Again we apply an implicit Euler scheme to avoid narrow CFL conditions. We want to minimize the following cost functional

$$J_{y_0,t}(u) = \int_0^T \left(\varphi(\|y(s)\|_2^2) + \gamma|u(s)|^2\right) ds + \varphi(\|y(T)\|_2^2)\,,$$

with $w_0(x) = \sin(\pi x)$, $w_1(x) = 0$, $\gamma = 0.01$, $T = 1$, $c = 0.5$, $\Omega = (0,1)$ and $\omega = (0.4, 0.6)$, $\Delta x = 10^{-3}, \Delta t = 0.05$. We note that the dimension of the semi-discrete problem is $d = 2000$.

Quadratic cost functional We first consider a standard tracking problem e.g. $\varphi(x) = x$ in the cost functional. In Figure 3.15 we show the uncontrolled solution in the top left panel and the controlled solution in the top-right. A comparison of the evaluations of the cost functional is given in the bottom panel. As expected the controlled solution is below the uncontrolled one for each time instance. This shows the capability of the method for high dimensional problem even for hyperbolic equations.

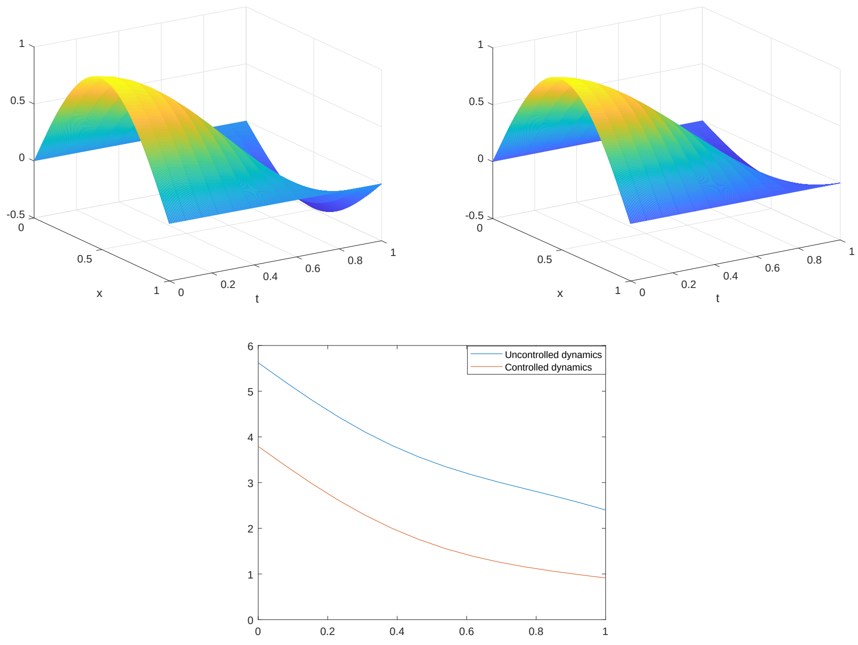


Figure 3.15: Test 5: Uncontrolled solution (top-left), optimal control solution (top-right), time comparison of the cost functional of the uncontrolled solution and controlled solution (bottom).

Non-quadratic cost functional Now, we consider a more complicated example which deals with a non-quadratic cost functional. Let us consider for example the following cost functional where

$$\varphi(x) = \begin{cases} sin(\pi|x|) & |x| \leq 0.5 \, , \\ 1 & 0.5 < |x| \leq 1 \, , \\ (|x|-1)^2 + 1 & |x| > 1 \, , \end{cases}$$

as shown in the top-left panel of Figure 3.16. We consider the same parameters as in the previous case, which lead to the same uncontrolled solution as shown in the top-left panel of Figure 3.15. In the top-right of Figure 3.16 one can see

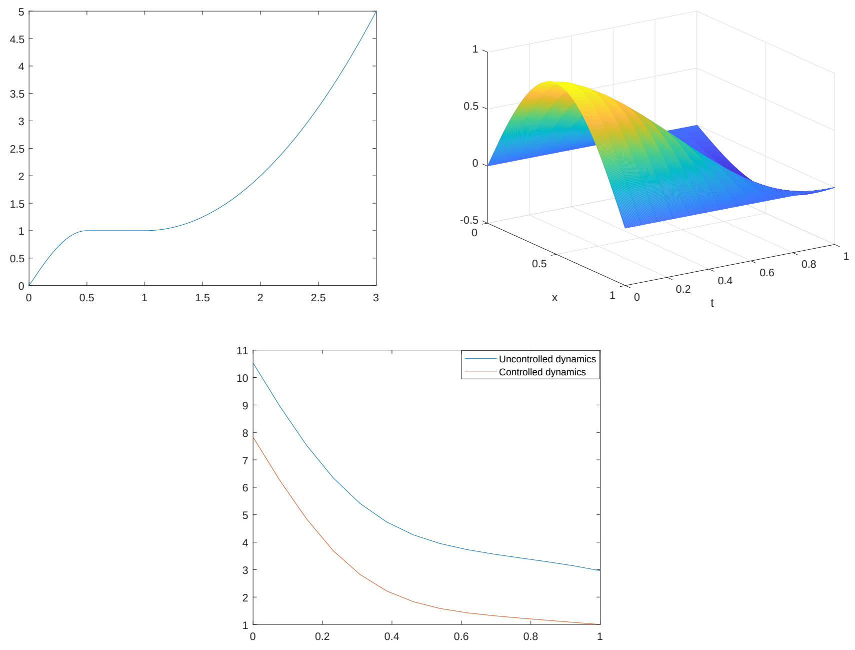


Figure 3.16: Test 5 (Non-quadratic cost functional): Graphics of $\varphi(x)$ (top left), optimal control solution (top right), time comparison of the cost functional of the uncontrolled solution and controlled solution (bottom).

the controlled solution and in the bottom panel a comparison of the evaluation of the cost functional. Again, here we would like to stress the capability of the method to work with high dimensional problem and with non-smooth cost functionals.

3.5.6 Test 6: Bilinear control for advection equation

Finally, we are going to test the method dealing with a linear PDE and we show the effectiveness of high-order methods. We consider the following advection equation:

$$\begin{cases} y_t + c y_x = y u(t) & (x,t) \in \Omega \times [0,T], \\ y(x,t) = 0 & (x,t) \in \partial\Omega \times [0,T], \\ y(x,0) = y_0(x) & x \in \Omega. \end{cases} \tag{3.30}$$

We consider a finite difference approximation for equation (3.30). Here we use $\Delta x = 0.01$, $\Delta t = 0.01$, $\Omega = [0,3]$, $c = 1.5$, $T = 1$ and $y_0(x) = sin(\pi x)\chi_{[0,1]}(x)$. The cost functional we want to minimize is of tracking-type, i.e. we want to stay close to a reference trajectory $\tilde{y}$:

$$J_{y_0,t}(u) = \int_t^T \left(\int_\Omega |y(x,s) - \tilde{y}(x,s)|^2 \, dx + \frac{1}{100}|u(s)|^2 \right) ds + \int_\Omega |y(x,T) - \tilde{y}(x,T)|^2 \, dx. \tag{3.31}$$

To avoid narrow CFL conditions we are going to consider first and second order implicit schemes, applying the pruning criteria (3.2) with $\varepsilon_{\mathcal{T}} = \Delta t^2$ for implicit Euler scheme and $\varepsilon_{\mathcal{T}} = \Delta t^3$ for trapezoidal rule. In the next chapter we are going to show that this choice for the pruning threshold will guarantee the same order of convergence of the underlying scheme.

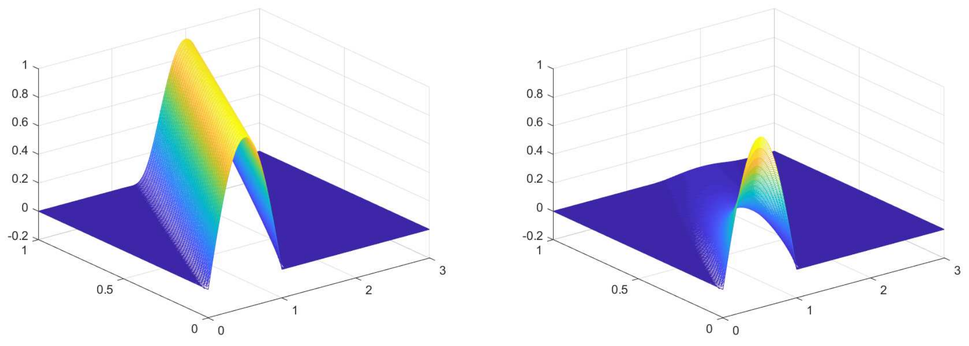


Figure 3.17: Test 6 (Case1): Uncontrolled (left) and controlled solution (right) using trapezoidal method.

Case 1: In the first case we consider the following parameters $U = [-4, 0]$ and $\tilde{y} = 0$ in (3.31). In Figure 3.17 we show the results of the uncontrolled solution and the controlled solution using TSA and trapezoidal rule to approximate the dynamics. We note that the feedback has been built on the tree structure with the same control set as in the computation of the value function as explained in (3.4). As expected that the controlled solution goes to zero faster than the uncontrolled one. Since we do not know the value function in this case, to show the effectiveness of the method we compare the values of the cost functionals in the right panel of Figure 3.18 for each time instance. As expected trapezoidal rule performs better than Euler method. In the bottom plot we show the final configuration at $T = 1$ for both controlled and uncontrolled solution with Euler and trapezoidal scheme. In Table 3.2 we compare the two techniques with 4 controls for Euler scheme and 4 couples of controls with trapezoidal rule. The comparison of the cost functional for the controlled dynamics may be not sufficient, since Euler scheme contains more numerical diffusion.

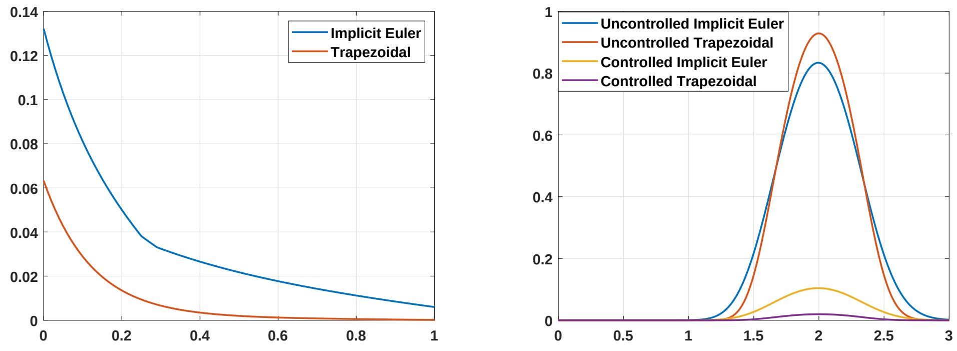


Figure 3.18: Test 6 (Case 1): Comparison of the cost functionals (left) and solutions at final time (right).

Method	Controls	Nodes	CPU	$J_{y_0,0}$
Implicit Euler	4	598204	365s	0.1322
Trapezoidal rule	2×2	348551	111s	0.0632

Table 3.1: Test 6 (Case 1): Comparison of the two methods.

Case 2: In the second case we set $\tilde{y}(x,t) = y_0(x - ct)$ in (3.31) to show better the efficiency of high order schemes. We aim at comparing the solutions which mimic the exact solution of the advection equation. In this case the control $u \in U = [0, 0.5]$ will balance the numerical diffusion of the numerical methods. In the left panel of Figure 3.19 we show the results at final time, while in the right panel we present the computed optimal control for the Euler scheme with 2 discrete controls and $\Delta t = 0.00625$. Clearly, higher order method improves the quality of the solution. In Table 3.2 we compare first and second order method, considering 10 discrete controls for implicit Euler and 4 couples (u_1, u_2) for the trapezoidal rule. As expected, we obtain a better result with lower CPU time in the latter case also considering the lower amount of numerical diffusion of the method.

In this example we have a nice behaviour for the cardinality of the pruned tree. In general we are not able to predict the number of nodes for the pruned tree, but this case shows an exception. The growth of the cardinality for

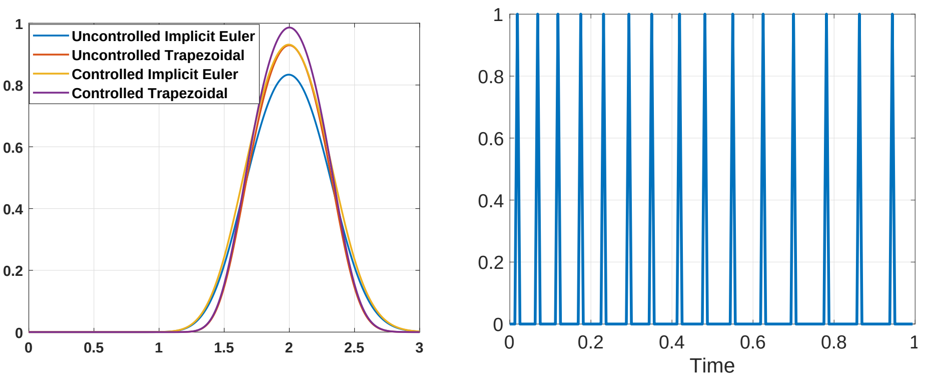


Figure 3.19: Test 6 (Case 2): Comparison of the solutions at final time (left) and and optimal control (right) with 2 discrete controls and $\Delta t = 0.00625$ by Euler scheme.

Method	Controls	Nodes	CPU	$J_{y_0,0}$
Implicit Euler	10	271105	276s	0.0228
Trapezoidal rule	2×2	348551	88s	0.0061

Table 3.2: Test 6 (Case 2): Comparison of the two methods.

the methods is shown in Figure 3.20. For the Euler scheme (left panel), the growth turns out to be linear, leading to a quadratic cardinality of the pruned tree quadratic in the time steps, *e.g* $|\mathcal{T}^P| = O(\overline{N}^2)$. For the trapezoidal rule the growth is quadratic, obtaining a cubic global cost, which is much smaller with respect to the exponential growth of the cardinality of the full tree. We aim at exploring better this behaviour, hopefully giving a precise result on the cardinality of the pruned tree.

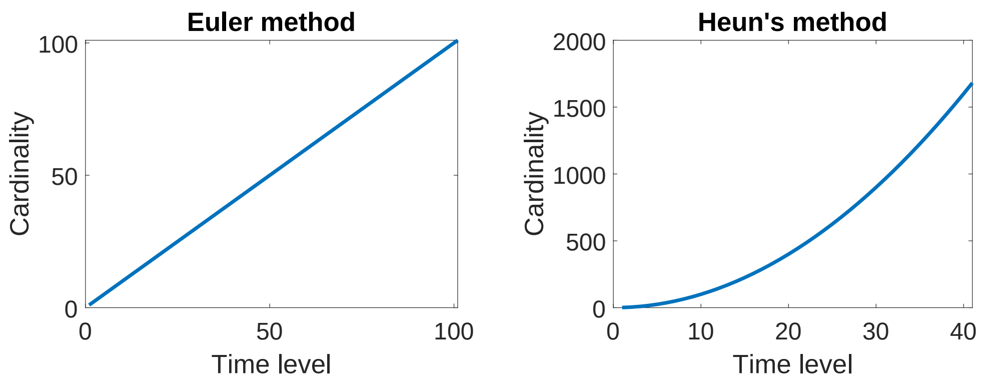


Figure 3.20: Test 6 (Case 2): Number of nodes for each time level for the Euler scheme (left) and for the trapezoidal rule (right)

Chapter 4

Error estimates for TSA

In this chapter we will provide an error analysis for the TSA. This result can be found in the submitted preprint [51]. We will prove the first order convergence for the TSA considering the full tree and then we will extend the result for the pruned case, under suitable assumptions on the pruning threshold. Then, we will extend the assumptions on the pruning threshold to high-order schemes. Finally, some numerical tests will confirm the theoretical findings.

4.1 Error estimates for the TSA without pruning

In this paragraph we first derive the error estimates for the TSA without the introduction of the pruning criteria. In the next paragraph we will extend the result in the pruning case.

We denote $y(s)$ as the exact continuous solution for (2.1) and whenever we want to stress the dependence on the control u, the initial condition x and initial time t we write $y(s; u, x, t)$. We further define $y^n(u)$ as its numerical approximation by an explicit Euler scheme at time t_n. We will consider the piecewise constant extension $\tilde{y}(s; u)$ of the approximation such that

$$\tilde{y}(s, u) := y^{[s/\Delta t]}(u) \quad s \in [t, T], \tag{4.1}$$

where $[\cdot]$ stands for the integer part. Let us now consider the discretized version of the cost functional (2.29):

$$J_{x,s}^{\Delta t}(u) = (t_{n+1} - s)L(x, u, s) + \Delta t \sum_{k=n+1}^{\overline{N}-1} L(y^k, u^k, t_k) e^{-\lambda(t_k - s)} + g(y^{\overline{N}}) e^{-\lambda(t_N - s)}$$

$$= \int_s^T L(\tilde{y}(\sigma, u; x, s), u(\sigma), \left[\frac{\sigma}{\Delta t}\right] \Delta t) e^{-\lambda\left(\left[\frac{\sigma}{\Delta t}\right]\Delta t - s\right)} d\sigma + g\left(\tilde{y}(T, u; x, s)\right) e^{-\lambda(T-s)}$$

for $s \in [t_n, t_{n+1})$ and $u \in \mathcal{U}^\Delta$, where

$$\mathcal{U}^\Delta = \{u : [t, T] \to U, \text{ such that } u(s) = \sum_{k=n}^{\overline{N}-1} \alpha_k \chi_{[t_k, t_{k+1})}(s)\}.$$

We define the discrete value function as

$$V(x,t) := \inf_{u \in \mathcal{U}^{\Delta}} J_{x,t}^{\Delta t}(u)$$

which can be computed by the backward problem

$$\begin{aligned} &V(x,s) = \min_{u \in U}\{e^{-\lambda(t_{n+1}-s)}V(x+(t_{n+1}-s)f(x,u,s),t_{n+1}) + (t_{n+1}-s)\, L(x,u,s)\}, \\ &V(x,T) = g(x), \qquad\qquad x \in \mathbb{R}^d, s \in [t_n, t_{n+1}). \end{aligned} \tag{4.2}$$

Let us recall the hypothesis introduced in Chapter 2. Let us assume that the functions f, L, g are bounded:

$$\begin{gathered} |f(x,u,s)| \leq M_f, \quad |L(x,u,s)| \leq M_L, \quad |g(x)| \leq M_g, \\ \forall\, x \in \mathbb{R}^d,\, u \in U \subset \mathbb{R}^m,\, s \in [t,T], \end{gathered} \tag{4.3}$$

the functions f, L are Lipschitz-continuous with respect to the first variable

$$\begin{gathered} |f(x,u,s) - f(y,u,s)| \leq L_f|x-y|, \quad |L(x,u,s) - L(y,u,s)| \leq L_L|x-y|, \\ \forall\, x,y \in \mathbb{R}^d, u \in U \subset \mathbb{R}^m, s \in [t,T], \end{gathered} \tag{4.4}$$

and finally the cost g is also Lipschitz-continuous:

$$|g(x) - g(y)| \leq L_g|x-y|, \quad \forall x,y \in \mathbb{R}^d. \tag{4.5}$$

The aim of this section is to find an a priori error estimates for the tree algorithm and show the rate of convergence of the approximation V. We show that if the dynamics is discretized by forward Euler method the error is $O(\Delta t)$:

$$\sup_{(x,t)\in\mathbb{R}^d\times[0,T]} |v(x,t) - V(x,t)| \leq \widehat{C}(T)\Delta t \tag{4.6}$$

where Δt is the time discretization of (2.1) and v is the exact solution (5.5). We remark that the estimate guarantees the same order of convergence of the discretization scheme for the dynamical system (2.1). To simplify the proof of the main result (4.6) we have splitted the proof into two parts (see Theorem 4.1.1 and Theorem 4.1.2). We note that this result improves the estimate in [27] under the semiconcavity assumption and it is in line with a similar result for the infinite horizon problem in [16]. To begin with, we show some estimates for the Euler scheme which will be useful to prove the error estimates for TSA. The proposition below follows directly from Grönwall's lemma and its discrete version.

Proposition 4.1.1. *Let us consider the exact solution trajectory $y(s;u,x,t)$ and its approximation $\tilde{y}(s;u,x,t)$ of* (2.1) *for a given control $u \in \mathcal{U}^{\Delta}$. Furthermore, let us assume that assumptions* (4.3) *and* (4.4) *hold true. We then* *obtain the following estimates applying the Euler scheme to* (2.1)*:*

$$|y(s;u,x,t)-\tilde{y}(s;u,x,t)| \leq M_f \Delta t e^{L_f(s-t)}, \tag{4.7}$$

$$\begin{aligned} |\tilde{y}(s;u,x+z,t+\tau)-\tilde{y}(s;u,x,t)| &\leq (|z|+M_f\tau)(1+L_f\Delta t)^{n-k} \\ s\in[t_n,t_{n+1}) \text{ and } t+\tau\in[t_k,t_{k+1}) &\text{ with } \tau\geq 0 \quad s\geq t+\tau. \end{aligned} \tag{4.8}$$

Using Proposition 4.1.1 we are able to prove one side of (4.6) as shown in the following theorem.

Theorem 4.1.1. *Let us assume that conditions* (4.3),(4.4) *and* (4.5) *hold true. Then*

$$\sup_{(x,t)\in\mathbb{R}^d\times[0,T]} (v(t,x)-V(t,x)) \leq C(T)\Delta t, \quad \forall t\in[0,T], \tag{4.9}$$

where $C(T)$ is a constant which does not depend on the time step Δt.

Proof. First, we have

$$v(t,x)-V(t,x) \leq \inf_{u\in\mathcal{U}^\Delta} J_{x,t}(u) - \inf_{u\in\mathcal{U}^\Delta} J^{\Delta t}_{x,t}(u) \leq \sup_{u\in\mathcal{U}^\Delta} |J_{x,t}(u)-J^{\Delta t}_{x,t}(u)|.$$

For a given control $u\in\mathcal{U}^\Delta$, we use the assumptions in Proposition 4.1.1 to obtain the following

$$\begin{aligned} \left|J_{x,t}(u)-J^{\Delta t}_{x,t}(u)\right| &\leq \int_t^T |L(y(x,s,u(s)) - L(\tilde{y}(x,s,u(s))|\,ds + |g(y(T))-g(\tilde{y}(T))| \\ &\leq L_L\int_t^T |y(x,s,u(s))-\tilde{y}(x,s,u(s))|\,ds + L_g|y(T)-\tilde{y}(T)| \\ &\leq L_L M_f\Delta t\int_t^T e^{L_f s}\,ds + L_g M_f \Delta t e^{L_f T} \\ &\leq M_f\Delta t\left(\frac{L_L}{L_f}e^{L_f T} + L_g e^{L_f T}\right). \end{aligned}$$

Then, we obtain the desired estimate (4.9) with $C(T) = M_f e^{L_f T}\left(\frac{L_L}{L_f}+L_g\right)$. □

To prove the remaining side of (4.6) we need to assume the semiconcavity of the functions g, L and a stronger assumption on f. The proof of Theorem 4.1.2 is based on some technical lemmas that are presented below.

Proposition 4.1.2. *Let us consider the assumptions of Proposition 4.1.1 and consider the dynamics $f(x,u,t)$ as a Lipschitz-continuous function in time and space uniformly in u with the following property*

$$\begin{aligned} |f(x+z,u,t+\tau)-2f(x,u,t)+f(x-z,u,t-\tau)| &\leq C_f(|z|^2+\tau^2), \\ \forall u\in U,\ \forall x,z\in\mathbb{R}^d,\ &\forall t,\tau>0, \end{aligned} \tag{4.10}$$

then

$$|\tilde{y}(s;u,x+z,t+\tau)-2\tilde{y}(s;u,x,t)+\tilde{y}(s;u,x-z,t-\tau)|\le \widetilde{C}(T)(|z|^2+\tau^2),$$
$$\forall s\ge t+\tau, \forall u\in U,\ \forall x,z\in\mathbb{R}^d,\ \forall t,\tau>0, \tag{4.11}$$

where $\widetilde{C}(T)$ *is a constant that depends on* T *but does not depend on the time step* Δt.

Proof. Let us suppose that $t+\tau\in[t_k,t_{k+1})$ for some $k>0$, $t\in[t_0,t_1)$ and $t-\tau\in[t_{-k-1},t_{-k})$. Let us consider $s\in[t_{n+1},t_{n+2})$, to ease the notation we will denote

$$\tilde{y}(s,u,x+z,t+\tau):=y_+^{n+1}, \qquad \tilde{y}(s,u,x,t):=y^{n+1},$$
$$\tilde{y}(s,u,x-z,t-\tau):=y_-^{n+1}, \qquad f(y,u,t_n):=f^n(y),$$

and we will drop the dependence on the control u since it is fixed for all the terms considered above. Applying only one step of the forward Euler scheme with $n\ge k$ we get

$$y_+^{n+1}-2y^{n+1}+y_-^{n+1}=y_+^n-2y^n+y_-^n+\Delta t\left(f^n(y_+^n)-2f^n(y^n)+f^n(y_-^n)\right).$$

Thus, from assumption (4.10) we obtain the following

$$\begin{aligned}&|f^n(y_+^n)-2f^n(y^n)+f^n(y_-^n)|=\\&|f^n(y_+^n)-2f^n(y^n)+(f^n(y^n-(y_+^n-y^n))-f^n(y^n-(y_+^n-y^n)))+f^n(y_-^n)|\le\\&|f^n(y^n+(y_+^n-y^n))-2f^n(y^n)+f^n(y^n-(y_+^n-y^n))|+\\&|f^n(y_-^n)-f^n(y^n-(y_+^n-y^n))|\le C_f|y_+^n-y^n|^2+L_f\left|y_+^n-2y^n+y_-^n\right|.\end{aligned}$$

Then, applying (4.8) we obtain

$$|y_+^{n+1}-2y^{n+1}+y_-^{n+1}|\le \Delta t C_1 C_2^{2(n-k)}+C_2|y_+^n-2y^n+y_-^n|, \tag{4.12}$$

with $C_1=C_f(|z|+M_f\tau)^2$ and $C_2=1+L_f\Delta t$. Then, iterating (4.12) we obtain

$$|y_+^{n+1}-2y^{n+1}+y_-^{n+1}|\le \Delta t C_1 C_2^{2(n-k)}\sum_{j=0}^{n-k}C_2^{-j}+C_2^{n-k+1}|x+z-2y^k+y_-^k|. \tag{4.13}$$

Writing the full discrete dynamics for y^k and y_-^k, the right hand side in (4.13) becomes

$$\begin{aligned}&\Delta t C_1 C_2^{2(n-k)}\frac{1-C_2^{-(n-k+1)}}{1-C_2^{-1}}+C_2^{n-k+1}\Delta t\left|-2\sum_{j=0}^{k-1}f^j(y^j)+\sum_{j=-k}^{k-1}f^j(y_-^j)\right|\le\\&\frac{C_1C_2^{2(n-k)+1}}{L_f}+C_2^{n-k+1}\Delta t\left|\sum_{j=0}^{k-1}\left(f^j(y_-^j)-f^j(y^j)+f^{j-k}(y_-^{j-k})-f^j(y^j)\right)\right|.\end{aligned} \tag{4.14}$$

Now we want to estimate last term in (4.14). Since the first term $f^j(y_-^j) - f^j(y^j)$ of the sum can be obtained as a particular case of the second one, with $k = 0$, let us now focus on the last term

$$\left|\sum_{j=0}^{k-1} \left(f^{j-k}(y_-^{j-k}) - f^j(y^j)\right)\right| \leq L_f \sum_{j=0}^{k-1} \left(\left|y_-^{j-k} - y^j\right| + \tau\right). \tag{4.15}$$

Using (4.8), we can write

$$\left|y_-^{j-k} - y^j\right| \leq \left|y_-^{j-k} - y_-^j\right| + \left|y_-^j - y^j\right| \leq \tau M_f + (|z| + M_f\tau)C_2^j.$$

Finally we get

$$|y_+^{n+1} - 2y^{n+1} + y_-^{n+1}| \leq$$

$$\frac{C_1 C_2^{2(n-k)+1}}{L_f} + 2C_2^{n-k+1} L_f \left(\tau^2 M_f + C_2^k \tau(|z| + M_f\tau)\right).$$

Noting that $C_2^n = (1 + L_f\Delta t)^n \leq e^{t_n L_f}$, we obtain the desired result with the constant $\widetilde{C}(T)$ equal to

$$\widetilde{C}(T) = 2e^{2T}\left(\frac{C_f(\max\{1, M_f\})^2}{L_f} + L_f(2M_f + 1)\right). \tag{4.16}$$

□

Let us recall some properties for the numerical value function which will be useful later since the reverse inequality in (4.9) needs the assumption of semiconcavity for the numerical approximation V. We refer to [15] for a detailed discussion of the importance of semiconcavity in control problems.

Proposition 4.1.3. *Let us suppose that the functions L and g are both Lipschitz-continuous and semiconcave. Furthermore, let us consider the function $f(x, u, t)$ as a Lipschitz-continuous function in time and space uniformly in u such that it verifies* (4.10)*. Then the numerical solution V is semiconcave:*

$$V(x+z, t+\tau) - 2V(x, t) + V(x-z, t-\tau) \leq C_V(|z|^2 + \tau^2) \quad \forall x, z \in \mathbb{R}^n, t, \tau \geq 0. \tag{4.17}$$

Proof. Given $x, z \in \mathbb{R}^n$ and $t, \tau \in [0, T]$ such that $t + \tau \in [t_k, t_{k+1})$, $t \in [t_0, t_1)$ and $t - \tau \in [t_{-k-1}, t_{-k})$, we need to prove (4.17). By the definition of value function, we can write

$$\begin{aligned} &V(x + z, t + \tau) + V(x - z, t - \tau) - 2V(x, t) \leq \sup_{u \in U} \{(t_{k+1} - t - \tau)L(x + z, t + \tau, u) + \\ &(t_{-k} - t + \tau)L(x - z, t - \tau, u) - 2(t_1 - t)L(x, t, u)\} \\ &+ \sup_{u \in \mathcal{U}^\Delta} \left(J_T^{\Delta t}(x + z, t_{k+1}, u) + J_T^{\Delta t}(x - z, t_{-k}, u) - 2J_T^{\Delta t}(x, t_1, u)\right). \end{aligned} \tag{4.18}$$

We can estimate the first term on the right hand side as follows

$$(t_{k+1} - t - \tau)L(x+z, t+\tau, u) + (t_{-k} - t + \tau)L(x-z, t-\tau, u) - 2(t_1 - t)L(x, t, u))$$
$$\leq \Delta t \max\{L(x+z, t+\tau, u) + L(x-z, t-\tau, u) - 2L(x, t, u), 0\}. \tag{4.19}$$

Without loss of generality, we will consider $\lambda = 0$. Given $u \in \mathcal{U}^\Delta$ and denoted by $L(y, u, t_n) = L^n(y)$, we have that the remaining right hand side is equal to

$$\Delta t \left(\sum_{n=k+1}^{N-1} \left(L^n(y_+^n) + L^n(y_-^n) - 2L^n(y^n) \right) + \sum_{n=1}^{k} \left(L^n(y_-^n) - 2L^n(y^n) \right) \right) +$$
$$\Delta t \left(\sum_{n=-k}^{0} L^n(y_-^n) \right) + g(y_+^N) + g(y_-^N) - 2g(y^N). \tag{4.20}$$

As already done in the proof of Proposition 4.1.2, exploiting the properties of L, i.e. Lipschitz-continuity and semiconcavity with constant $C_L > 0$, for the first summation in (4.20) we have:

$$L^n(y_+^n) + L^n(y_-^n) - 2L^n(y^n) \leq C_L |y_+^n - y^n|^2 + L_L |y_+^n - 2y^n + y_-^n|. \tag{4.21}$$

Using (4.8), we obtain the following bound for the first term

$$\begin{aligned} \Delta t C_L \sum_{n=k+1}^{N-1} |y_+^n - y^n|^2 &\leq \Delta t C_L (|z| + M_f \tau)^2 \sum_{n=k+1}^{N-1} (1 + L_f \Delta t)^{2(N-k)} \leq \\ \leq \frac{C_L}{L_f} (1 + L_f \Delta t)^{2N} (|z| + M_f \tau)^2 &\leq 2 \max\{M_f^2, 1\} \frac{C_L}{L_f} e^{2L_f T} (|z|^2 + \tau^2). \end{aligned} \tag{4.22}$$

Using (4.11), we obtain directly

$$\Delta t L_L \sum_{n=k+1}^{N-1} |y^n(x+z, t+\tau) - 2y^n(x, t) + y^n(x-z, t-\tau)| \leq T L_L \widetilde{C}(T)(|z|^2 + \tau^2).$$

Finally we rewrite the second and third summation in (4.20) in the following way

$$\Delta t \sum_{n=1}^{k} [(L^n(y_-^n) - L^n(y^n)) + (L^{n-k-1}(y_-^n) - L^n(y^n))]$$

and with the same procedure used in the proof of Proposition 4.1.2 and applying (4.21) with g, we obtain the desired estimate. □

Next, we introduce a further characterization of V which will turn out to be useful to prove Theorem 4.1.2.

Proposition 4.1.4. *Assume that assumptions* (4.3), (4.4), (4.5) *hold true. Then the solution* V *of* (3.1) *is bounded (and uniformly continuous). Furthermore, the following estimate holds*

$$|V(y_0,s)-V(x_0,T)| \leq C\left(|y_0-x_0|+(T-t_n)+\Delta t\right), \quad s\in[t_n,t_{n+1}), \forall x_0,y_0\in\mathbb{R}^n. \tag{4.23}$$

The proof of this statement can be found in [27]. Finally, before proving Theorem 4.1.2 we introduce the following lemma (proved in [16, Lemma 4.2, p. 170]).

Lemma 4.1.1. *Let* $\xi:\mathbb{R}^n\times[0,T]\to\mathbb{R}$ *satisfy*

$$\xi(y+z,t+\tau)-2\xi(y,t)+\xi(y-z,t-\tau)\leq C_\xi\left(|z|^2+|\tau|^2\right),$$

$\forall y,z\in\mathbb{R}^n$, $\forall t,\tau\in[0,T]$ *such that* $t+\tau,t,t-\tau\in[0,T]$ *and*

$$\xi(0,0)=0, \quad \limsup_{(y,t)\to(0,0)}\frac{\xi(y,t)}{|y|+|t|}\leq 0.$$

Then

$$\xi(y,t)\leq\frac{C_\xi}{6}(|y|^2+|t|^2) \quad \forall y\in\mathbb{R}^n, t\in[0,T].$$

We are now able to prove our main result.

Theorem 4.1.2. *Let the assumptions* (4.3),(4.4),(4.5) *hold true. Moreover, let us assume that the functions* L *and* g *are semiconcave and that the function* $f(x,u,t)$ *is Lipschitz continuous in space and time uniformly in* u *and it satisfies* (4.10). *Then*

$$\sup_{(x,t)\in\mathbb{R}^d\times[0,T]}(V(t,x)-v(t,x))\leq\overline{C}(T)\Delta t, \quad \forall t\in[0,T]. \tag{4.24}$$

Proof. The first part of the proof follows closely from [27]. We introduce the auxiliary function

$$\phi(y,t,x,s)=V(y,t)-v(x,s)+\beta_\epsilon(x-y)+\eta_\alpha(t-s),$$

where $\beta_\epsilon(x)=-\frac{|x|^2}{\epsilon^2}$ and $\eta_\alpha(s)=-\frac{s^2}{\alpha^2}$.

Since v and V are bounded, then for any $\delta>0$, there exist $(y_1,\tau_1),(x_1,s_1)$ such that

$$\phi(y_1,\tau_1,x_1,s_1)>\sup\phi-\delta.$$

Choosing $\theta(y,x)\in C_0^\infty(\mathbb{R}^d\times\mathbb{R}^d)$, with $\theta(y_1,x_1)=1$ and $0\leq\theta\leq1$, $|D\theta|\leq1$, such that for any $\delta\in(0,1)$,

$$\zeta(y,t,x,s)=\phi(y,t,x,s)+\delta\theta(y,x)$$

has a maximum point (y_0, τ_0, x_0, s_0), with $y_0, x_0 \in \operatorname{supp}\theta$ and $\tau_0, s_0 \in [0,T]$. Therefore, if we set

$$\Phi(x,s) = V(y_0,\tau_0) + \beta_\epsilon(y_0 - x) + \eta_\alpha(\tau_0 - s) + \delta\theta(y_0,x),$$

we can observe that (x_0, s_0) is a local min for $v(x,s) - \Phi(x,s)$. By definition of ζ, we have that

$$\begin{aligned} &V(y_0,\tau_0) - v(x_0,s_0) + \beta_\epsilon(y_0 - x_0) + \eta_\alpha(\tau_0 - s_0) + \delta\theta(y_0,x_0) \geq \\ &\geq V(y,t) - v(x,s) + \beta_\epsilon(y - x) + \eta_\alpha(t - s) + \delta\theta(y,x). \end{aligned} \tag{4.25}$$

From (4.25) with $x = y = y_0$, $s = s_0$ and $t = \tau_0$, we get

$$|y_0 - x_0| \leq \epsilon^2(L_v + \delta), \tag{4.26}$$

and similarly, with $x = x_0$, $y = y_0$ and $s = t = \tau_0$:

$$|s_0 - \tau_0| \leq \alpha^2 L_v, \tag{4.27}$$

where L_v is the Lipschitz constant of v with respect to time and space. Using (4.25), (4.26) and (4.27), we obtain

$$V(x,s) - v(x,s) \leq V(y_0,\tau_0) - v(x_0,s_0) + (L_v + \delta)\epsilon^2 + \alpha^2 L_v + 2\delta. \tag{4.28}$$

Let us now consider three cases as suggested in [27]. We recall that in this theorem we improve their approximation by means of the semiconcavity which turns out to be essential in the third case of the proof. However, in the first two cases we can directly obtain first order convergence. Without this property we can only prove an order of convergence of $\frac{1}{2}$.

First case ($\tau_0 = T$) In this case $V(y_0,T) = g(y_0) = v(y_0,T)$. Thus, using the Lipschitz-continuity of g we obtain the desired result, setting $\alpha = \epsilon = \sqrt{\Delta t}$.

Second case ($\tau_0 \neq T$, $s_0 = T$) In this case $v(x_0,T) = g(x_0) = V(x_0,T)$. Supposing $\tau_0 \in [t_n, t_{n+1})$ and using the estimate (4.23) in (4.28), we obtain

$$V(x,s) - v(x,s) \leq C\left(|y_0 - x_0| + (T - t_n) + \Delta t\right) + (L_v + \delta)\epsilon^2 + \alpha^2 L_v + 2\delta.$$

Since $\tau_0 - t_n \leq \Delta t$, using (4.27) we can write that

$$T - t_n \leq L_v\alpha^2 + \Delta t,$$

and using (4.26), finally we get

$$V(x,s) - v(x,s) \leq C_3\epsilon^2 + C_4\alpha^2 + C_5\Delta t + 2\delta.$$

If we set $\alpha = \epsilon = \sqrt{\Delta t}$, we get the result, since δ is arbitrary.

Third case ($\tau_0, s_0 \neq T$) We know that v is a viscosity solution, this means that there exists a control $u^* \in U$ such that

$$-\partial_s \Phi(x_0, s_0) + \lambda v(x_0, s_0) - f(x_0, s_0, u^*) \cdot \nabla_x \Phi(x_0, s_0) - L(x_0, s_0, u^*) \geq 0.$$

Thus, we obtain

$$\begin{aligned} &\nabla \eta_\alpha(\tau_0 - s_0) + \lambda v(x_0, s_0) + \\ &\quad + f(x_0, s_0, u^*) \cdot (\nabla \beta_\epsilon(x_0 - y_0) - \delta \nabla_x \theta(y_0, x_0)) - L(x_0, s_0, u^*) \geq 0. \end{aligned} \tag{4.29}$$

By the definition of V (5.30), assuming $\tau_0 \in [t_n, t_{n+1})$ we have

$$\begin{aligned} &V(y_0, \tau_0) - (t_{n+1} - \tau_0)\, L(y_0, \tau_0, u^*) + \\ &\quad - e^{-\lambda(t_{n+1} - \tau_0)} V(y_0 + (t_{n+1} - \tau_0)\, f(y_0, \tau_0, u^*), t_{n+1}) \leq 0. \end{aligned} \tag{4.30}$$

Let us introduce

$$\xi(y, t) = V(y_0 + y, \tau_0 + t) - V(y_0, \tau_0) + (\nabla \beta_\epsilon(x_0 - y_0) + \delta \nabla_y \theta(y_0, x_0)) \cdot y + t \nabla \eta_\alpha(\tau_0 - s_0) \tag{4.31}$$

and follows that

$$\begin{aligned} &\xi(y + z, t + \tau) - 2\xi(y, t) + \xi(y - z, t - \tau) = \\ &\quad = V(y_0 + y + z, \tau_0 + t + \tau) - 2V(y_0 + y, \tau_0 + t) + V(y_0 + y - z, \tau_0 + t - \tau). \end{aligned}$$

By Proposition 4.1.3, we know that the function V is semiconcave, from which it follows the semiconcavity of ξ with $\xi(0, 0) = 0$. Let us now check the last hypothesis of Lemma 4.1.1. Since (y_0, x_0, τ_0, s_0) is a maximum point for ζ, we obtain

$$\begin{aligned} &V(y_0 + y, \tau_0 + t) - V(y_0, \tau_0) \leq \\ &\quad \leq \beta_\epsilon(y_0 - x_0) - \beta_\epsilon(y_0 + y - x_0) + \eta_\alpha(\tau_0 - s_0) - \eta_\alpha(\tau_0 + t - s_0) + \\ &\qquad\qquad + \delta[\theta(y_0, x_0) - \theta(y_0 + y, x_0)], \end{aligned}$$

$$\begin{aligned} &\xi(y, t) \leq \beta_\epsilon(y_0 - x_0) - \beta_\epsilon(y_0 + y - x_0) + \nabla \beta_\epsilon(x_0 - y_0) \cdot y + \eta_\alpha(\tau_0 - s_0) \\ &- \eta_\alpha(\tau_0 + t - s_0) + t \nabla \eta_\alpha(\tau_0 - s_0) + \delta(\theta(y_0, x_0) - \theta(y_0 + y, x_0) + \nabla_y \theta(y_0, x_0) \cdot y). \end{aligned}$$

We note that

$$\limsup_{(t,y) \to (0,0)} \frac{\xi(y, t)}{|y| + |t|} \leq 0.$$

Applying Lemma 4.1.1 with $y = (t_{n+1} - \tau_0)\, f(y_0, \tau_0, u^*)$ and $t = t_{n+1} - \tau_0$, we obtain

$$\begin{aligned} &V(y_0 + (t_{n+1} - \tau_0)\, f(y_0, \tau_0, u^*), t_{n+1}) \leq \\ &\qquad V(y_0, \tau_0) - (t_{n+1} - \tau_0)(\nabla \beta_\epsilon(x_0 - y_0) + \delta \nabla_y \theta(y_0, x_0)) \cdot \end{aligned}$$

$$f(y_0,\tau_0,u^*)-(t_{n+1}-\tau_0)\nabla\eta_\alpha(\tau_0-s_0)+C_\xi(t_{n+1}-\tau_0)^2(1+|f(y_0,\tau_0,u^*)|^2). \quad (4.32)$$

Inserting (4.32) in (4.30) and dividing by $t_{n+1}-\tau_0$ we obtain

$$\frac{1-e^{-\lambda(t_{n+1}-\tau_0)}}{t_{n+1}-\tau_0}V(y_0,\tau_0) \le L(y_0,\tau_0,u^*) - e^{-\lambda(t_{n+1}-\tau_0)}(\nabla\beta_\epsilon(x_0-y_0)+$$
$$\delta\nabla_y\theta(y_0,x_0))\cdot f(y_0,\tau_0,u^*) + \nabla\eta_\alpha(\tau_0-s_0) - C(t_{n+1}-\tau_0)(1+|f(y_0,\tau_0,u^*)|^2)).$$

Finally, subtracting (4.29), we obtain

$$\begin{aligned}
&\frac{1-e^{-\lambda(t_{n+1}-\tau_0)}}{t_{n+1}-\tau_0}V(y_0,\tau_0) - \lambda v(x_0,s_0) \le L(y_0,\tau_0,u^*) - L(x_0,s_0,u^*)+\\
&\nabla\beta_\epsilon(y_0-x_0)\cdot(-e^{-\lambda(t_{n+1}-\tau_0)}f(y_0,\tau_0,u^*)+f(x_0,s_0,u^*))+\\
&\nabla n_\alpha(\tau_0-s_0)(1-e^{-\lambda(t_{n+1}-\tau_0)}) + \delta\left(-e^{-\lambda(t_{n+1}-\tau_0)}\nabla_y\theta(y_0,x_0)\cdot f(y_0,\tau_0,u^*)\right)\\
&+\delta\left(-\nabla_x\theta(y_0,x_0)\cdot f(x_0,s_0,u^*)\right) + C(t_{n+1}-\tau_0)(1+|f(y_0,\tau_0,u^*)|^2)\\
&\le L_L(|y_0-x_0|+|\tau_0-s_0|) + 2(L_v+\delta)L_f(|y_0-x_0|+|\tau_0-s_0|) + 2L_v\Delta t+\\
&M\delta + C\Delta t(1+M_f^2) \le L(\epsilon^2+\alpha^2) + C\Delta t + M\delta.
\end{aligned}$$

Since δ is arbitrary, choosing $\alpha=\epsilon=\sqrt{\Delta t}$, we obtain the thesis. □

Error estimate in the control space

The previous results do not take into account the error in the minimization procedure computed by comparison in the discrete set $U^{\Delta u}$. For this reason let us define the value function computed with discrete controls as

$$\overline{V}(x,t) = \inf_{u\in\overline{\mathcal{U}}^\Delta} J_{x,t}^{\Delta t}(u), \quad (4.33)$$

where

$$\overline{\mathcal{U}}^\Delta = \{u:[t,T]\to U^{\Delta u},\ \text{such that } u(s)=\sum_{k=n}^{\overline{N}-1}\alpha_k\chi_{[t_k,t_{k+1})}(s)\}.$$

We can obtain an error estimate for the error of the minimization by comparison adding the hypothesis of Lipschitz-continuity for f and L with respect to the state and the control uniformly in time, *i.e.*

$$\begin{gathered}
|f(x,u,s)-f(y,\tilde{u},s)| \le L_f(|x-y|+|u-\tilde{u}|),\\
|L(x,u,s)-L(y,\tilde{u},s)| \le L_L(|x-y|+|u-\tilde{u}|),\\
\forall\, x,y\in\mathbb{R}^d, u,\tilde{u}\in U\subset\mathbb{R}^m, s\in[t,T].
\end{gathered} \quad (4.34)$$

Proposition 4.1.5. *Under the assumptions* (4.34) *and* (4.5)*, then*

$$\sup_{(x,t)\in\mathbb{R}^d\times[0,T]} |V(x,t)-\overline{V}(x,t)| \leq C(T,m)\Delta u, \tag{4.35}$$

where m is the dimension of the control set U.

Proof. First, we can observe that $\overline{\mathcal{U}}^{\Delta} \subset \mathcal{U}^{\Delta}$, then $V(x,t) \leq \overline{V}(x,t)$. Then, supposing $t\in[t_n,t_{n+1})$ and imposing $\lambda=0$ for simplicity , we obtain

$$\overline{V}(x,t)-V(x,t) \leq V^{n+1}(x+(t_{n+1}-t)f(x,\overline{u}^n,t)+(t_{n+1}-t)L(x,\overline{u}^n,t)$$
$$-V^{n+1}(x+(t_{n+1}-t)f(x,u^n_*,t)-(t_{n+1}-t)L(x,u^n_*,t),$$

where

$$u^n_* = \arg\min_{u\in U}\{V^{n+1}(x+(t_{n+1}-t)f(x,u,t)+(t_{n+1}-t)L(x,u,t)\}$$

and $\overline{u}^n$ is chosen such that $|\overline{u}^n-u^n_*|\leq \frac{\sqrt{m}}{2}\Delta u$. This choice is possible since $U^{\Delta u}$ is a discretization of U with step-size Δu in all directions. Proceeding in the same fashion of Proposition 2.3.1, we obtain

$$\overline{V}(x,t)-V(x,t) \leq L_L\sum_{k=n}^{N}\alpha_k\left(|\overline{x}^k-x^k_*|+|\overline{u}^k-u^k_*|\right)+L_g|\overline{x}^N-x^N_*|, \tag{4.36}$$

where

$$\overline{x}^k := x+\sum_{j=n}^{k-1}\alpha_j f(\overline{x}^j,\overline{u}^j,\bar{t}_j), \quad x^k_* = x+\sum_{j=n}^{k-1}\alpha_j f(x^j_*,u^j_*,\bar{t}_j),$$

$$\alpha_j = \begin{cases} t_{n+1}-t & j=n \\ \Delta t & k\geq n+1\end{cases}, \quad \bar{t}_j = \begin{cases} t & j=n \\ t_k & j\geq n+1\end{cases},$$

$$u^j_* = \arg\min_{u\in U}\{V^{j+1}(x^j_*+\alpha_j f(x^j_*,u,\bar{t}_j))+\alpha_j L(x^j_*,u,\bar{t}_j)\}, \quad j\geq n,$$

and $\overline{u}^j$ chosen such that

$$|\overline{u}^j-u^j_*|\leq\frac{\sqrt{m}}{2}\Delta u, \quad j\geq n.$$

By Grönwall's lemma we obtain

$$|\overline{x}^k-x^k_*|\leq e^{(t_k-t)L_f}(t_k-t)L_f\frac{\sqrt{m}}{2}\Delta u, \quad j\geq n, \tag{4.37}$$

and finally coupling (4.36) and (4.37)

$$\sup_{(x,t)\in\mathbb{R}^d\times[0,T]}|\overline{V}(x,t)-V(x,t)|\leq\Delta u\frac{\sqrt{m}}{2}\left(e^{TL_f}T(L_L+L_g)+TL_L\right). \tag{4.38}$$

□

4.2 Error estimate for the TSA with pruning

In the previous section we presented an error estimate for the TSA where a first order of convergence is achieved. However, as shown numerically in [3], one can obtain the same order of convergence in the case of the pruned tree if the pruning tolerance $\varepsilon_{\mathcal{T}}$ in (3.2) is chosen properly. In this section, we extend the previous theoretical results to the pruning case. Thus, let us define the *pruned trajectory:*

$$\eta_{i_n}^{n+1} = \eta_{i_{n-1}}^n + \Delta t f(\eta_{i_{n-1}}^n, u_{j_n}, t_n) + \mathcal{E}_{\varepsilon_{\mathcal{T}}}(\eta_{i_{n-1}}^n + \Delta t f(\eta_{i_{n-1}}^n, u_{j_n}, t_n), \{\eta_i^{n+1}\}_i), \tag{4.39}$$

where the indices i_n and j_n consider the pruning strategy with

$$\mathcal{E}_{\varepsilon_{\mathcal{T}}}(x, \{x_n\}) = \begin{cases} x_k - x & \text{if } k \in \arg\min_n |x - x_n| \text{ and } |x - x_k| \leq \varepsilon_{\mathcal{T}}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.40}$$

The function $\mathcal{E}_{\varepsilon_{\mathcal{T}}}(x, \{x_n\})$ can be interpreted as a perturbation of the numerical scheme and $|\mathcal{E}_{\varepsilon_{\mathcal{T}}}(x, \{x_n\})| \leq \varepsilon_{\mathcal{T}}$. As already done in (4.1), we consider the piecewise constant extension $\tilde{\eta}(s; u)$ of the approximation such that

$$\tilde{\eta}(s, u) := \eta^{[s/\Delta t]}(u) \quad s \in [t, T]. \tag{4.41}$$

First step is to prove that the tolerance must be chosen properly to guarantee a first order convergence of the scheme. The following result is obtained easily through Grönwall's lemma.

Proposition 4.2.1. *Given the approximation $\tilde{y}(s; u, x, t)$ of equation* (2.1) *and its perturbation $\tilde{\eta}(s; u, x, t)$ expressed in* (4.2)*, then*

$$|\tilde{y}(s; u, x, t) - \tilde{\eta}(s; u, x, t)| \leq \varepsilon_{\mathcal{T}} \frac{s - t}{\Delta t} e^{L_f (s-t)}, \quad \forall s \in [t, T]. \tag{4.42}$$

Finally, to guarantee first order convergence, the tolerance must be chosen such that

$$\varepsilon_{\mathcal{T}} \leq C \Delta t^2. \tag{4.43}$$

Then we can define the *pruned* discrete cost functional as

$$J_{x,s}^{\Delta t, P}(u) = (t_{n+1} - s) L(x, u, s) + \Delta t \sum_{k=n+1}^{N-1} L(\eta^k, u, t_k) e^{-\lambda(t_k - s)} + g(\eta^{\overline{N}}) e^{-\lambda(t_N - s)}, \tag{4.44}$$

for $s \in [t_n, t_{n+1})$ and define the *pruned* discrete value function as

$$V^P(x, t) := \inf_{u \in \mathcal{U}^\Delta} J_{x,t}^{\Delta t, P}(u)$$

which now satisfies the following equation

$$\begin{aligned} &V^P(x,s) = \min_{u\in U} \left\{ e^{-\lambda(t_{n+1}-s)} V^{P,n+1}\left(\eta_u^{n+1}(x)\right) + (t_{n+1}-s)L(x,u,s) \right\}, \\ &V(x,T) = g(x), \qquad x \in \mathbb{R}^d, s \in [t_n, t_{n+1}) \end{aligned} \tag{4.45}$$

where

$$\eta_u^{n+1}(x) = x + (t_{n+1}-s)f(x,u,s) + \mathcal{E}_{\varepsilon_\mathcal{T}}(x + (t_{n+1}-s)f(x,u,s), \{\eta_i^{n+1}\}_i).$$

Then, we can prove the following result.

Proposition 4.2.2. *Under the condition* (4.43)*, we have*

$$|V(x,t) - V^P(x,t)| \leq C^*(T)\Delta t. \tag{4.46}$$

Proof. As done in the proof of Theorem 4.1.1, we can write

$$|V(x,t) - V^P(x,t)| \leq \sup_{u\in\mathcal{U}^\Delta} \left| J_{x,t}^{\Delta t}(u) - J_{x,t}^{\Delta t,P}(u) \right|.$$

Then using (4.42) we obtain the desired result as follows:

$$\begin{aligned} \left| J_{x,t}^{\Delta t}(u) - J_{x,t}^{\Delta t,P}(u) \right| &\leq \\ &\leq L_L \int_t^T |\tilde{y}(x,s,u(s)) - \tilde{\eta}(x,s,u(s))|\, ds + L_g |\tilde{y}(T) - \tilde{\eta}(T)| \\ &\leq L_L C \Delta t \int_t^T s\, e^{L_f(s-t)}\, ds + L_g C T \Delta t e^{L_f T} \\ &\leq T C \Delta t e^{L_f T} \left(T L_L + L_g \right). \end{aligned}$$

□

Finally by triangular inequality and using estimate (4.6) and (4.46), we obtain the desired result:

$$|v(x,t) - V^P(x,t)| \leq \left(C^*(T) + \widehat{C}(T) \right) \Delta t. \tag{4.47}$$

whenever condition (4.43) holds true.

Pruning threshold for high-order TSA In the last paragraph of Chapter 3 we considered the extension of the TSA for high order schemes. Given the high-order approximation (3.6), we can define the pruned trajectory in the following way

$$\eta_j^{n+1} = \eta^n + \Delta t\, \Phi(\eta^n, \mathbf{U}_n, t_n, \Delta t) + \mathcal{E}_{\varepsilon_\mathcal{T}}(\eta^n + \Delta t\, \Phi(\eta^n, \mathbf{U}_n, t_n, \Delta t), \{\eta_i^{n+1}\}_i).$$

Under the assumption of Lipschitz-continuity for Φ, it is possible to prove an estimate similar to (4.42).

Proposition 4.2.3. *Given a one-step approximation $\{y^n\}_n$ and its perturbation $\{\eta^n\}_n$, if the function Φ is Lipschitz-continuous of constant L_Φ, then*

$$|y^n - \eta^n| \leq n\, \varepsilon_{\mathcal{T}} \frac{s-t}{\Delta t} e^{L_\Phi(t_n - t)}.$$

Furthermore, to guarantee a convergence of order p, the tolerance $\varepsilon_{\mathcal{T}}$ must satisfy the condition

$$\varepsilon_{\mathcal{T}} \leq C \Delta t^{p+1}.$$

By this last result, we understand that the tolerance must scale an order more than the convergence order of the underlying scheme. We will show in the numerical tests that the use of higher order schemes leads to more accurate results in less computational time.

4.3 Numerical Tests

In this section we are going to show numerically the theoretical results proven in the previous sections. We will present two test cases. In the first example, provided the analytical value function, we show the order of convergence of the method for Euler and Heun's schemes. Furthermore, we emphasize the importance of the assumptions provided in the previous pages with respect to the semiconcavity of the value function. The second example deals with a high-dimensional problem: the control of the heat equation. In this setting we approximate the value function by a very accurate simulation of the Riccati equation and, then, compute the order of convergence of our method.

4.3.1 Test 1: Comparison with an exact solution

In this test we compute the order and the error of the TSA in an example where the exact value function is known analytically. We consider again the following dynamics in (2.1)

$$f(x,u) = \begin{pmatrix} u \\ x_1^2 \end{pmatrix}, \; u \in U \equiv [-1,1], \tag{4.48}$$

where $x = (x_1, x_2) \in \mathbb{R}^2$ and $T = 1$. We recall that the solution of the HJB in this case is

$$v(x,t) = -x_2 - x_1^2(T-t) - \frac{1}{3}(T-t)^3 - |x_1|(T-t)^2. \tag{4.49}$$

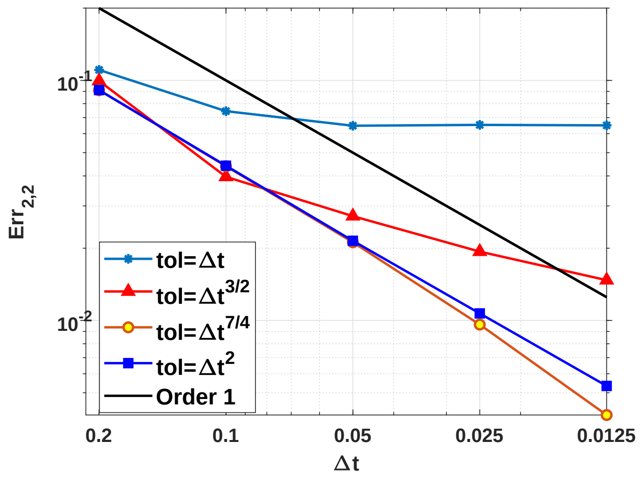
In this example, we use the TSA for both forward Euler and Heun's scheme with and without the pruning criteria (3.2). We compare two different approximations according to ℓ_2-relative error with the exact solution on the tree nodes

$$\mathcal{E}_2(t_n) = \sqrt{\frac{\sum\limits_{x_i \in \mathcal{T}^n} |v(x_i, t_n) - V^n(x_i)|^2}{\sum\limits_{x_i \in \mathcal{T}^n} |v(x_i, t_n)|^2}}.$$

TSA easily provides higher order converging methods only modifying the numerical scheme for ODEs and the quadrature formula for the cost functional. However, the case without pruning criteria becomes unfeasable for more than 10 time steps since it requires to store $O(M^{22})$ nodes applying Heun's scheme, whereas the application of pruning criteria (3.2) provides a real improvement. We are going to compute ℓ_2- error in time and in space

$$Err_{2,2} = \sqrt{\Delta t \sum_{n=0}^{N} \frac{\|v(x_i, t_n) - V^n(x_i)\|^2_{\ell^2(\mathcal{T}^n)}}{\|v\|^2_{\ell^2(\mathcal{T}^n)}}}.$$

Figure 4.1 shows the order of convergence for forward Euler and Heun's method using different $\varepsilon_{\mathcal{T}}$. We note that we obtain first order of convergence when dealing with Euler scheme and $\varepsilon_{\mathcal{T}} = \Delta t^2$ and second order for Heun's approximation with $\varepsilon_{\mathcal{T}} = \Delta t^3$. We also show how crucial is the selection of the tolerance $\varepsilon_{\mathcal{T}}$.



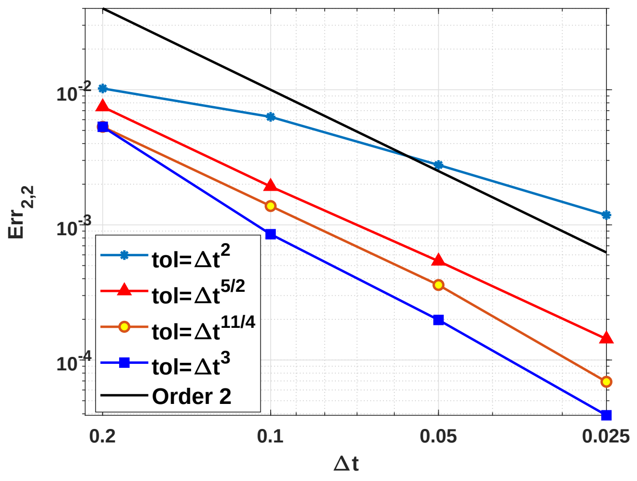


Figure 4.1: Test 1: Comparison of the order of convergence for the pruned TSA with different tolerances (top) with Euler method to approximate (2.1), (bottom) with Heun's method to approximate (2.1).

In Table 4.1 and Table 4.2 we present the results of the TSA applying the Euler scheme for $\varepsilon_{\mathcal{T}} = \{0, \Delta t^2\}$ respectively. We first note that the pruning criterium allows to solve the problem for a smaller temporal step size Δt since the cardinality of the tree is smaller. The CPU time is then proportional to the cardinality of the tree. We also note that, as expected, the order of convergence is 1 in both cases.

In Table 4.3 and Table 4.4 we present the results obtained by means of the Heun's method. Similar considerations to the tables which refer to Euler scheme hold true. However, we note that the order of convergence is improved as Heun's method is a second order scheme.

Finally, we note that to reach an error of order $O(10^{-3})$ using Euler method with pruning needs $150s$, $\Delta t = 0.0125$ and $|\mathcal{T}| = 252620$, whereas Heun's with pruning requires only $\Delta t = 0.2$, $|\mathcal{T}| = 160$ in $0.35s$. This shows that the choice of the numerical scheme of higher order allows accurate approximations in rea-

Δt	Nodes	CPU	$Err_{2,2}$	$Order_{2,2}$
0.2	63	0.05s	9.0e-02	
0.1	2047	0.35s	4.4e-02	1.04
0.05	2097151	1.1s	2.2e-02	1.02

Table 4.1: Test 1: Error analysis and order of convergence for forward Euler scheme of the TSA without pruning rule ($\varepsilon_{\mathcal{T}} = 0$).

Δt	Nodes	CPU	$Err_{2,2}$	$Order_{2,2}$
0.2	42	0.05s	9.1e-02	
0.1	324	0.08s	4.4e-02	1.05
0.05	3151	0.6s	2.1e-02	1.04
0.025	29248	2.5s	1.1e-02	1.005
0.0125	252620	150s	5.3e-03	1.004

Table 4.2: Test 1: Error analysis and order of convergence for forward Euler scheme of the TSA with $\varepsilon_{\mathcal{T}} = \Delta t^2$.

Δt	Nodes	CPU	$Err_{2,2}$	$Order_2$
0.2	1365	0.29s	3.51e-03	
0.1	1398101	3.92s	8.59e-04	2.0316

Table 4.3: Test 1: Error analysis and order of convergence for Heun's scheme of the TSA without pruning ($\varepsilon_{\mathcal{T}} = 0$).

Δt	Nodes	CPU	$Err_{2,2}$	$Order_2$
0.2	160	0.35s	5.32e-03	
0.1	2895	0.61s	8.53e-04	2.65
0.05	58888	60s	1.98e-04	2.11
0.025	1018012	9051s	3.9e-05	2.34

Table 4.4: Test 1: Error analysis and order of convergence for Heun's scheme of the TSA with $\varepsilon_{\mathcal{T}} = \Delta t^3$.

sonable time. However, one should also take into account that the comparison is performed with different pruning criteria. Thus, Euler scheme has order of convergence $O(\Delta t)$, whereas Heun's $O(\Delta t^2)$, which means that the same order is applied when $\Delta t_{euler} = \Delta t^2_{heun}$, if we apply Δt_{heun} for Heun's method. On the other hand the tolerance for Euler scheme will be Δt^4_{heun}, while for Heun's Δt^3_{heun}. To summarize Heun's scheme requires a bigger tolerance to obtain the same order of accuracy and, clearly, lower CPU time.

4.3.2 Test 2: Heat Equation

In the second test we deal with the control of the linear heat equation.

$$\begin{cases} y_t = \sigma y_{xx} + y_0(x)u(t) & (x,t) \in [0,1] \times [0,T], \\ y(0,t) = y(1,t) = 0 & t \in [0,T], \\ y(x,0) = y_0(x) & x \in [0,1], \end{cases} \tag{4.50}$$

and we set $T = 1$, $\sigma = 0.15$ and $y_0(x) = x - x^2$. The discretization of (4.50) with a centered finite difference method leads to a system of ODEs:

$$\begin{cases} \dot{y}(t) & = Ay(t) + Bu(t), \\ y(0) & = y_0 \end{cases} \tag{4.51}$$

where $A \in \mathbb{R}^{d\times d}$ is the stiffness matrix and $B \in \mathbb{R}^d$ is given by $B_i = y_0(x_i)$ for $i = 1, \dots, d$, where x_i are the points of the spatial grid. We want to minimize the following cost functional:

$$J_{y_0,t}(u) = \int_t^T \left(\|y(s)\|_2^2 + \frac{1}{100}|u(s)|^2 \right) ds + \|y(T)\|_2^2.$$

If the control is unconstrained, we can derive an *exact* solution solving the problem solving the Riccati differential equation as in (2.30). We will compare the numerical value function computed by the TSA and by the Riccati equation. We will modify the time steps and the number of controls for the TSA and compute the following relative errors

$$Err_2 = \frac{\sum_{n=0}^N |V(y_*^n, t_n) - v(y_R^n, t_n)|^2}{\sum_{n=0}^N |v(y_R^n, t_n)|^2}, \quad Err_\infty = \frac{\max\limits_{n=0,\dots,N} |V(y_*^n, t_n) - v(y_R^n, t_n)|}{\max\limits_{n=0,\dots,N} |v(y_R^n, t_n)|},$$

where $\{y_*^n\}_n$ is the optimal trajectory computed via TSA, whereas $\{y_R^n\}_n$ is obtained solving the Riccati equation. For both methods we set the spatial step size $\Delta x = 10^{-2}$, obtaining a system of dimension $d = 100$. We consider a time step equal to $\Delta t = 10^{-4}$ for the Riccati equation to obtain an accurate solution. To make a fair comparison, we first computed the LQR problem and then set the control space in the TSA. In this example we set $U = [-1, 0]$. To accelerate the algorithm, we consider the efficient pruning introduced in

Remark 3.2.2. We first create a coarse tree with $\Delta t = 0.1$ and only 2 discrete controls, we compute the SVD of the snapshots matrix and we consider the first ℓ columns as basis, the so-called *POD* basis. After that, during the construction of the tree with a finer discretization, we project the nodes onto this subspace which is divided into "buckets" of lengths equal to the pruning tolerance. We will compare only the nodes falling in the same bucket. In this case we choose $\ell = 1$ since the first singular value captures enough information of the dynamics. It is possible to consider directly a projected dynamics instead of the full dimensional one. We will discuss this kind of approach in Chapter 5.

Δt	Nodes	Pruned/Full	CPU	Err_2	Err_∞	$Order_2$	$Order_\infty$
0.1	134	4.7e-09	0.14s	0.279	0.241		
0.05	863	1.2e-18	0.65s	0.144	0.118	0.95	1.03
0.025	15453	3.1e-38	12.88s	5.5e-2	5.3e-2	1.40	1.17
0.0125	849717	3.8e-78	1.1e3s	1.6e-2	1.6e-2	1.77	1.42

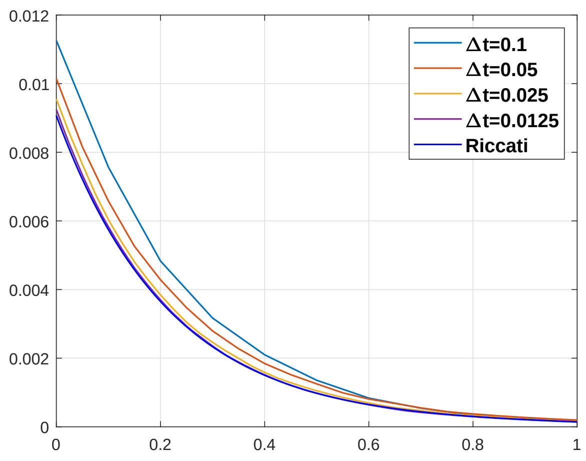
Table 4.5: Test 2: Error analysis and order of convergence for forward Euler scheme of the TSA with $\varepsilon_{\mathcal{T}} = \Delta t^2$ and 11 discrete controls.

Δt	Ratio pruned	Ratio Full
0.05	6.44	2.6e10
0.025	17.9	6.7e20
0.0125	984	4.5e41

Table 4.6: Test 2: Comparison between the ratio of cardinality for the full tree and the pruned tree with $\varepsilon_{\mathcal{T}} = \Delta t^2$ and 11 discrete controls.

In Table 4.5, we provide the order of convergence for the TSA with 11 discrete controls. We note that in LQR problem the control space is not discretized and that may provide some different results as the order of convergence. The table also shows the number of nodes corresponding for each Δt and the ratio of the number of nodes for the full and the pruned tree. The full tree grows with an order of $O(11^{\frac{1}{\Delta t}})$, while the pruned tree grows with a much smaller order as expressed by the ratio number. Table 4.6 shows the ratio cardinality between the full/pruned tree with $2\Delta t$ and Δt.

Finally, in the top panel of Figure 5.1 we show the convergence of the cost functional decreasing the temporal step size Δt for a given amount of controls. As it gets smaller, we are closer to what we consider the *exact* solution. In the bottom panel we show the optimal policy. Clearly, since our control space is not continuous, a chattering phenomenon appears, even if it does not influence the quality of our results.



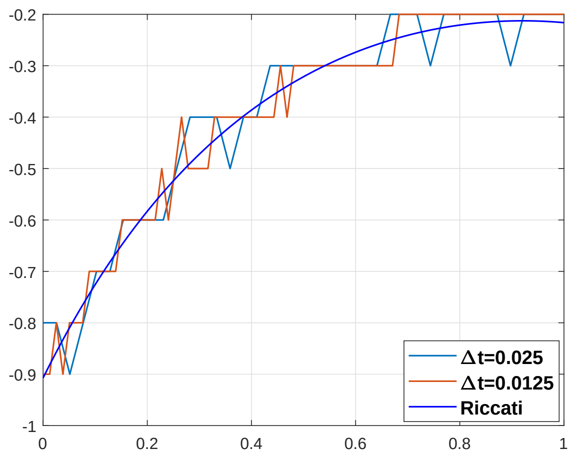


Figure 4.2: Test 2: Cost functional (top) and optimal control (bottom) with 11 discrete controls.

Chapter 5

Optimal control for PDEs

In this chapter we are going to treat a topic very useful in terms of applications: the optimal control of PDEs. We will first present the problem, then we will pass to introduce the Proper Orthogonal Decomposition (POD) and the Discrete Empirical Interpolation Method (DEIM). After that, we will study the coupling of the TSA with POD considering the coupling of the TSA with POD. Finally, we present a error estimate for the coupled method. We address the interested reader to [6] for a complete description.

5.1 Formulation of the problem

Let us consider an evolutive PDE and its spacial semidiscretization (though e.g. finite difference or finite elements methods). It will lead to a (possibly large) system of ODEs:

$$\begin{cases} \dot{y}(s) & = Ay(s) + F(t, y(s)) + Bu(s), \quad s \in (t, T], \\ y(t) & = x, \end{cases} \tag{5.1}$$

where $x \in \mathbb{R}^d$ is a given initial data, $A \in \mathbb{R}^{d\times d}, B \in \mathbb{R}^{d\times m}$ are given matrices and $F : [t, T] \times \mathbb{R}^d \to \mathbb{R}^d$ is a continuous function in both arguments and locally Lipschitz-type with respect to the second variable. We are considering a linear dependence on the control input just for simplicity, but all the proposed methods can be applied to more general frameworks.

In the same fashion of the previous chapters, we denote by $y : [t, T] \to \mathbb{R}^d$ the solution, by $u : [t, T] \to \mathbb{R}^m$ the control and by

$$\mathcal{U} = \{u : [t, T] \to U, \text{measurable}\}$$

the set of admissible controls where $U \subset \mathbb{R}^m$ is a compact set. We will assume that there exists a unique solution for (5.1) for each $u \in \mathcal{U}$.

In the numerical approximation of PDEs, the dimension of the problem is the number of spatial grid points or basis elements and it can be very large. We will see how to apply model order reduction techniques in order to reduce the spacial dimensionality.

To ease the notation we will denote the right hand side as follows:

$$f(y(t), u(t), t) := Ay(t) + F(t, y(t)) + Bu(t). \tag{5.2}$$

To select the optimal trajectory, we consider the following cost functional

$$J_{x,t}(u) := \int_t^T L(y(s, u), u(s), s)e^{-\lambda(s-t)}\, ds + g(y(T, u))e^{-\lambda(T-t)}, \tag{5.3}$$

where $L : \mathbb{R}^d \times \mathbb{R}^m \times [t, T] \to \mathbb{R}$ is the running cost, $g : \mathbb{R}^d \to \mathbb{R}$ is the final cost and $\lambda \geq 0$ is the discount factor. We will suppose that the functions $L(\cdot, u, t)$ and $g(\cdot)$ are Lipschitz continuous. The optimal control problem then reads:

$$\min_{u \in \mathcal{U}} J_{x,t}(u) \text{ s.t. } y(s) \text{ satisfies (5.1)}. \tag{5.4}$$

The final goal is the computation of the control in feedback form $u(t) = \eta(y(t), t)$, in terms of the state equation $y(t)$, where η is the feedback map. As explained before, we use the Dynamic Programming Principle and the corresponding HJB equation to obtain the value function, defined for an initial condition $(x, t) \in \mathbb{R}^d \times [0, T]$ as:

$$v(x, t) := \inf_{u \in \mathcal{U}} J_{x,t}(u). \tag{5.5}$$

Once the value function has been computed, it is possible to obtain the optimal feedback control as:

$$u^*(t) := \arg\min_{u \in U} \left\{ L(x, u, t) + \nabla v(x, t) \cdot f(x, u, t) \right\}. \tag{5.6}$$

5.2 Model order reduction and POD method

In this section first we recall the POD method for the state equation (5.1) and later how to apply it to reduce the dimension of the optimal control problem (5.4).

5.2.1 POD for the state equation

The solution of the system (5.1) may be very expensive and it is useful to deal with projection techniques to reduce the complexity of the problem. Here we recall the POD method and we refer the interested reader to [55, 57] for more details on the topic and to [10] for a review of different projection techniques.

Let us assume we have computed a numerical (or analytical if possible) solution of (5.1) on the time grid points t_j, $j \in \{0, \dots, N\}$ for some given control inputs. Then, we collect the *snapshots* $\{y(t_i)\}_{i=0}^N$ into the matrix $Y = [y(t_0), \dots, y(t_N)] \in \mathbb{R}^{d \times (N+1)}$. The aim of the method is to determine a

POD basis $\Psi = \{\psi_1, \ldots, \psi_\ell\}$ of rank $\ell \ll \min\{d, N+1\}$ to describe the set of data collected in time by solving the following minimization problem:

$$\min_{\psi_1,\ldots,\psi_\ell \in \mathbb{R}^d} \sum_{j=0}^{N} \left| y(t_j) - \sum_{i=1}^{\ell} \langle y(t_j), \psi_i \rangle \psi_i \right|^2 \quad \text{such that } \langle \psi_i, \psi_j \rangle = \delta_{ij}. \tag{5.7}$$

The associated norm is given by the Euclidean inner product $|\cdot|^2 = \langle \cdot, \cdot \rangle$. The solution of (5.7) is obtained by the SVD of the snapshots matrix $Y = \Psi \Sigma V^T$, where we consider the first $\ell-$columns $\{\psi_i\}_{i=1}^{\ell}$ of the orthogonal matrix Ψ. The selection of the rank of POD basis is based on the error computed in (5.7) which is related to the singular values neglected, *i.e.*

$$\mathcal{E}(\ell) = \frac{\sum_{i=1}^{\ell} \sigma_i^2}{\sum_{i=1}^{\min\{d,N+1\}} \sigma_i^2}, \tag{5.8}$$

where $\{\sigma_i\}_{i=1}^{\min\{d,N+1\}}$ are the singular values of Y.

However, the error strongly depends on the quality of the computed snapshots. This is clearly a limit when dealing with optimal control problems, since the control input is not known a-priori and it is necessary to have a reasonable forecast. In Section 5.3 we will explain how to select the control input $u(t)$ to solve (5.4).

To ease the notation, in what follows, we will denote by $\Psi \in \mathbb{R}^{d\times\ell}$ the POD basis of rank ℓ. Let us assume that the POD basis Ψ have been computed and make use of the following assumption to obtain a reduced dynamical system:

$$y(t) \approx \Psi y^\ell(t), \tag{5.9}$$

where $y^\ell(t)$ is a function from $[t, T]$ to $\mathbb{R}^\ell$. If we plug (5.9) into the full model (5.1) and exploit the orthogonality of the POD basis, the reduced model reads:

$$\begin{cases} \dot{y}^\ell(s) = A^\ell y^\ell(s) + \Psi^T F(s, \Psi y^\ell(s)) + B^\ell u(s), \\ y^\ell(t) = x^\ell, \end{cases} \tag{5.10}$$

where $A^\ell = \Psi^T A \Psi, B^\ell = \Psi^T B$ and $x^\ell = \Psi^T x \in \mathbb{R}^\ell$. We also note that $A^\ell \in \mathbb{R}^{\ell\times\ell}$ and $B^\ell \in \mathbb{R}^{\ell\times m}$. Error estimates for the reduced system (5.10) can be found in [42].

In what follows we are going to define the reduced dynamics as:

$$f^\ell(y^\ell(t), u(t), t) := A^\ell y^\ell(t) + \Psi^T F(t, \Psi y^\ell(t)) + B^\ell u(t). \tag{5.11}$$

Discrete Empirical Interpolation Method The solution of (5.10) is still computationally expensive, since the nonlinear term $F(t, \Psi y^\ell(t))$ depends on the dimension of the original problem, i.e. the variable $\Psi y^\ell(t) \in \mathbb{R}^d$. To avoid this issue, the *Empirical Interpolation Method* (EIM, [8]) and *Discrete Empirical Interpolation Method* (DEIM, [17]) were introduced.

The computation of the POD basis functions for the nonlinear part is related to the set of the snapshots $F(t_j, y(t_j))$, where $y(t_j)$ are already computed from (5.1). We denote by $\Phi \in \mathbb{R}^{d\times k}$ the POD basis functions of rank $k \ll \min\{d, N+1\}$ of the nonlinear part. The DEIM approximation of $f(t, y(t))$ is given in the following form:

$$F^{\text{DEIM}}(t, y^{\text{DEIM}}(t)) := \Phi(S^T\Phi)^{-1}F(t, y^{\text{DEIM}}(t)), \tag{5.12}$$

where $S \in \mathbb{R}^{d\times k}$ and $y^{\text{DEIM}}(t) := S^T\Psi y^\ell(t)$.Here, we assume that each component of the nonlinearity is independent from each other, i.e. we assume that $F(s,y) := [\bar{F}(s, y_1(s)), \dots, \bar{F}(s, y_d(s))]$, with $\bar{F} : [t,T] \times \mathbb{R} \to \mathbb{R}$, then the matrix S can be moved into the nonlinearity. Again, we refer to [17] for a complete description of the method and extensions to more general nonlinear functions. The role of the matrix S is to select interpolation points to evaluate the nonlinearity. The selection is made according to the LU decomposition algorithm with pivoting [17], or following the QR decomposition with pivoting [23]. We finally note that all the quantities in (5.12) are independent of the full dimension d, since the quantity $\Psi^T\Phi(S^T\Phi)^{-1} \in \mathbb{R}^{\ell\times k}$ can be precomputed. Typically the dimension k is much smaller than the full dimension. This allows the reduced order model to be completely independent of the full dimension as follows:

$$\begin{cases} \dot{y}^\ell(s) = A^\ell y^\ell(s) + \Psi^T F^{\text{ DEIM}}(s, y^{\text{DEIM}}(s)) + B^\ell u(s), \\ y^\ell(t) = x^\ell. \end{cases} \tag{5.13}$$

In what follows, we are going to define the reduced POD-DEIM dynamics as:

$$f^{\ell,\text{DEIM}}(y^\ell(s), u(s), s) := A^\ell y^\ell(s) + \Psi^T F^{\text{DEIM}}(s, S^T\Psi y^\ell(s)) + B^\ell u(s). \tag{5.14}$$

The DEIM error for a given snapshots set $\widetilde{F} = \{F(t_j, y(t_j))\}_{j=0}^N$ and its DEIM approximation $\widetilde{F}^{\text{DEIM}} = \Phi(S^T\Phi)^{-1}S^T\widetilde{F}$ is given by:

$$\|\widetilde{F} - \widetilde{F}^{\text{DEIM}}\|_2 \le c\|(I - \Phi\Phi^T)\widetilde{F}\|_2, \qquad \text{with } c = \|(S^T\Phi)^{-1}\|_2, \tag{5.15}$$

as shown in [17, 23].

5.2.2 POD for the optimal control problem

The key ingredient to compute feedback control is the knowledge of the value function, which is expressed by a nonlinear PDE whose dimension is given by the dimension of (5.1). It is clear that its approximation is very expensive. Therefore, we are going to apply the POD method to reduce the dimension of the dynamics and then solve the corresponding (reduced) discrete DPP which is now feasible and defined below. Let us first define the reduced running cost and the reduced final cost as

$$L^\ell(x^\ell, u, t) = L(\Psi x^\ell, u, t), \quad g^\ell(x^\ell) = g(\Psi x^\ell).$$

Next, we introduce the reduced optimal control problem for (5.4). For a given control u, we denote by $y^\ell(s,u)$ the unique solution to (5.13) at time s. Then, the reduced cost is given by

$$J^\ell_{x^\ell,t}(u) = \int_t^T L^\ell\big(y^\ell(s,u),u(s),s\big)e^{-\lambda(s-t)}\,ds + g^\ell(y^\ell(T))e^{-\lambda(T-t)}, \tag{5.16}$$

and, the POD approximation for (5.4) reads as follows:

$$\min_{u\in U} J^\ell_{x^\ell,t}(u) \quad \text{such that} \quad y^\ell(t) \text{ solves (5.10)}. \tag{5.17}$$

Finally, we define the reduced value function $v^\ell(x^\ell,t)$ as

$$v^\ell(x^\ell,t) := \inf_{u\in\mathcal{U}} J^\ell_{x^\ell,t}(u) \tag{5.18}$$

and the reduced HJB equation:

$$\begin{cases} \dfrac{\partial v^\ell}{\partial s}(x^\ell,s) - \lambda v^\ell(x^\ell,s) + \min\limits_{u\in U}\left\{L^\ell(x^\ell,u,s) + \nabla v^\ell(x^\ell,s)\cdot f^\ell(x^\ell,u,s)\right\} = 0, \\ v^\ell(x^\ell,T) = g^\ell(x^\ell), \qquad\qquad (x^\ell,s)\in\mathbb{R}^\ell\times[t,T). \end{cases} \tag{5.19}$$

Alternatively, one could further approximate the nonlinear term using DEIM and replace the dynamics (5.10) with (5.13) in (5.17), providing an impressive acceleration of the algorithm as shown in the section of numerical tests.

5.3 HJB-POD method on a tree structure

In this section we explain, step by step, how to use model reduction techniques on a tree structure in order to obtain an efficient approximation of the value function and to deal with complex problems such as PDEs. We also provide an error estimate for the presented method.

Computation of the snapshots When applying POD for optimal control problems there is a major bottleneck: the choice of the control inputs to compute the snapshots. Thus, we store the tree $\mathcal{T} = \cup_{n=0}^N \mathcal{T}^n$ for a chosen Δt and discrete control set U. This set turns out to be a very good candidate for the snapshots matrix since it delivers all the possible trajectories we want to consider. To summarize the snapshots set is $Y = \mathcal{T} = \cup_{n=0}^N \mathcal{T}^n$.

In the numerical tests, we will use $\Delta t = 0.1$ and 2 controls to compute the snapshots and it will be sufficient to catch the main features of the controlled problem.

Computation of the basis functions The computation of the basis Ψ has been described in Section 5.2. We are going to solve the following optimization problem:

$$\min_{\psi_1,\ldots,\psi_\ell\in\mathbb{R}^d} \sum_{j=1}^{N} \sum_{\underline{u}_j\subset U^j} \left| y(t_j,\underline{u}_j) - \sum_{i=1}^{\ell} \langle y(t_j,\underline{u}_j),\psi_i\rangle \psi_i \right|^2 \quad \text{such that } \langle \psi_i,\psi_j\rangle = \delta_{ij}, \tag{5.20}$$

where $\underline{u}_j = (u_1,\ldots,u_j) \subset U^j = U\times\ldots\times U$ and

$$y(t_j,\underline{u}_j) = y_0 + \Delta t \sum_{k=0}^{j-1} f(y_k,u_{k+1},t_k).$$

In this context we have no restrictions on the choice of the number of basis ℓ, since we will solve the HJB equation on a tree structure. In former works , e.g. [43, 5], the authors were restricted to choose $\ell \approx 4$ to have a feasible reduction of the HJB equation. Here, the dimension of the state variable is not a major issue. On the other hand, the pruning strategy will turn out to be crucial for the feasibility of the problem.

It is well-known that the error in (5.7) is given by the sum of the singular values neglected. We recall that we will choose ℓ such that $\mathcal{E}(\ell) \approx 0.999$, with $\mathcal{E}(\ell)$ defined in (5.8).

Construction of the reduced tree Having computed the POD basis, we build a new tree which might consider a different Δt and/or a finer control space with respect to the snapshots set. We will denote the projected tree as $\mathcal{T}^\ell$ with its generic $n-$th level given by:

$$\mathcal{T}^{n,\ell} = \{\zeta_i^{n-1,\ell} + \Delta t f^\ell(\zeta_i^{n-1,\ell},u_j,t_{n-1})\}_{j=1}^{M} \quad i=1,\ldots,M^{n-1},$$

where the reduction of the nonlinear term f^ℓ can be done via POD or POD-DEIM as in (5.12). The first level of the tree is clearly given by the projection of the initial condition, i.e. $\mathcal{T}^{0,\ell} = \Psi^T x$. Then, the procedure follows the full dimensional case, but with the projected dynamics. We will show how this approach speeds up the method keeping high accuracy. Even if we have reduced the dimension of the problem, the cardinality of the tree $\mathcal{T}^{n,\ell}$ depends on the number of the discrete controls and the time step chosen as in the high-dimensional case. It is clear that each resolution of the PDE will be faster, but it is still necessary to apply a pruning rule which reads:

$$\|\zeta_i^{n,\ell} - \zeta_j^{n,\ell}\| \le \varepsilon_{\mathcal{T}}, \text{ for } i\neq j \text{ and } n=0,\ldots,\overline{N}. \tag{5.21}$$

As proposed in [3], the evaluation of (5.21) can be computed in a more efficient way, considering the most variable components by the principal component analysis. This technique is incorporated in our algorithm, since we have already computed the POD basis and the most variable component turns out to be the first one y_1^ℓ. It will be sufficient to reorder the nodes according to their first components to accelerate the pruning criteria.

Approximation of the reduced value function The numerical reduced value function $V^\ell(x^\ell, t)$ will be computed on the tree nodes in space as

$$V^\ell(x^\ell, t_n) = V^{n,\ell}(x^\ell), \quad \forall x^\ell \in \mathcal{T}^{n,\ell}. \tag{5.22}$$

Then, the computation of the reduced value function follows directly from the DPP. Defined the grid $\mathcal{T}^{n,\ell} = \{\zeta_j^{n,\ell}\}_{j=1}^{M^n}$ for $n = 0, \dots, \overline{N}$, we can write a time discretization for (5.19) as follows:

$$\begin{cases} V^{n,\ell}(\zeta_i^{n,\ell}) = \min\limits_{u\in U}\{e^{-\lambda\Delta t}V^{n+1,\ell}(\zeta_i^{n,\ell} + \Delta t f^\ell(\zeta_i^{n,\ell}, u, t_n)) + \Delta t\, L^\ell(\zeta_i^{n,\ell}, u, t_n)\}, \\ \qquad\qquad\qquad\qquad \zeta_i^{n,\ell} \in \mathcal{T}^{n,\ell}, n = \overline{N} - 1, \dots, 0, \\ V^{\overline{N},\ell}(\zeta_i^{\overline{N},\ell}) = g^\ell(\zeta_i^{\overline{N},\ell}), \qquad\qquad\qquad\qquad \zeta_i^{\overline{N},\ell} \in \mathcal{T}^{\overline{N},\ell}. \end{cases} \tag{5.23}$$

Computation of the feedback control The computation of the feedback control strongly relies on the fact we deal with a discrete control set U. Indeed, when we compute the reduced value function, we store the control indices corresponding to the arg min in (5.23). The optimal trajectory is than obtained by following the path of the tree with the controls chosen such that

$$\begin{gathered} u_*^{n,\ell} := \arg\min_{u\in U}\left\{e^{-\lambda\Delta t}V^{n+1,\ell}(\zeta_*^{n,\ell} + \Delta t f^\ell(\zeta_*^{n,\ell}, u, t_n)) + \Delta t\, L^\ell(\zeta_*^{n,\ell}, u, t_n)\right\}, \\ \zeta_*^{n+1,\ell} \in \mathcal{T}^{n+1,\ell} \; s.t. \; \zeta_*^{n,\ell} \to^{u_n^*} \zeta_*^{n+1,\ell}, \end{gathered} \tag{5.24}$$

for $n = 0, \dots, \overline{N} - 1$, where the symbol $\to^u$ stands for the connection of two nodes by the dynamics corresponding to the control u.

Once the control $u_*^{n,\ell}$ has been computed, we plug it into the high dimensional problem (5.1) and compute the optimal trajectory.

5.4 Error estimates for the HJB-POD method on a TSA

In this section we derive an error estimate for the HJB-POD approximation (5.23) on a tree structure. We recall the hypothesis considered in the previous chapters. We assume that the functions f, L, g are bounded:

$$\begin{gathered} |f(x, u, s)| \le M_f, \quad |L(x, u, s)| \le M_L, \quad |g(x)| \le M_g, \\ \forall\, x \in \mathbb{R}^d, u \in U \subset \mathbb{R}^m, s \in [t, T], \end{gathered} \tag{5.25}$$

the functions f and L are Lipschitz-continuous with respect to the first variable

$$\begin{gathered} |f(x, u, s) - f(y, u, s)| \le L_f|x - y|, \quad |L(x, u, s) - L(y, u, s)| \le L_L|x - y|, \\ \forall\, x, y \in \mathbb{R}^d, u \in U \subset \mathbb{R}^m, s \in [t, T], \end{gathered} \tag{5.26}$$

and the cost g is also Lipschitz-continuous:

$$|g(x) - g(y)| \leq L_g|x - y|, \quad \forall x, y \in \mathbb{R}^d. \tag{5.27}$$

Furthermore, let us assume that the functions L and g are semiconcave

$$\begin{aligned}&L(x + z, u, t + \tau) - 2L(x, u, t) + L(x - z, u, t - \tau) \leq C_L(|z|^2 + \tau^2),\\ &g(x + z) - 2g(x) + g(x - z) \leq C_g|z|^2, \qquad \forall x, z \in \mathbb{R}^d, u \in U, t, \tau \geq 0,\end{aligned} \tag{5.28}$$

and assume that f verifies the following inequality:

$$\begin{aligned}&|f(x + z, u, t + \tau) - 2f(x, u, t) + f(x - z, u, t - \tau)| \leq C_f(|z|^2 + \tau^2),\\ &\qquad \forall u \in U,\ \forall x, z \in \mathbb{R}^d,\ \forall t, \tau \geq 0.\end{aligned} \tag{5.29}$$

We also introduce the continuous-time extension of the DDP

$$\begin{aligned}&V(x, s) = \min_{u\in U}\{e^{-\lambda(t_{n+1}-s)}V(x + (t_{n+1} - s)f(x, u, s), t_{n+1}) + (t_{n+1} - s)\, L(x, u, s)\},\\ &V(x, T) = g(x), \qquad\qquad x \in \mathbb{R}^d, s \in [t_n, t_{n+1}),\end{aligned} \tag{5.30}$$

and the POD version for the continuous-time extension (5.30) which reads:

$$\begin{aligned}V^\ell(x^\ell, s) = \min_{u\in U}\{&e^{-\lambda(t_{n+1}-s)}V^\ell(x^\ell + (t_{n+1} - s)f^\ell(x^\ell, u, s), t_{n+1})+\\ &+ (t_{n+1} - s)\, L^\ell(x^\ell, u, s)\},\\ V^\ell(x^\ell, T) = g^\ell(x^\ell),& \qquad\qquad x^\ell \in \mathbb{R}^\ell, s \in [t_n, t_{n+1}).\end{aligned} \tag{5.31}$$

Given the exact solution $v(x, s)$ and its POD discrete approximation $V^\ell(x^\ell, s)$, we prove the following theorem which provides an error estimate for the proposed method.

Theorem 5.4.1. *Let us assume* (5.25) – (5.29) *hold true, then there exists a constant* $C(T)$ *such that*

$$\sup_{s\in[t,T]} |v(x, s) - V^\ell(x^\ell, s)| \leq C(T) \left(\left(\sum_{i\geq\ell+1} \sigma_i^2 \right)^{1/2} + \Delta t \right), \tag{5.32}$$

where $\{\sigma_i\}$ *are the singular values of the snapshots matrix.*

Proof. We observe that, by triangular inequality, the approximation error can be decomposed in two parts:

$$|v(x, s) - V^\ell(x^\ell, s)| \leq |v(x, s) - V(x, s)| + |V(x, s) - V^\ell(x^\ell, s)|. \tag{5.33}$$

An error estimate for the first term has been already obtained in Section 4:

$$\sup_{(x,s)\in\mathbb{R}^d\times[0,T]} |V(x,s)-v(x,s)| \le \widehat{C}(T)\Delta t. \tag{5.34}$$

Let us focus on the second term of the right hand side of (5.33). Without loss of generality, we consider $\lambda = 0$. For $s = T$, the estimate follows directly by the assumptions on g. Considering $x \in \mathbb{R}^d$ and $s \in [t_n, t_{n+1})$, we can write

$$V(x,s) - V^\ell(x^\ell, s) \le$$

$$V(x_{n+1}, t_{n+1}) - V^\ell(x^\ell_{n+1}, t_{n+1}) + (t_{n+1}-s)\left(L(x,u^n_*,s) - L^\ell(x^\ell,u^n_*,s)\right) \le$$

$$V(x_{n+1}, t_{n+1}) - V^\ell(x^\ell_{n+1}, t_{n+1}) + (t_{n+1}-s)\, L_L|x-\Psi x^\ell|, \tag{5.35}$$

where u^n_*, x_{n+1} and x^ℓ_{n+1} are defined as

$$u^n_* = \operatorname*{arg\,min}_{u\in U}\left\{V^\ell(x^\ell + (t_{n+1}-s)f^\ell(x^\ell,u,s), t_{n+1}) + (t_{n+1}-s)\, L^\ell(x^\ell,u,s)\right\},$$

$$x_{n+1} = x + (t_{n+1}-s)f(x,u^n_*,s), \qquad x^\ell_{n+1} = x^\ell + (t_{n+1}-s)f^\ell(x^\ell,u^n_*,s).$$

We define the trajectory path and its POD approximation respectively as

$$x_m := x + \sum_{k=n}^{m-1}\alpha_k f(x_k, u^k_*, \bar{t}_k), \quad x^\ell_m := x^\ell + \sum_{k=n}^{m-1}\alpha_k f^\ell(x^\ell_k, u^k_*, \bar{t}_k),$$

where

$$\alpha_k = \begin{cases} t_{n+1}-s & k=n \\ \Delta t & k\ge n+1 \end{cases}, \quad \bar{t}_k = \begin{cases} s & k=n \\ t_k & k\ge n+1 \end{cases},$$

$$u^k_* = \operatorname*{arg\,min}_{u\in U}\left\{V^\ell\left(x^\ell_k + \alpha_k f^\ell(x^\ell_k,u,\bar{t}_k), t_{k+1}\right) + \alpha_k L^\ell(x^\ell_k,u,\bar{t}_k)\right\}, k \ge n,$$

with $x_n = x$ and $x^\ell_n = x^\ell$. Then, iterating (5.35) we obtain

$$V(x,s) - V^\ell(x^\ell,s) \le L_L \sum_{m=n}^{\overline{N}-1}\alpha_m|x_m - \Psi x^\ell_m| + L_g|x_{\overline{N}} - \Psi x^\ell_{\overline{N}}|. \tag{5.36}$$

Defining

$$\eta_m = \begin{cases} L_L\alpha_m & m\in\{n,\dots \overline{N}-1\} \\ L_g & m=\overline{N} \end{cases},$$

we can write

$$V(x,s) - V^\ell(x^\ell,s) \le \sum_{m=n}^{\overline{N}} \eta_m|x_m - \Psi x^\ell_m|.$$

By triangular inequality and Cauchy-Schwarz inequality, we can write

$$V(x,s) - V^\ell(x^\ell,s) \le \sum_{m=n}^{\overline{N}} \eta_m \left(|x_m - \mathcal{P}^\ell x_m| + |\mathcal{P}^\ell x_m - \Psi x_m^\ell|\right) \le$$

$$\left(\sum_{m=n}^{\overline{N}} \eta_m^2\right)^{1/2} \left(\left(\sum_{m=n}^{\overline{N}} |x_m - \mathcal{P}^\ell x_m|^2\right)^{1/2} + \left(\sum_{m=n}^{\overline{N}} |\mathcal{P}^\ell x_m - \Psi x_m^\ell|^2\right)^{1/2} \right), \tag{5.37}$$

where $\mathcal{P}^\ell = \Psi^T\Psi$ is a projection operator. Since $\{x_m\}_m \subset \mathcal{T}$, by the definition of POD basis we get

$$\left(\sum_{m=n}^{\overline{N}} |x_m - \mathcal{P}^\ell x_m|^2\right)^{1/2} \le \left(\sum_{i\ge \ell+1} \sigma_i^2\right)^{1/2}. \tag{5.38}$$

Let us denote by $Err(\ell) = \left(\sum_{i\ge\ell+1}\sigma_i^2\right)^{1/2}$ the error related to the orthogonal projection onto V^ℓ. Let us focus now on the generic term $|\mathcal{P}^\ell x_m - \Psi x_m^\ell|$:

$$|\mathcal{P}^\ell x_m - \Psi x_m^\ell| \le \sum_{k=n}^{m-1} \alpha_k \|\mathcal{P}^\ell\|_2 |f(x_k, u_*^k, \bar{t}_k) - f(\Psi x_k^\ell, u_*^k, \bar{t}_k)| \le$$

$$L_f\|\mathcal{P}^\ell\|_2 \sum_{k=n}^{m-1} \alpha_k |x_k - \Psi x_k^\ell| \le L_f \|\mathcal{P}^\ell\|_2 \sum_{k=n}^{m-1} \alpha_k \left(|x_k - \mathcal{P}^\ell x_k| + |\mathcal{P}^\ell x_k - \Psi x_k^\ell|\right).$$

By the discrete Grönwall's lemma and noticing that $\|\mathcal{P}^\ell\|_2 = 1$, we get

$$|\mathcal{P}^\ell x_m - \Psi x_m^\ell| \le L_f \sum_{k=n}^{m-1} \alpha_k |x_k - \mathcal{P}^\ell x_k| e^{L_f(t_m - s)},$$

and since $\alpha_k \le \Delta t \ \forall k$, we obtain

$$\left(\sum_{m=n}^{\overline{N}} |\mathcal{P}^\ell x_m - \Psi x_m^\ell|^2\right)^{1/2} \le \sqrt{T-s} L_f e^{L_f(T-s)} Err(\ell). \tag{5.39}$$

Inserting (5.38) and (5.39) in (5.37) we get

$$V(x,s) - V^\ell(x^\ell, s) \le Err(\ell) \left(\sum_{m=n}^{\overline{N}} \eta_m^2\right)^{1/2} \left(\sqrt{T} L_f e^{L_f T} + 1\right).$$

Finally, noticing that

$$\sum_{m=n}^{\overline{N}} \eta_m^2 \le (TL_L)^2 + L_g^2,$$

we obtain

$$V(x,s) - V^\ell(x^\ell, s) \le C_1(T) Err(\ell),$$

where

$$C_1(T) = \left((TL_L)^2 + L_g^2\right)^{1/2} \left(\sqrt{T} L_f e^{L_f T} + 1\right).$$

Analogously, it is possible to obtain the same estimate for $V^\ell(x^\ell, s) - V(x,s)$ and, defining $C(T) = \max\{\widehat{C}(T), C_1(T)\}$, we get the desired result.

$\square$

Remark 5.4.1. *The error estimate presented in Theorem 5.4.1 depends strongly on the initial condition, since the POD reduction is based on the tree generated by the starting point x. We can extend the error estimate to other initial conditions if we enlarge the snapshots set with these new data and their evolutions up to the final time T.*

5.5 Numerical Tests

In this section we apply our proposed algorithm to show the effectiveness of the method with three test cases. In the first test we consider the one dimensional heat equation and we will compute the error and the order according to the solution of the Riccati equation. We will also consider the techniques for the feedback reconstruction introduced in Chapter 3.3. The second test deals with a parabolic PDE with a polynomial nonlinear term, which is usually not a trivial task when applying open-loop control tools. Finally, we will consider the bilinear control of the viscous Burgers' equation.

For the semidiscretization we use a Finite Difference scheme and we integrate in time using an implicit Euler scheme. In presence of non-linearities, we will apply the Newton's method with tolerance equal to 10^{-4}. We will denote by U_n the discretized set of U with n equi-distributed controls.

5.5.1 Test 1: Heat equation

In this first example we consider the one dimensional heat equation with homogeneous Dirichlet boundary condition

$$\begin{cases} \partial_t y(x,t) = \sigma y_{xx}(x,t) + y_0(x)u(t) & (x,t) \in \Omega \times [0,T], \\ y(x,t) = 0 & (x,t) \in \partial\Omega \times [0,T], \\ y(x,0) = y_0(x) & x \in \Omega, \end{cases} \tag{5.40}$$

where the control $u(t)$ is taken in the admissible set $\mathcal{U} = \{u : [0,T] \to [-1,0]\}$, $\sigma = 0.15$, $T = 1$ and $\Omega = [0,1]$. This example has been considered also in Section 4.3.2 since we are able to compute the error and the order of the method comparing the numerical solution with the solution given by the Riccati equation. First, we create a rough tree with two discrete controls and

$\Delta t = 0.1$. We compute the SVD of the snapshots matrix generated by the tree and we consider $\ell = 2$ basis, getting a projection error $Err = 6.9 \cdot 10^{-4}$ and a ratio $\mathcal{E}(\ell) = 0.999998$. We note that in this case two POD basis get enough information for a complete description of the system. This is a typical situation when dealing with parabolic problems. Then, given the reduced dynamics, we compute the reduced tree structure and value function with 11 discrete controls and we compare the numerical value function computed by POD-TSA and by the Riccati equation according to the following errors

$$Err_2 = \frac{\sum_{n=0}^{N} |V(y_*^n, t_n) - v(y_R^n, t_n)|^2}{\sum_{n=0}^{N} |v(y_R^n, t_n)|^2}, \quad Err_\infty = \frac{\max\limits_{n=0,\dots,N} |V(y_*^n, t_n) - v(y_R^n, t_n)|}{\max\limits_{n=0,\dots,N} |v(y_R^n, t_n)|}, \tag{5.41}$$

where $\{y_*^n\}_n$ is the optimal trajectory computed via POD-TSA, while $\{y_R^n\}_n$ is the solution of the Riccati equation. The results are presented in Table 5.1. Moreover, in Table 5.2 we recall the values obtained for the full dimensional problem. The coupling of the TSA with POD leads to better results in less time. In particular, we see that for $\Delta t = 0.0125$ we obtain a speed-up of a factor 7 and a reduction of order 4 for the cardinality of tree. In Figure 5.1 we can observe the convergence of the cost functional and the approximation of the optimal control.

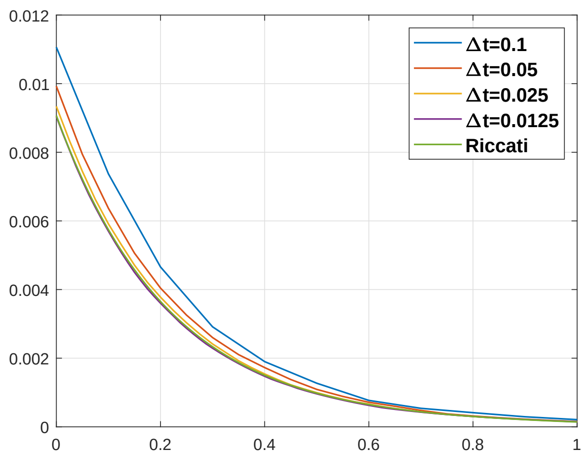
Δt	Nodes	Pruned/Full	CPU	Err_2	Err_∞	$Order_2$	$Order_\infty$
0.1	134	4.3e-10	0.1s	0.244	0.220		
0.05	825	1.0e-19	0.56s	0.102	9.4e-2	1.25	1.22
0.025	11524	2.1e-39	8.74s	3.1e-2	3.0e-2	1.73	1.67
0.0125	194426	7.8e-80	151s	1.0e-2	8.2e-3	1.60	1.85

Table 5.1: Test 1: Error analysis and order of convergence for TSA-POD method with $\varepsilon_{\mathcal{T}} = \Delta t^2$, 11 discrete controls and 2 POD basis.

Δt	Nodes	Pruned/Full	CPU	Err_2	Err_∞	$Order_2$	$Order_\infty$
0.1	134	4.7e-09	0.14s	0.279	0.241		
0.05	863	1.2e-18	0.65s	0.144	0.118	0.95	1.03
0.025	15453	3.1e-38	12.88s	5.5e-2	5.3e-2	1.40	1.17
0.0125	849717	3.8e-78	1.1e3s	1.6e-2	1.6e-2	1.77	1.42

Table 5.2: Test 1: Error analysis and order of convergence for forward Euler scheme of the TSA with $\varepsilon_{\mathcal{T}} = \Delta t^2$ and 11 discrete controls.

Feedback reconstruction In this paragraph we are going to apply the feedback reconstruction techniques introduced in Chapter 3.3. We reconsider



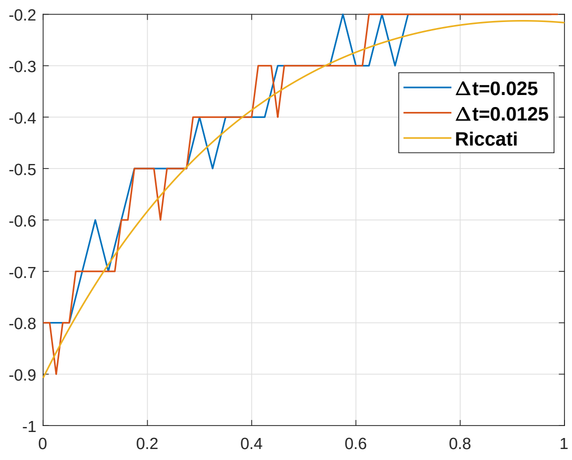


Figure 5.1: Test 1: Cost functional (top) and optimal control (bottom) with 11 discrete controls.

2 POD basis and we solve the optimal control problem via POD-TSA with 3 discrete controls. The results for this case are presented in Table 5.3.

Δt	Nodes	Pruned/Full	CPU	Err_2	Err_∞	$Order_2$	$Order_\infty$
0.1	122	4.6e-04	0.02s	0.376	0.283		
0.05	689	4.4e-08	0.19s	0.178	0.136	1.08	1.06
0.025	9536	1.7e-16	2.3s	0.107	6.9e-2	0.73	0.98
0.0125	155293	2.3e-34	37s	0.0655	0.0394	0.71	0.80

Table 5.3: Test 1: Error analysis and order of convergence for TSA-POD method with $\varepsilon_{\mathcal{T}} = \Delta t^2$, 3 discrete controls and 2 POD basis.

Then, we pass to the post-processing procedure: we consider Algorithm 4, computing the optimal trajectory/control with a finer control set. We consider a control set $\widetilde{U}$ with 100 discrete controls. In Table 5.4 we present the errors and the orders according to the definition (5.41).

Δt	CPU	Err_2	Err_∞	$Order_2$	$Order_\infty$
0.1	0.03s	0.315	0.250		
0.05	0.07s	9.6e-2	0.100	1.71	1.32
0.025	0.74s	2.5e-2	3.1e-2	1.93	1.68
0.0125	25s	1.4e-2	9.0e-3	0.89	1.81

Table 5.4: Test 1: Error analysis and order of convergence for TSA-POD method with $\varepsilon_{\mathcal{T}} = \Delta t^2$, 2 POD basis and reconstruction with 100 controls.

Since we have constructed the tree with three discrete controls, we can apply also the quadratic feedback reconstruction presented in Algorithm 5. The results for this case are presented in Table 5.5. In Figure 5.2 we show the cost functional and the optimal control with all these techniques. In particular, it is possible to see from the plot of the optimal control that the quadratic feedback reconstruction is more stable, while the reconstruction by comparison presents a scattering behaviour.

Δt	CPU	Err_2	Err_∞	$Order_2$	$Order_\infty$
0.1	0.02s	0.251	0.229		
0.05	0.04s	0.109	9.5e-2	1.21	1.27
0.025	0.63s	3.3e-2	3.0e-2	1.71	1.65
0.0125	24s	1.1e-2	5.9e-3	1.58	2.36

Table 5.5: Test 1: Error analysis and order of convergence for TSA-POD method with $\varepsilon_{\mathcal{T}} = \Delta t^2$, 2 POD basis and quadratic reconstruction.

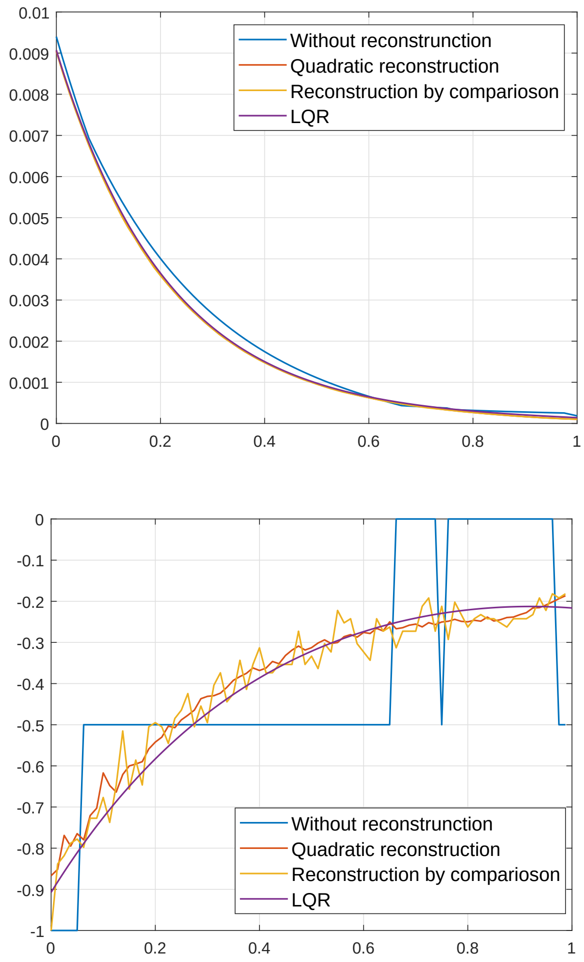


Figure 5.2: Test 1: Cost functional (top) and optimal control (bottom) with different techniques for the feedback reconstruction.

5.5.2 Test 2: Nonlinear reaction diffusion equation

In this example we consider the following bidimensional PDE with polynomial nonlinearity and homogeneous Neumann boundary conditions

$$\begin{cases} \partial_t y(x,t) = \sigma\Delta y(x,t) + \mu\left(y^2(x,t) - y^3(x,t)\right) + y_0(x)u(t) & (x,t) \in \Omega \times [0,T], \\ \partial_n y(x,t) = 0 & (x,t) \in \partial\Omega \times [0,T], \\ y(x,0) = y_0(x) & x \in \Omega, \end{cases} \tag{5.42}$$

where the control $u(t)$ is taken in the admissible set $\mathcal{U} = \{u : [0,T] \to [-2,0]\}$ and $\Omega = [0,1]^2$. In (5.42) we consider: $T = 1, \sigma = 0.1, \mu = 5$ and $y_0(x_1,x_2) = sin(\pi x_1)sin(\pi x_2)$. We discretize the space domain Ω in 31 points in each direction, obtaining a discrete domain with $d = 961$ points. As shown in Figure 5.3, the solution of the uncontrolled equation (5.42) (i.e. $u(t) \equiv 0$) converges asymptotically to the stable equilibrium $\overline{y}_1(x) = 1$.

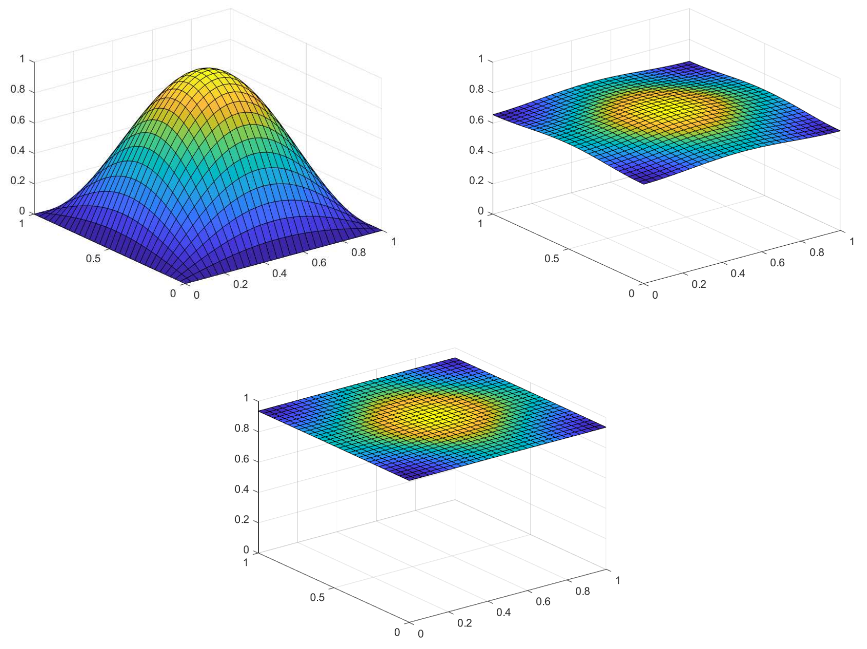


Figure 5.3: Test 2: Uncontrolled solution for equation (5.42) for time $t = \{0, 0.5, 1\}$ (from top-left to bottom).

Our aim is to steer the solution to the unstable equilibrium $\overline{y}_2(x) = 0$. For this reason, we introduce the following cost functional

$$J_{y_0,t}(u) = \int_t^T \left(\int_\Omega |y(x,s)|^2 dx + \frac{1}{100}|u(s)|^2\right) ds + \int_\Omega |y(x,T)|^2 dx. \tag{5.43}$$

Case 1: Full TSA We first consider the results using the TSA without model order reduction. In Figure 5.4 we report the optimal trajectory obtained using the full tree structure algorithm with 2 controls. As one can see, we steer

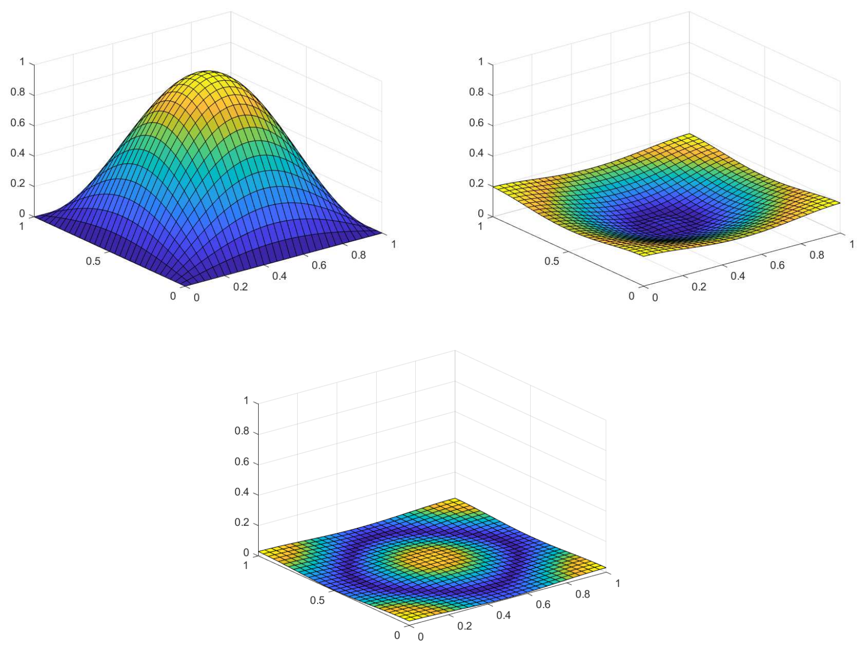


Figure 5.4: Test 2: Controlled solution with TSA for equation (5.42) with full tree for time $t = \{0, 0.5, 1\}$ (from top-left to bottom) with U_2.

the solution to the unstable equilibrium using $U_2 = \{-2, 0\}$ as discrete control set.

In the left panel of Figure 5.5, we show the control policy obtained with 2, 3 and 4 discrete controls. In the right panel we show the behaviour of the cost functional, and it is easy to check that the optimal trajectories are very similar. An analysis of the CPU time is provided in Table 5.6 and discussed below.

Case 2: TSA with POD The computation of the full TSA is already expensive with only 3 controls. For this reason, we replace the dynamics with its reduced order modeling. Then, we set the number of POD basis $\ell = 6$ such that $\mathcal{E}(\ell) = 0.999$. Similarly, we consider 6 DEIM basis for the nonlinear term. In what follows, whenever we will talk about POD, we will refer to POD-DEIM approach.

The snapshots matrix Y is computed with a full TSA using the discrete control space U_2 and $\Delta t = 0.1$. In the online stage we considered again $\Delta t = 0.1$, a pruning criteria with $\varepsilon_{\mathcal{T}} = \Delta t^2$ and different discrete controls.

In the top-left panel of Figure 5.6 we show the optimal policy with a number of controls varying from two up to five. As one can see comparing the left panel of Figure 5.5 and the top-left panel of Figure 5.6, there is no difference in terms of optimal control between the high dimensional case discretized

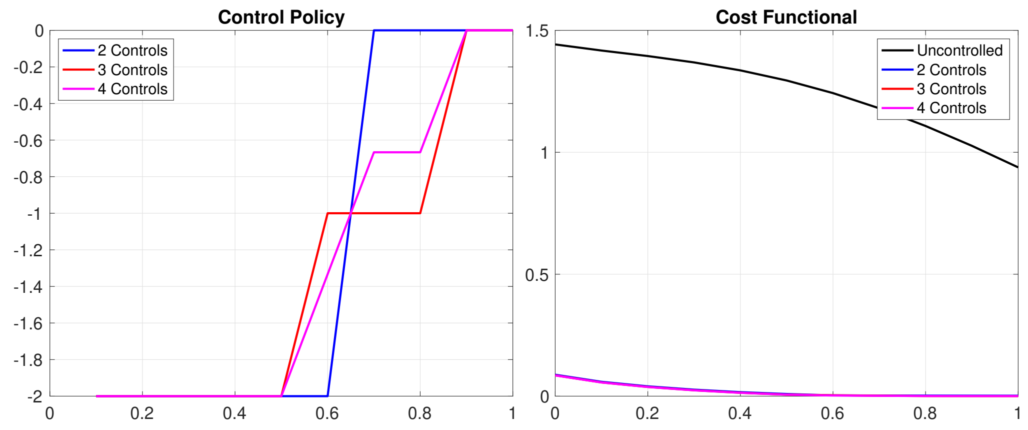


Figure 5.5: Test 2: Control policy (left) and cost functional (right) for U_2, U_3 and U_4.

with Finite Difference and the low dimensional case obtained via POD. We remind that the optimal trajectory is obtained plugging the suboptimal control u_*^ℓ into the high dimensional model. Finally, in the bottom panel of Figure 5.6 we show a zoom of the cost functional $J_{y_0,0}$ and it is possible to see the improvement obtained using more controls.

The CPU time, expressed in seconds, is shown in Table 5.6. The online phase of the TSA-POD is always faster than the full TSA. We tried to compute the full TSA with 5 controls and we stopped the computation after 4 days. If we also consider the amount of time to compute the snapshots, the offline phase, using the TSA with 2 controls and then running online, e.g. the TSA-POD with 3 controls, we get a speed up of factor 10 with respect to the full problem, having the same approximation.

	U_2	U_3	U_4	U_5
TSA	$5.8312s$	241.5773s	3845.77s	> 4 days
TSA-POD	0.5157s	19.7969s	432.0990s	$1.0871e+04$s

Table 5.6: Test 2: CPU time of the TSA and the TSA-POD with a different number of controls and pruning rule $\epsilon_{\mathcal{T}} = 0.01$.

Remark 5.5.1. *The offline stage of the proposed method is clearly expensive due to the cardinality of the tree. We have also tried to compute snapshots for some given control input setting, e.g. $u(t) \equiv \overline{u}$, with $\overline{u} \in \{-2, -1, 0\}$. In this setting we are able to achieve the same results shown in the section, improving the computational performances of the method in the offline phase.*

Remark 5.5.2. *Using the same set of snapshots, we can perform the online simulation with $\Delta t = 0.05$ and U_2. The results for the optimal control and cost functional can be found in Figure 5.7.*

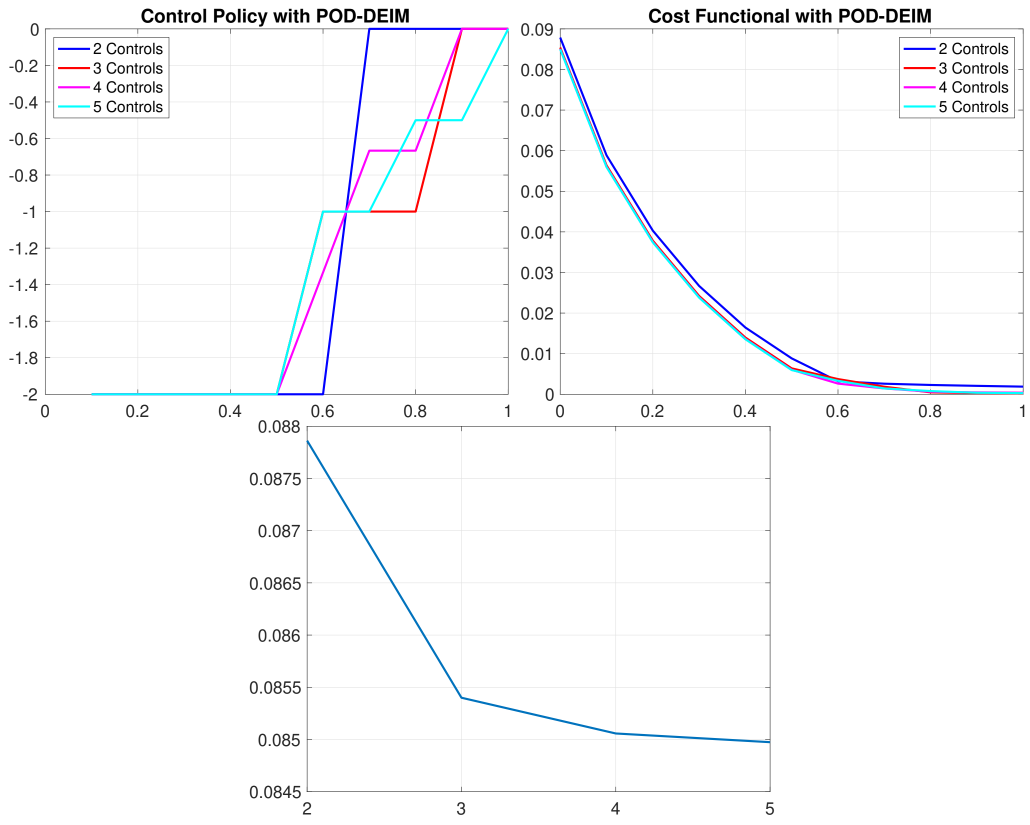


Figure 5.6: Test 2: Optimal policy (top-left), cost functional (top-right) and $J_{y_0,0}$ (bottom) for U_n with $n = \{2, 3, 4, 5\}$.

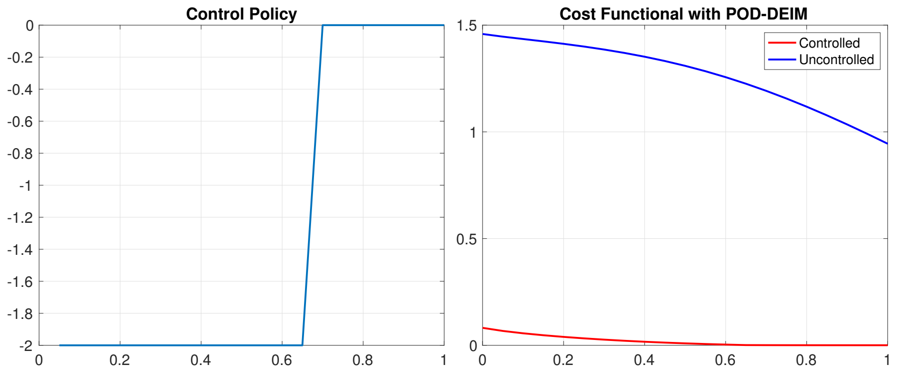


Figure 5.7: Test 2: Optimal policy (left) and cost functional (right) with $\Delta t = 0.05$ and U_2.

5.5.3 Test 3: Viscous Burgers' equation

In the last example we consider the well-known viscous Burgers' equation with homogeneous Dirichlet boundary conditions:

$$\begin{cases} \partial_t y(x,t) = \sigma \Delta y(x,t) + y(x,t) \cdot \nabla y(x,t) + y(x,t)u(t) & (x,t) \in \Omega \times [0,T], \\ y(x,t) = 0 & (x,t) \in \partial\Omega \times [0,T], \\ y(x,0) = y_0(x) & x \in \Omega, \end{cases} \tag{5.44}$$

where the control $u(t)$ is taken in the admissible set $\mathcal{U} = \{u : [0,T] \to [-2,0]\}$ and $\Omega = [0,1]^2$. In (5.44) we consider: $T = 1, \sigma = 0.01$ and $y_0(x_1,x_2) = sin(\pi x_1) sin(\pi x_2)$. We discretize the space domain in 41 points in each direction, obtaining a problem of dimension $d = 1681$ points. In Figure 5.8 we show the solution of the uncontrolled equation (5.44) for different time instances. Our aim is to steer the solution to the steady state $\tilde{y}(x) = 0$, using the cost functional (5.43), as in Test 2, using a bilinear control, e.g. controlling the system through a reaction term.

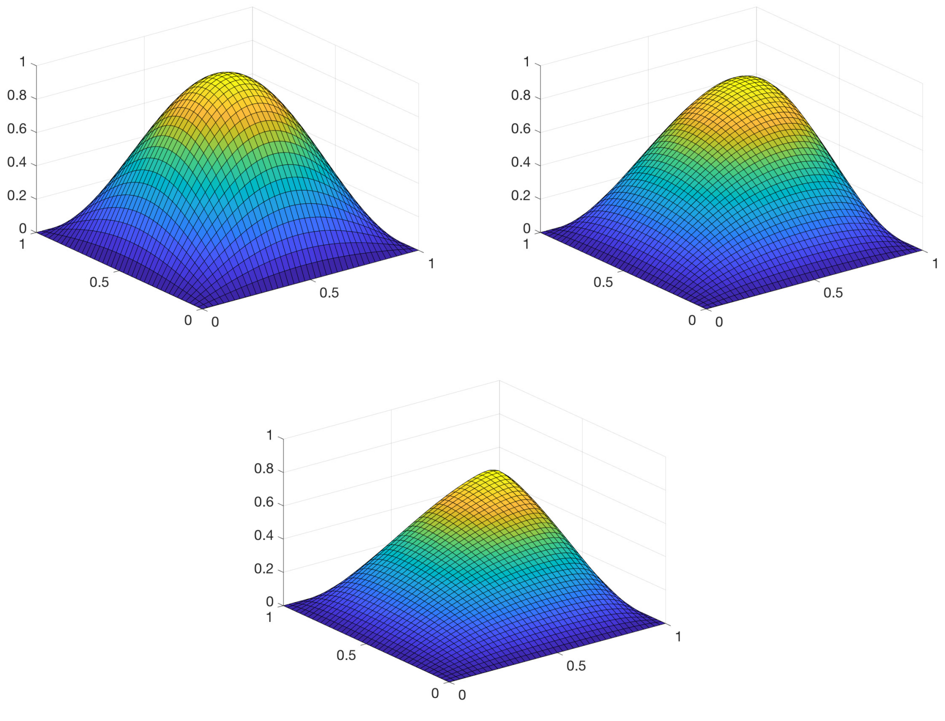


Figure 5.8: Test 3: Uncontrolled solution for equation (5.44) for time instances $t = \{0, 0.5, 1\}$ (from top-left to bottom).

Case 1: Full TSA Let us first consider the results of the full TSA. In Figure 5.9 we show the results of the controlled problem. As we can see, the

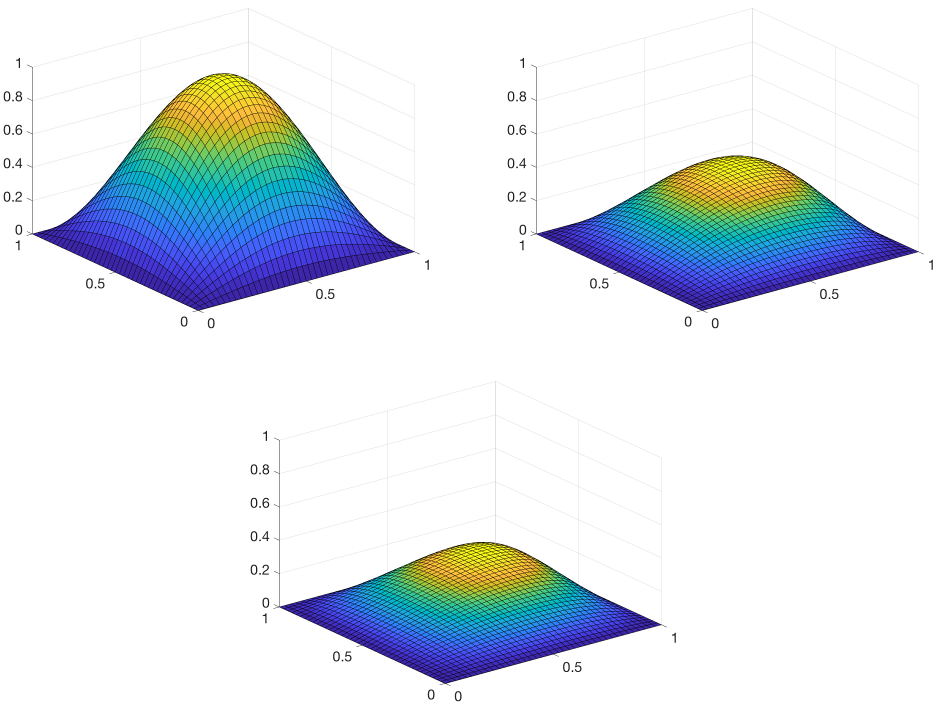


Figure 5.9: Test 3: Controlled solution with 3 controls for equation (5.44) with full tree for time $t = \{0, 0.5, 1\}$ (from top-left to bottom).

solution gets close to $\tilde{y}(x)$ as expected. We also note that for this example the viscosity term σ is rather low, making the problem hard to be controlled.

In the left panel of Figure 5.10, we show the optimal control computed to obtain the controlled solution. When the control set is only given by 2 controls, the algorithm uses the control $u^*(t) = -2$ for $0 \leq t \leq 0.7$ and $u(t) = 0$ for $0.7 < t \leq 1$, whereas with 3 controls we use the control -2 for $0 \leq t \leq 0.5$ and -1 for $0.5 < t \leq 1$. We can see that passing from 2 to 3 controls, we obtain a slightly better result in terms of cost functional (see the right panel of Figure 5.10).

Finally, the cardinality of the full tree is reported in Table 5.7. We can observe that the tree is considerably pruned compared to the previous example. This happens when we deal with a bilinear control for both Test 2 and Test 3.

	U_2	U_3	U_4	U_5
TSA with $\varepsilon_{\mathcal{T}} = 0$	2047	88573		
TSA with $\varepsilon_{\mathcal{T}} = 0.01$	1681	17680		
TSA-POD with $\varepsilon_{\mathcal{T}} = 0.01$	1717	17627	48372	83201

Table 5.7: Test 3: Cardinality of the tree for the full TSA and for the pruned TSA and pruned TSA-POD with $\varepsilon_{\mathcal{T}} = 0.01$, varying the control sets.

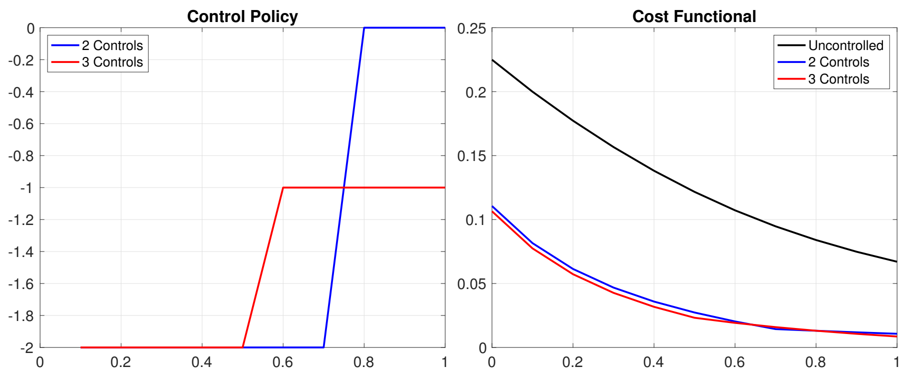


Figure 5.10: Test 3: Optimal policy (left) and cost functional for U_2 and U_3 (right).

Case 2: TSA with POD To accelerate and use a finer control set, we use model order reduction. The snapshots are computed with $\Delta t = 0.1$, U_2 and a pruning criteria with $\varepsilon_{\mathcal{T}} = \Delta t^2$. For this problem, we only project the dynamics with POD since the nonlinear term can be written as a tensor and can be projected offline. We took $\ell = 8$ POD basis to have $\mathcal{E}(\ell) = 0.999$. Thanks to the reduced problem, we are able to solve the problem with more controls, keeping $\Delta t = 0.1$. In the top-left panel of Figure 5.11 we show the behaviour of the optimal policy. We note that the cases with 2 and 3 controls are equivalent to the full case (compare with Figure 5.10). The computed controls show a chattering behaviour which is then reflected in the plot of $J_{y_0,0}$ in the bottom panel of Figure 5.11, considering the control space U_n for $n = \{2, 3, \dots, 11\}$. In the top right panel, we show $J_{y_0,t}$ for $t \in [0, 1]$. We can see a rather similar behaviour when increasing the number of controls.

The CPU time is reported in the bottom panel of Figure 5.11 and it is possible to capture visually the big advantage of using model order reduction.

With the same set of snapshots, we can also decrease the temporal step size, e.g. $\Delta t = 0.05$, and compute the online stage with U_n with $n = 2, 3$. We see in Figure 5.12 that the behaviour of the control policy is similar when dealing with 2 controls, whereas the switch from $u = -2$ to $u = -1$ happens for $t = 0.45$.

We can also observe that the cost functional is slightly lower when dealing with $\Delta t = 0.05$ as summarized in Table 5.8.

Δt	U_2	U_3
0.1	0.1106	0.1065
0.05	0.0995	0.0956

Table 5.8: Test 3: Cost functional $J^{\ell}_{y_0,0}$ with $\Delta t = \{0.1, 0.05\}$, U_2 and U_3.

The cardinality of the pruned TSA-POD approach is reported in the last line of Table 5.7, whereas the first line is still valid for the full TSA-POD

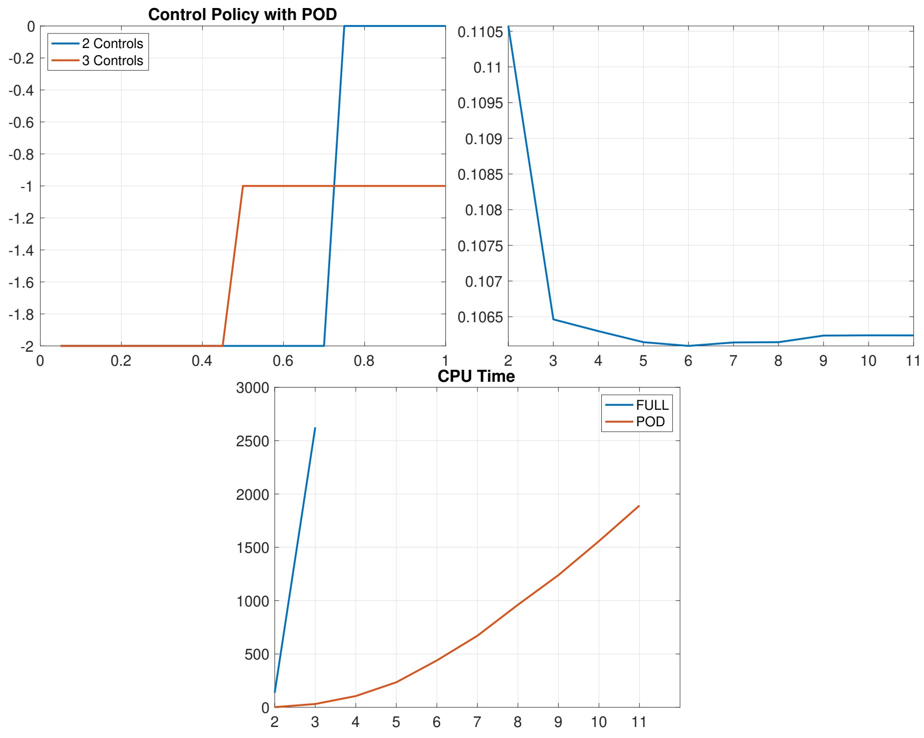


Figure 5.11: Test 3: Optimal policy (top-left), zoom of the cost for U_n with $n = 2, 3, 4, \ldots, 11$ (top-right) and CPU time increasing the number of controls (bottom).

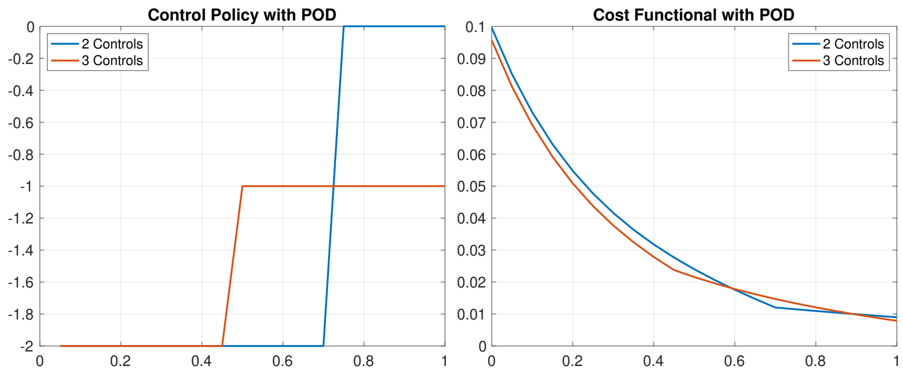


Figure 5.12: Test 3: Optimal policy (left) and cost functional for U_n with $n = 2, 3$ and $\Delta t = 0.05$ (right).

method. As expected, even when we apply model reduction, we can observe an impressive pruning if we compare with the unpruned method.

Chapter 6

Conclusions

We have proposed a novel method to approximate time dependent HJB equations, related to optimal control problems, via DP scheme on a tree structure. The proposed algorithm creates a tree structure according to all the possible directions of the controlled dynamical system for a finite set of controls. This procedure has several advantages with respect to the DP algorithm based on the classical time and space discretization. The first advantage is that we do not need to choose a space grid and a local space interpolation. Furthermore, TSA does not require a priori fictitious numerical domain Ω to set the numerical scheme and, consequently, there is no need to impose boundary conditions. Since the cardinality of the tree increases as the number of the control increases and the time step size decreases, we introduced a pruning criteria to reduce the complexity and to save in memory allocations and CPU time. Thus, the complexity of the problem is drastically reduced, cutting all the branches laying in a small neighbourhood. After the application of the pruning criteria, the efficiency of TSA is greatly improved in terms of CPU time. The whole technique can be coupled with a post-processing procedure allowing for the synthesis of the feedback control via an interpolation step on the data structure. This approach allows to apply the DP method to high-dimensional problems as has been shown in the numerical section for both ODEs and PDEs. We have also considered the extension of the algorithm to high-order schemes and by numerical simulation we noticed that we achieve the same order of convergence of the underlying scheme.

Furthermore, we have proved error estimates for the presented algorithm. In particular, we have shown that with a tree structure we can achieve the same order of convergence of the numerical method used in the time discretization of the dynamics. Our error estimate improves previous existing results on the convergence of the semi-discrete value function. Indeed, in [27] under the assumptions of Lipschitz-continuity of the data, the authors proved a convergence of the scheme with order $\frac{1}{2}$. In [51] we extended the previous result proving a first order convergence under the further assumption of semi-concavity for the data. We have also shown that if the pruning technique has a reasonable tolerance, e.g. one order higher of the order of convergence of the numerical method of the ODE, we can achieve the same order of the TSA method without pruning.

Consequently, we have presented a new method that couples model order reduction with the TSA. The tree structure needs to solve many PDEs for a given control input and, therefore, model order reduction helps to speed up its construction and also to work with a finer discretization of the control set. We have also proved an error estimate to guarantee the convergence of the method which depends, as expected, on the projection error of the POD method and on the temporal discretization of the differential equations considered. We showed through numerical tests the efficiency of the method and we would like to emphasize that the tree structure algorithm combined with model order reduction allows to solve numerical optimal control problems for nonlinear PDEs.

Future directions

Due to its flexibility, the TSA can be extended to stochastic optimal control problems, exploiting the semi-Lagrangian scheme introduced by Camilli and Falcone ([13]). Moreover, we plan to analyze the extension of the TSA algorithm to other classical optimal control problems, such as state constrained, the infinite time horizon problems and the minimum time problem. A further research direction will be the insurance of robustness with respect to perturbations, since the TSA finds the optimal trajectory only on the tree structure. Finally, we showed that we can get a synthesis of the feedback control considering a post-processing step. In this thesis we considered the low-dimensional case, but we would like to explore the high-dimensional case in the next future, considering for instance the interpolation on scattered data based on the Radial Basis function (see [58, 52] for a detailed description of this method).

Bibliography

[1] E. G. Al'brekht, *On the optimal stabilization of nonlinear systems*, J. Appl. Math. Mech., 25 (1961), pp. 254–266.

[2] A. Alla, M. Falcone, and D. Kalise, *An Efficient Policy Iteration Algorithm for Dynamic Programming Equations*, SIAM Journal on Scientific Computing, 37 (2015), pp. A181–A200.

[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 (2019), pp. A2384–A2406.

[4] ——, *High-order approximation of the finite horizon control problem via a tree structure algorithm*, IFAC-PapersOnLine, 52 (2019), pp. 19–24.

[5] A. Alla, M. Falcone, and S. Volkwein, *Error Analysis for POD Approximations of Infinite Horizon Problems via the Dynamic Programming Approach*, SIAM J. Control Optim., 55 (2017), p. 3091–3115.

[6] A. Alla and L. Saluzzi, *A HJB-POD approach for the control of nonlinear PDEs on a tree structure.* to appear on Applied Numerical Mathematics.

[7] M. Bardi and I. Capuzzo-Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*, Modern Birkhäuser Classics, Birkhäuser Boston, 2008.

[8] M. Barrault, N. Nguyen, Y. Maday, and A. Patera, *An "empirical interpolation" method: Application to efficient reduced-basis discretization of partial differential equations*, Comptes Rendus Mathematique, 339 (2004), p. 667–672.

[9] R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[10] P. Benner, S. Gugercin, and K. Willcox, *A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems*, SIAM Rev., 57 (2015), pp. 483–531.

[11] R. Brent, *Algorithms for minimization without derivatives*, Prentice-Hall, 1973.

[12] S. Cacace, E. Cristiani, M. Falcone, and A. Picarelli, *A patchy dynamic programming scheme for a class of hamilton–jacobi–bellman equations*, SIAM Journal on Scientific Computing, 34 (2012), pp. A2625–A2649.

[13] F. Camilli and M. Falcone, *An approximation scheme for the optimal control of diffusion processes*, ESAIM: Mathematical Modelling and Numerical Analysis, 29 (1995), pp. 97–122.

[14] F. Camilli, M. Falcone, P. Lanucara, and A. Seghini, *A domain decomposition method for bellman equations*, in Domain Decomposition Methods in Scientific and Engineering Computing, D. E. Keyes and J. Xu, eds., Contemp. Math. 180, American Mathematical Society, Providence, RI, (1994), pp. 477–483.

[15] P. Cannarsa and C. Sinestrari, *Semiconcave Functions, Hamilton-Jacobi Equations, and Optimal Control*, Progress in Nonlinear Differential Equations and Their Applications, Birkhäuser Boston, 2004.

[16] I. Capuzzo-Dolcetta and H. Ishii, *On the rate of convergence in homogenization of Hamilton-Jacobi equations*, Indiana University Mathematics Journal, 50 (2001), pp. 1113–1128.

[17] S. Chaturantabut and D. Sorensen, *Nonlinear Model Reduction via Discrete Empirical Interpolation*, SIAM J. Sci. Comput., 32 (2010), pp. 2737–2764.

[18] P. Combettes and J. Pesquet, *Proximal Thresholding Algorithm for Minimization over Orthonormal Bases*, SIAM Journal on Optimization, 18 (2008), pp. 1351–1376.

[19] M. G. Crandall and P. L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 277 (1983), pp. 1–42.

[20] R. Cuninghame-Green, *Minimax algebra and applications*, Fuzzy Sets and Systems, 41 (1991), pp. 251–267.

[21] J. Darbon and S. Osher, *Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere*, Research in the Mathematical Sciences, 3 (2016), p. 19.

[22] J. Darbon and S. Osher, *Splitting enables overcoming the curse of dimensionality*, in Splitting Methods in Communication, Imaging, Science, and Engineering, Springer International Publishing, 2016, pp. 427–432.

[23] Z. Drmac and S. Gugercin, *A new selection operator for the discrete empirical interpolation method - improved a priori error bound and extensions*, SIAM J. Sci. Comput., 38 (2016), pp. A631–A648.

[24] L. C. Evans, *Partial Differential Equations*, American Mathematical Society, 2010.

[25] M. Falcone and R. Ferretti, *Discrete time high-order schemes for viscosity solutions of Hamilton-Jacobi-Bellman equations*, Numerische Mathematik, 67 (1994), pp. 315–344.

[26] ——, *Semi-Lagrangian Approximation Schemes for Linear and Hamilton-Jacobi Equations*, SIAM, 2013.

[27] M. Falcone and T. Giorgi, *An Approximation Scheme for Evolutive Hamilton-Jacobi Equations*, in Stochastic Analysis, Control, Optimization and Applications, Birkhäuser Boston, 1999, pp. 289–303.

[28] M. Falcone, P. Lanucara, and A. Seghini, *A splitting algorithm for hamilton-jacobi-bellman equations*, Appl. Numer. Math., 15 (1994), pp. 207–218.

[29] R. Fedkiw and S. Osher, *Level Set Methods and Dynamic Implicit Surfaces*, Springer New York, 2002.

[30] R. Ferretti, *High-order approximations of linear control systems via runge-kutta schemes*, Computing, 58 (1997), pp. 351–364.

[31] A. Festa, *Reconstruction of independent sub-domains for a class of hamilton–jacobi equations and application to parallel computing*, ESAIM: Mathematical Modelling and Numerical Analysis, 50 (2016), pp. 1223–1240.

[32] ——, *Domain decomposition based parallel howard's algorithm*, Mathematics and Computers in Simulation, 147 (2018), pp. 121–139.

[33] W. Fleming and W. McEneaney, *A max-plus based algorithm for an HJB equation of nonlinear filtering*, SIAM J. Control and Optim., 38 (2000), p. 683–710.

[34] W. H. Fleming and H. M. Soner, *Controlled Markov Processes and Viscosity Solutions*, Springer New York, 2005.

[35] T. Goldstein and S. Osher, *The Split Bregman Method for L1-Regularized Problems*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 323–343.

[36] L. Grüne, *Handbook of Model Predictive Control*, Springer-Verlag GmbH, 2018.

[37] L. Grüne and P. Kloeden, *Higher order numerical schemes for affinely controlled nonlinear systems*, Numerische Mathematik, 89 (2001), pp. 669–690.

[38] E. Hopf, *Generalized Solutions of non-linear Equations of First Order*, Journal of Mathematics and Mechanics, 14 (1965), pp. 951–973.

[39] I. Jollie, *Principal component analysis*, Springer Series in Statistics, 2002.

[40] D. Kalise, A. Kroener, and K. Kunisch, *Local minimization algorithms for dynamic programming equations*, SIAM Journal on Scientic Computing, 38 (2016), pp. A1587–A1615.

[41] G. Kirsten and V. Simoncini, *Order reduction methods for solving large-scale differential matrix Riccati equations.* arXiv:1905.12119, 2019.

[42] K. Kunisch and S. Volkwein, *Galerkin Proper Orthogonal Decomposition Methods for a General Equation in Fluid Dynamics*, SIAM J. Numer. Anal., 40 (2002), pp. 492–515.

[43] K. Kunisch, S. Volkwein, and L. Xie, *HJB-POD-Based Feedback Design for the Optimal Control of Evolution Problems*, SIAM Journal on Applied Dynamical Systems, 3 (2004), pp. 701–722.

[44] L.Grüne and J. Pannek, *Nonlinear Model Predictive Control*, Springer-Verlag GmbH, 2011.

[45] W. M. McEneaney, *A Curse-of-Dimensionality-Free Numerical Method for Solution of Certain HJB PDEs*, SIAM Journal on Control and Optimization, 46 (2007), pp. 1239–1276.

[46] ——, *Convergence Rate for a Curse-of-dimensionality-Free Method for Hamilton–Jacobi–Bellman PDEs Represented as Maxima of Quadratic Forms*, SIAM Journal on Control and Optimization, 48 (2009), pp. 2651–2685.

[47] C. Navasca and A. J. Krener, *Patchy solutions of hamilton-jacobi-bellman partial differential equations*, in Lecture Notes in Control and Information Sciences, Springer Berlin Heidelberg, (2007), pp. 251–270.

[48] K. Pearson, *On Lines and Planes of Closest Fit to Systems of Points in Space*, Philosophical Magazine, 2 (1901), pp. 559–572.

[49] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishechenko, *The mathematical theory of optimal processes*, Interscience, New York, (1962).

[50] W. T. Reid, *Riccati Differential Equations*, Academic Press, New York, 1972.

[51] L. Saluzzi, A. Alla, and M. Falcone, *Error estimates for a tree structure algorithm for dynamic programming equations.* arXiv:1812.11194, 2019.

[52] G. Santin and B. Haasdonk, *Kernel Methods for Surrogate Modeling.* arXiv:1907.10556, 2019.

[53] J. Sethian and A.Vladimirsky, *Ordered upwind methods for static hamilton–jacobi equations: Theory and algorithms*, SIAM Journal on Numerical Analysis, 41 (2003), pp. 325–363.

[54] J. A. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, 2009.

[55] L. Sirovich, *Turbulence and the dynamics of coherent structures. parts I-II*, Quarterly of Applied Mathematics, XVL (1987), pp. 561–590.

[56] Y. Tsai, L. Cheng, S. Osher, and H. Zhao, *Fast sweeping algorithms for a class of hamilton-jacobi equations*, SIAM J. Numer. Anal., 41 (2004), pp. 673–694.

[57] S. Volkwein, *Model Reduction using Proper Orthogonal Decomposition.* Lecture Notes, University of Konstanz, 2013.

[58] H. Wendland, *Scattered Data Approximation*, Cambridge University Press, 2009.