Universidad de Valladolid

Facultad de Ciencias

Trabajo Fin de Máster

Máster en Matemáticas

Numerical resolution of Dynamic Games.

Autor: Beatriz Gómez Martín.

Tutor: Víctor Gatón Bustillo.

Contents

Chapter 1

Introduction

Nowadays, the necessity of understanding the world around us leads to the study of complex mathematical models. Modelling physical, social or biological phenomena helps researchers to understand system evolution and to study the effects of the different factors influencing the behaviour of a system. Furthermore, models can be employed to make predictions on the performance of an event.

The focus of the present work is on dynamical systems. Dynamical systems are defined as systems that evolve over time. These systems can be studied through differential equation models. In some cases, there exist one or more variables which can be controlled by an agent, and that determine the dynamic evolution of the system. The objective is then to study the strategy that an agent has to follow in order to achieve a certain objective. This is called an Optimal Control problem.

Another aspect to take into account is that a model does not always present only a single agent. Then, it is essential to study the relations among several agents. The branch of mathematics that investigates this relations is called Game Theory. Applying the results in this area, the aim is to achieve an equilibrium point where all agents maximize their benefits.

In order to obtain the previous equilibrium point, it would be necessary to solve the dynamics of the problem. However, analytical solution of differential models is hardly ever conceivable. This is the reason why the design of numerical methods to approximate system solutions has taken great importance. Additionally, in the recent years this field of study has been boosted by the computational improvements.

However, even though the computational capacity has improved, the cost required for solving most practical problems becomes too large. It is important to note that most numerical methods require a discretization of the problem dynamics.

Let us illustrate the size of a regular problem with an example. Suppose a system evolving over a d-dimensional spatial region within a given period of time. When the space is discretized, the solution must be computed separately for each node. Taking N nodes in each dimension, N^d nodes must be computed. If a refined result is required, for example doubling each dimension nodes, the magnitude is multiplied by 2^d. Furthermore, a time discretization is also required, so that the numerical method must be computed for each node at each time step. Finally, the same problem has to be solved for each of the agents in the system. All these conditions result in a computationally expensive numerical methods.

For this reason, it is crucial to develop efficient algorithms for computing numerical approximations to the solutions. The idea is to search the optimal strategy for each agent iteratively until an equilibrium point is achieved. There exist two popular algorithms from Control Theory that can be employed to find an equilibrium point in a multi-agent dynamical system: *Value iteration* and *Policy iteration*.

Both algorithms require the interpolation of certain value functions that represent the agents objective function. In this work, a Chebyshev interpolation scheme is proposed. The Chebyshev interpolation is a very useful technique that can be applied to obtain numerical approximations in different problems such as global optimization, ODE solving or integration. Chebyshev polynomials are especially helpful because they provide very precise and numerically efficient approximations.

In essence, the objective of the work is to study methods for finding a numerical approximation to the solution for multi-agent dynamical systems. Two different numerical methods are employed to compute the Markovian Nash equilibrium for a multi-player differential game: Value iteration and

Policy iteration.

The present document is organized as follows. In Section 2.1, the most important results in Optimal Control Theory are detailed. Pontryagin Maximum Principle (2.3) and the Hamilton-Jacobi-Bellman theorem (2.7) are the key results in this section. In Section 2.2 Game Theory is presented and different equilibrium concepts are explained in this section.

Theoretical aspects and numerical methods to compute the Chebyshev polynomial interpolation are explained in Chapter 3. Afterwards, two numerical methods to solve multi-agent games are detailed in Chapter 4. These methods are Value iteration, in Section 4.1, and Policy iteration, in Section 4.2. The previous algorithms are implemented with a Chebyshev interpolation approach.

Finally, a numerical example is detailed in Chapter 5. A transboundary pollution model developed in [9], which was numerically solved with a spline-based numerical method, is presented. The model proposes a N-players game and, for particular cases, analytical solutions are quite straightforward to obtain. Therefore, it is a very useful model to make an error analysis of the numerical methods. Moreover, the performance of the numerical methods can be compared with the spline-based solution. The results of the numerical solution of more than two player situations are also shown.

Finally, the conclusions of the work are presented, together with future work possibilities.

Chapter 2

From Optimal Control to Game Theory

2.1 Optimal Control Theory

The Optimal Control Theory is a branch of mathematics that studies the way to determinate the behaviour of a system in order to maximize a certain benefit. The systems to control are dynamical systems, that is, systems that evolve over time.

For a given dynamical system, a control is a variable which can be adjusted by an agent and that determines the evolution of the system. Then, the objective in an Optimal Control problem is to determine a control such that the benefits are maximized (or in some problems, such that a cost is minimized).

Formally, the Optimal Control Theory is a branch of mathematics that deals with finding a control for a given dynamical system over a period of time, such that an optimality criterion is achieved.

Let us illustrate the previous Optimal Control problem definition with a simple example, similar to the presented in [19, Ch 1].

Example 2.1. *Consider a promoter who owns an initial capital K_0. This capital at time t, denoted $K = K(t)$, can be inverted or consumed. $I = I(t)$ denotes the capital inverted, and $C = C(t)$ denotes the consumption. It is* *satisfied that $K(t) = C(t) + I(t),\ t \in [t_0, t_1]$.*

On one hand, the consumed fraction of the capital gives to the promoter a certain satisfaction, that can be measured by a utility function. Utility functions are assumed to be strictly increasing ($U'(C) > 0$) and concave ($U''(C) \leq 0$). Clearly, when the consumption raises, the satisfaction increases. It is also intuitive that a raise in the consumption produces more satisfaction when the previous consumption was low, than if the consumption was very high.

On the other hand, the investment $I(t)$ produces some benefits given by $f(I)$. The inversion benefits will constitute the capital increments in the next time instant. That is $K'(t) = f(I(t))$.

Let $\alpha(t) \in [0,1]$ be the consumed fraction of the capital at time t. Then, $C = \alpha K$ and $I = (1-\alpha)K$. The promoter aims to maximize his utility function over the time interval $[t_0, t_1]$. Therefore, the objective is to maximize the following functional,

$$\int_{t_0}^{t_1} U(\alpha(t)K(t))\ dt.$$

The total capital evolves over time as

$$K'(t) = f(I(t)) = f\left([1-\alpha(t)]K(t)\right), \quad \textit{with} \quad K(0) = K_0.$$

The promoter has to choose a value of $\alpha = \alpha(t),\ t \in [t_0, t_1]$. The variable α is called the control variable of the system. The best selection for α is called the optimal control.

Optimal Control Theory is an extension of the Calculus of Variations (see [15]). The Calculus of Variations is a field that uses variations, which are small changes in functions, to find the extrema of a function. However, in the Calculus of Variations there is not a variable that controls the system.

As illustrated in the previous example, in an Optimal Control problem there exists a variable that determines the dynamics of the system. For the rest of the work, we denote by $u(t)$ the control variable at the time t.

In control problems, there is also a variable $x(t)$ that indicates the state of the system. For example, in an economic context, the state variable can be the capital of an enterprise, and the control can represent the inversion in publicity, which might increase future benefits. The objective is then to maximize the capital minus the inversion cost.

Apart from the economic context, Optimal Control problems are employed in several fields, such as social behaviour, engineering, biology or environment. In [9], a list of contributions to economic growth theory, and to environmental and resource economics is provided. The referenced articles mainly study extended versions of the Pontryagin maximum principle (studied in section 2.1.1) to obtain necessary conditions for optimality in control problems.

In this section, most important results in Optimal Control Theory are presented.

For the general case, the aim of a Control problem is to maximize a function $f(s, x(s), u(s))$ in a time interval $[t, T]$. Function f is assumed to be a benefit. However, if f represents a cost, the minimization problem is equivalent to the maximization of $-f$.

The functional to maximize is

$$\begin{aligned} J(t, x_0, u) = \int_t^T & f(s, x(s), u(s))\ ds\ + \Psi(x(T)), \\ \text{s.t.} \quad & \dot{x}(s) = g(s, x(s), u(s)), \\ & x(t) = x_0. \end{aligned} \tag{2.1}$$

where $\Psi(x(T))$ is a residual value that depends on the model interpretation.

For example, in an economic context, the residual can represent the expected value of an enterprise at time T.

It should be noted that the state variable evolution over time is expressed as a function that depends of the control.

Definition 2.2. *Let u^* be the control that maximizes the functional J. We define the value function as*

$$V(t,x_0) = J(t,x_0,u^*) = \max_u J(t,x(t),u).$$

Two of the main results of classical Optimal Control Theory are presented. The Pontryagin maximum principle provides necessary conditions for optimality that enable to find a unique solution or, at least, reduce the possible solutions to a few. Meanwhile, the Hamilton-Jacobi-Bellman equation provides a sufficient condition for the solution of the problem. The solution is obtained by backward induction, also called dynamic programming approach.

2.1.1 Pontryagin maximum principle

The objective of this section is to derive necessary conditions for the optimality of u in the maximization problem

$$\begin{aligned} &\max_{u(s)} \int_t^T f(s,x(s),u(s))ds + \Psi(x(T)), \\ &\text{s.t.} \quad \dot{x}(s) = g(s,x(s),u(s)), \\ &\qquad\;\; x(t) = x_0. \end{aligned} \tag{2.2}$$

Let $u^*(s)$ be the optimal control for (2.2), and $(u^*(s),x^*(s))$, $s \in [t,T]$ the control and corresponding optimal state trajectory that maximizes the value function.

In order to solve the optimization problem with restrictions on the state variable, we need to define an auxiliary function H (similar to the Lagrangian in static optimization). Function H is called the *Hamiltonian*, and sets the Lagrange multipliers as functions depending on time rather than constants.

The Hamiltonian of problem (2.2) is defined as

$$H(s,x(s),u(s),\lambda) = f(s,x(s),u(s)) + \lambda g(s,x(s),u(s)).$$

The Pontryagin maximum principle provides some necessary conditions for the optimality of $(u^*(t),x^*(t))$ when there are no constraints on the final state of the problem.

This result is presented for one control and one state variables. A generalization of the result is presented in [19].

First, some assumptions have to be made (see [19, Ch 2]).

- The set of admissible controls must be a compact set $\mathbb{U} \in \mathbb{R}^m$. The control $u(s)$ is allowed to take values at the boundary of $\mathbb{U}$. It is very usual to assume that the control variable takes values in $[0, 1]$, as it frequently represents a fraction of total investment or stock.
- The control variable $u(s)$ is assumed to be piecewise continuous.

Depending on the problem, a continuity assumption may be too restrictive. For example, in an economic context the optimal strategy can consist in investing all the capital only when $s \in [t, t']$.

The control function is then defined as

$$u = \begin{cases} 1 & s \in [t, t'], \\ 0 & s > t'. \end{cases}$$

Next theorem is known as Pontryagin maximum principle. Given the previous assumptions, the Pontryagin maximum principle provides necessary conditions for optimality.

Theorem 2.3 (**Pontryagin maximum principle**).
Let $u^*(s)$ be a piecewise continuous control defined on $[t, T]$, which solves (2.2), and let $x^*(s)$ be the associated optimal path. Then, there exists a continuous function $\lambda(s)$, piecewise continuously differentiable, such that $\forall s \in [t, T]$ the following conditions are satisfied:

- *$u^*(s)$ maximizes the Hamiltonian,*

$$f_u(s, x^*(s), u^*(s)) + \lambda(s) g_u(s, x^*(s), u^*(s)) = 0. \tag{2.3}$$

- *Except at the points of discontinuities of $u^*(s)$ it holds*

$$-\dot{\lambda}(s) = f_x(s, x^*(s), u^*(s)) + \lambda(s) g_x(s, x^*(s), u^*(s)). \tag{2.4}$$

- $-\lambda(T) + \Psi'(x^*(T)) = 0$ *(natural transversality condition).*

The optimal solution satisfies also the initial conditions

- $x^*(t) = x_0$.
- $\dot{x}^*(s) = g(s, x^*(s), u^*(s))$.

In general, to solve a control problem by the Pontryagin maximum principle, the Hamiltonian is derived with respect to u and x. Replacing in the conditions and solving the resulting differential equations, the number of possible optimal solutions is restricted to a few or just one.

Let us see an example of a problem resolution by the Pontryagin maximum principle.

Example 2.4. *A company owner wants to maximize the value of his enterprise (in millions of dollars) over the next 20 years. The objective is to find the optimal trajectories for the inversion $I = I(t)$ and the capital stock $K = K(t)$. That is, the aim is to solve the following problem*

$$\begin{aligned} \max_{I(t)} J(K) = \max_{I(t)} \int_0^{20} \left(2K(t) - I(t) - \frac{1}{2}I(t)^2\right) dt + K(20), \\ s.t. \quad \dot{K}(t) = I(t) - 0.2K(t), \quad K(0) = 25. \end{aligned} \tag{2.5}$$

In this problem the control variable is the inversion, and the state is the capital stock. The Hamiltonian of the problem is

$$H(t, K(t), I(t), \lambda) = 2K(t) - I(t) - \frac{1}{2}I(t)^2 + \lambda(I(t) - 0.2K(t)).$$

Applying the Pontryagin maximum principle, the following relations are obtained.

1. *By* (2.3) $H_I = -1 - I(t) + \lambda(t) = 0$, *so* $I(t) = \lambda(t) - 1$.
2. *By* (2.4) $H_K = 2 - 0.2\lambda(t) = -\dot{\lambda}(t)$.
3. *The transversality condition reads* $\lambda(20) = 1$.
4. *The initial conditions are satisfied* $\dot{K}(t) = I(t) - 0.2K(t)$, $K(0) = 25$.

Conditions 2 and 3 allow to find the value of $\lambda(t)$ solving the differential equation. Once the value of $\lambda(t)$ is obtained, it is immediate to find the optimal control with condition 1.

Finally, substituting 1 in 4 we obtain

$$\dot{K}(t) = \lambda(t) - 1 - 0.2K(t), \quad K(0) = 25.$$

Solving the previous differential equation, the optimal trajectory of $K(t)$ is obtained.

2.1.2 Hamilton-Jacobi-Bellman

The Hamilton-Jacobi-Bellman approach to control problems is based on the Dynamic Programming (DP) principle. According to this principle, given an optimal trajectory, any subtrajectory must also be optimal.

Therefore, the problem can be decomposed in an infinite number of subproblems, one for each time instant. The DP principle allows solving problems by backward induction, what might be numerically easier to address.

The following lemma shows that the optimal solution for a control problem must be optimal in any subinterval of time in $[t, T]$.

Lemma 2.5. *Let (u^*, x^*) be the optimal solution of problem (2.2). Let $\hat{u}$ be the optimal control for the following problem in $[t + \Delta t, T]$*

$$\begin{aligned} &\max_{u(s)} \int_{t+\Delta t}^{T} f(s, x(s), u(s))ds + \Psi(x(T)), \\ &s.t. \quad \dot{x}(s) = g(s, x(s), u(s)), \\ &\qquad\quad x(t + \Delta t) = x^*(t + \Delta t). \end{aligned} \tag{2.6}$$

It must hold that $\hat{u} = u^*$ (in an optimality sense) for all $s \in [t + \Delta t, T]$.

Proof. Suppose that $\hat{u}$ gives a better solution than u^*. Define a new strategy

$$v(s) = \begin{cases} u^*(s) & \text{if } s \in [t, t + \Delta t], \\ \hat{u}(s) & \text{if } s \in [t + \Delta t, T]. \end{cases}$$

Set also

$$x_v(s) = \begin{cases} x^*(s) & \text{if } s \in [t, t+\Delta t], \\ \hat{x}(s) & \text{if } s \in [t+\Delta t, T], \end{cases}$$
$$\dot{x}_v(s) = g(s, x(s), v(s)),$$
$$x_v(t) = x_0.$$

Therefore,

$$\begin{aligned} J(t, x_0, v) &= \int_t^T f(s, x_v, v)ds + \Psi(x_v(T)) \\ &= \int_t^{t+\Delta t} f(s, x_v, v)ds + \int_{t+\Delta t}^T f(s, x_v, v)ds + \Psi(x_v(T)) \\ &> \int_t^{t+\Delta t} f(s, x, u^*)ds + \int_{t+\Delta t}^T f(s, x, u^*)ds + \Psi(x^*(T)) \\ &= J(t, x_0, u^*). \end{aligned}$$

This leads to a contradiction since $J(t, x_0, u^*)$ was the optimal solution with initial conditions (t, x_0). Consequently $\hat{u} = u^*$ in an *optimality sense* $\forall s \in [t, T]$.

□

Lemma 2.6 (**Bellman's principle of optimality**).
An optimal policy has the property that, whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy. Following this principle, the problem (2.2) *can be rewritten as*

$$\begin{aligned} V(t, x_0) = \max_{u(s)} \int_t^{t+\Delta t} & f(s, x(s), u(s))ds + V(t+\Delta t, x(t+\Delta t)), \\ s.t. \quad & \dot{x}(s) = g(s, x(s), u(s)), \\ & x(t) = x_0. \end{aligned} \tag{2.7}$$

On the basis of the previous result, the next theorem provides some sufficient conditions for optimality.

Theorem 2.7 (**Hamilton-Jacobi-Bellman**).
If the value function V is continuously differentiable, then it satisfies

$$-V_t(t, x) = \max_u \{f(t, x, u) + V_x(t, x)g(t, x, u)\},$$

and the condition

$$V(T, x(T)) = \Psi(x(T)).$$

Proof. Let us consider the problem

$$V(t, x) = \max_{u(s)} \int_t^{t+\eta} f(s, x(s), u(s))ds + V(t+\eta, x(t+\eta))$$

. By the Bellman's principle of optimality 2.6, any subtrajectory of the optimal control must be optimal. This means that the previous value function does not depend on the value of η,

$$\frac{\partial V(t, x)}{\partial \eta} = 0.$$

Therefore, for all η we have

$$\begin{aligned} \frac{\partial V(t,x)}{\partial \eta} = \max_{u(t)} \Bigg\{ & f(t+\eta, x(t+\eta), u(t+\eta)) + \frac{\partial V}{\partial t}(t+\eta, x(t+\eta)) \\ & + \frac{\partial V}{\partial x}(t+\eta, x(t+\eta)) \frac{dx}{dt}(t+\eta) \Bigg\} = 0. \end{aligned}$$

Note that

$$\frac{dx}{dt}(t+\eta) = g(t+\eta, x(t+\eta), u(t+\eta)).$$

Taking the limit when $\eta \to 0$, the following equation is obtained

$$0 = \max_u \{ f(t, x, u) + V_x(t, x) g(t, x, u) + V_t(t, x) \}.$$

As the value function does not depend on u, it is concluded that

$$-V_t(t, x) = \max_u \{ f(t, x, u) + V_x(t, x) g(t, x, u) \}.$$

□

The control $\mathcal{U}^*$ that maximizes the Hamiltonian

$$\begin{aligned} \mathcal{U}^*(s, x(s), V_x) &= \arg\max_u \ H(t, x, u, V_x(t, x)) \\ &= \arg\max_u \ \{ f(t, x, u) + V_x(t, x) g(t, x, u) \}. \end{aligned}$$

is called the *feedback solution.* With the obtained feedback control, the optimal trajectory can be derived by solving

$$\begin{cases} \dot{x}^*(s) = g(s, x^*(s), \mathcal{U}^*(s, x^*(s)), \\ x^*(t) = x. \end{cases}$$

Finally, we get that the optimal control $u^*(s)$ satisfies

$$u^*(s) = u^*(s, x^*(s)) = \mathcal{U}^*(s, x^*(s), V_x).$$

2.1.3 Infinite-Horizon problems

In the previous sections finite-time horizon problems were considered. A residual value $\Psi(T)$ was included to represent the reward at the end of the time interval.

In some applications, such as economic ones, problems with an infinite-horizon time arise. In infinite-horizon problems, the payoff does not include a terminal reward. Instead, a running cost is included in the optimization problem. The payoff to maximize is then

$$J(u, x_0) = \int_0^\infty f(s, x(s), u(s))e^{-\rho t}\, dt,$$

where $\rho \geq 0$ is a discount rate.

The term $e^{-\rho t}$ is called the discount factor, and it depends on the running time t.

It should be remarked that in infinite-horizon problems, specifications where function f does not depend explicitly on time t, i.e. autonomous problems, are very much employed, since they have a very natural interpretation, specially in the field of Economy.

Theorems 2.3 and 2.7 where stated under a finite time assumption. For an infinite time horizon, some changes have to be made.

For the Pontryagin maximum principle, the transversality condition $\lambda(T) = \Psi'(x^*(T))$ turns to the following condition

$$\lim_{t\to+\infty} e^{-\rho t}\lambda(t) = 0.$$

For the Hamilton-Jacobi-Bellman, the transversality condition

$$\lim_{t\to\infty} e^{-\rho t} V(x^*(t)) = 0,$$

is added to ensure convergence.

The problems considered in this work will be infinite time horizon problems.

2.2 Game Theory

In Optimal Control Theory, there is only one agent determining the control or controls with the objective of maximizing a payoff. However, if the aim to study is the relations among several agents, the Optimal Control Theory must be extended to the field referred as Game Theory.

Game theory can be seen as a generalization of an optimal control problem involving multiple players and multiple payoffs. There are other extension possibilities, such as Multi-Objective optimization problems (1 player, multiple payoffs) and Team Theory (multiple players, 1 payoff).

Formally, Game Theory is the study of strategic interactions, which might evolve over time, among rational agents. In this section, basic concepts in Game Theory are presented.

2.2.1 Game classification

Games can be classified following several criteria, such as players interaction, available information, evolution over time... The most important features employed to classify games are:

- **Discrete / Continuous games**: In continuous games, each player has an infinite number of available actions, whereas the set of possible actions in a discrete game is finite.
- **Cooperative / Non cooperative**: In a non cooperative game, every player chooses an action without any previous agreement with the other players. In a cooperative game, players are able to form alliances to improve their benefits.
- **Zero-sum / Non-zero-sum**: In zero sum games, the total benefit remains constant. This means that one player's gain implies another player's loss. In non-zero sum games, the previous condition is not forced, so all players can have benefits or losses at the same time.
- **Simultaneous / Sequential (or dynamic)**: In simultaneous or static games, all players act at the same time. The game is played

in one step and there is no need of a state variable. Simultaneous games are usually represented in *normal form*, that is, by a matrix containing the players, strategies, and payoffs.

Sequential or dynamic games are played in multiple steps, and there is an established order of action. Later players have information (perfect or imperfect, see below) about the actions of previous players. A state variable summarizes the information in each step. The sequential character of these games allows a representation in *extensive* or *tree form*.

- **Perfect, imperfect and complete information**: In a game of perfect information, all players know the previous movements of the other players. A game that is not perfect is called imperfect.

  On the other side, complete information requires that every player knows the strategies and the payoffs available to other players, but not necessarily the actions taken. A game that is not complete is called incomplete.

2.2.2 Ingredients of a game

A game in normal form can be represented as a tuple $\{J; \mathbb{U}_1, \mathbb{U}_2, ..., \mathbb{U}_n; \mathbf{V}\}$ where

- The set $J = \{1, 2, ..., n\}$ is the set of players.
- The set $\mathbb{U}_i$ is the set of actions or strategies of the player i, $i \in J$.
- A n-uple $\mathbf{u} = (u_1, u_2, ..., u_n)$ where each $u_i \in \mathbb{U}_i$ is an action profile. The expression $u_{-i} = (u_j)_{j \in J, j \neq i}$ denotes the action profile of all players except i, and $(u_1, ..., u_n) = (u_i, u_{-i})$.
- The function $\mathbf{V} : \mathbb{U}_1 \times \mathbb{U}_2 \times ... \times \mathbb{U}_n \longmapsto \mathbb{V} \subseteq \mathbb{R}^n$ is the payoff function. We can write $\mathbf{V} = (V_1, V_2, ..., V_n)$, V_i being the payoff of player i. The payoff is assumed to be a profit. Therefore, players aim to maximize its value.

A game in extensive form includes the previous elements, plus a state variable $\mathbb{X}$ and a family of information sets $\mathbf{H}$ (see [17, Ch 1]). A collection of information sets H_i is available for each player. The game can be represented as a tuple $\{J; \mathbb{U}_1, \mathbb{U}_2, ..., \mathbb{U}_n; \mathbf{V}, \mathbb{X}, \mathbf{H}\}$ where

- $\mathbb{X}$ is the set of game nodes. Each node represents a possible situation of the game. For discrete space games, a sequential game begins at a starting node O (origin) and finishes at the terminal nodes. A node is called terminal if it does not have any subsequent nodes.

- $X_i \subseteq \mathbb{X}$ is the set of nodes where player i makes a decision, and $\mathbb{U}_i(x)$ the set of possible actions at node $x \in X_i$. The function $\mathbf{u}_i(x) \in \mathbb{U}_i(x)$

$$\begin{aligned} \mathbf{u}_i(x) :\ & X_i \longrightarrow \mathbb{X} \\ & x \longmapsto \mathbf{u}_i(x) = x', \end{aligned}$$

  denotes the action for player i that takes the game from node x to a following node x'. Actions that start in the same node and lead to different nodes must be different.

- The function $\mathbf{V}(x)$ is the payoff function at node x.

- $\mathbf{H}$ is the family of information sets. $\mathbb{U}(H_x)$ is the set of actions available at node x for an information set $H \in \mathbf{H}$.

2.2.3 Differential games

Differential games (see [2, Ch 9]) are characterized by a state variable (of one or more dimensions) whose evolution is determined by a differential equation

$$\dot{x} = g(x, u_1, ..., u_N), \qquad x(t_0) = x_0,$$

where x_0 is the initial state value.

Differential games can also be seen as a generalization of Optimal Control problems. In an optimal control problem there is a single control $u(t)$ and a single criterion to be optimized, while differential game theory generalizes this to multiple players, their controls $u_1, u_2, ..., u_n$, and multiple optimization problems. Consequently, each player aims to control the state of the system in order to maximize his payoff.

2.2.4 Equilibrium points

When considering a game, the aim is to find an equilibrium solution where all players maximize their payoffs. The common way to define the solution for non-cooperative games is the Nash equilibrium.

Definition 2.1. *An action profile* $(u_1^*, u_2^*, ..., u_n^*)$ *is a* Nash equilibrium *if*

$$V_i(u_i^*, u_{-i}^*) \geq V_i(u_i, u_{-i}^*) \quad \forall u_i \in \mathbb{U}_i, \ \forall i \in J.$$

where recall that u_{-i} *denotes the strategies of all players but player* i.

In other words, a Nash equilibrium is an action profile such that no player can get a better payoff by deviating from it, assuming that the other players do not change their actions.

Definition 2.2. *The* best-response set *for player* i *is the set*

$$B_i(u_{-i}) = \{u_i^* \in \mathbb{U}_i \mid V_i(u_i^*, u_{-i}) = \max_{u_i \in \mathbb{U}_i} V_i(u_i, u_{-i})\}.$$

In a Nash equilibrium, all players actions are the best responses to the other players actions, that is $u_i^* \in B_i(u_{-i})$ for all $i \in J$.

Existence of a Nash equilibrium
Using the Kakutani's (fixed point) theorem, Nash proved the existence of a Nash equilibrium for non-zero sum games with a finite number of players. The existence is proved under the following conditions:

- *Each player set of actions is a compact and convex subset of* $\mathbb{R}^n$.
- *The payoff functions are continuous and concave.*

The $\varepsilon-$Nash equilibrium is another commonly used solution for non-cooperative games, weaker than the Nash equilibrium.

Definition 2.3. *An action profile* $(u_1^*, u_2^*, ..., u_n^*)$ *is an* ε - Nash equilibrium *if, for a given* $\varepsilon > 0$,

$$V_i(u_i^*, u_{-i}^*) \geq V_i(u_i, u_{-i}^*) - \varepsilon \quad \forall u_i \in \mathbb{U}_i, \ \forall i \in J.$$

The ε-Nash equilibrium equals the Nash equilibrium for $\varepsilon = 0$.

In Chapter 4, a time discretization of a non-cooperative infinite time horizon differential game is proposed. In [7], it is proved that, under certain regularity hypothesis, the discrete-time Nash game equilibrium is an $\epsilon-$Nash equilibrium of the continuous game. This result is crucial to ensure that a time-discretization can be applied to find an approximation to the continuous time solution, with an error which depends on the magnitude of the time discretization step (see [7]).

When considering sequential games, in which there is an explicit order of events, subgame perfect Nash equilibrium solutions arise. A subgame is defined as a subset of game nodes that still forms a game.

Definition 2.4. *Let* $\mathbf{u}^* = (u_1^*, u_2^*, ..., u_n^*)$ *a Nash equilibrium action profile. We say that* $\mathbf{u}^*$ *is a* subgame perfect Nash equilibrium *of a game if the restriction of* $\mathbf{u}^*$ *to any subgame constitutes a Nash equilibrium of the subgame.*

Subgame perfectness is a property of prime importance in applications. This is due to the fact that, in dynamic games with perfect and complete information, backward induction can be employed to compute subgame perfect Nash equilibrium solutions.

Backward induction, also called dynamic programming, is an algorithm that consists in studying subgames from the terminal state to the initial state of the game.

In the process of backward induction, an optimal strategy is identified for each decision node. Consequently, the obtained action profile is a subgame perfect Nash equilibrium.

Strategies in game theory can be classified in *open-loop* and *closed-loop* strategies (see [1]).

Definition 2.5. *Assuming that the admissible control sets* $\mathbb{U}_j$ *are not state dependent, an open-loop strategy selects a control action depending only on time* s *for a fixed initial state* x_0,

$$u_j^* = \gamma_j(s, x_0), \quad \textit{where} \quad \gamma_j : \mathbb{R} \times \mathbb{R}^n \to \mathbb{U}_j.$$

Definition 2.6. *A closed-loop strategy or state-feedback strategy selects a control depending on time s and on the state at time s*

$$u_j^* = \mathcal{U}_j^*(s, x^*(s)), \quad \textit{where} \quad \mathcal{U}_j^* : \mathbb{R} \times \mathbb{R}^n \to \mathbb{U}_j(x,t).$$

The function $\mathcal{U}_j^*(s, x^*(s))$ is called the feedback function.

A state-feedback strategy is also called *Markovian*, in contrast to open-loop. One can look for an equilibrium point in open-loop strategies, or in closed-loop strategies. The main drawback of an open-loop Nash equilibria is that it is not subgame perfect, in opposition to feedback strategies.

In the present work, the aim is to obtain Markovian Nash equilibria through the dynamic programming approach. Nash equilibria are studied in non-cooperative scenarios. However, other equilibrium concepts for the cooperative case can also be analysed.

Cooperation among players offers the possibility to improve the value of the non-cooperative payoff. We introduce the concept of Pareto optimality.

Definition 2.7. *An action profile $\mathbf{u}^P = (u_1^P, u_2^P, ..., u_n^P)$ is said to be Pareto optimal if there exists no other profile $\mathbf{u} = (u_1, u_2, ..., u_n)$ such that for $i = 1, ..., n$*

$$V_i(\mathbf{u}) > V_i(\mathbf{u}^P) \quad \textit{and} \quad V_{-i}(\mathbf{u}) \geq V_{-i}(\mathbf{u}^P).$$

In other words, given a Pareto optimal solution, it is not possible to strictly increase the payoff of one player without strictly decreasing the payoff of other player.

Noncooperative strategies are not, in general, Pareto optimal. For this reason, players might get organized under a unique decision maker, whose objective is to maximize the weighted sum of players payoffs (see [20]).

For the payoff weights vector $\alpha = (\alpha_1, ..., \alpha_n)$, $\sum_{j=1}^n \alpha_j = 1$ and $\alpha_j > 0$, the value function of the cooperative game is

$$W^\alpha(t, x_0) = \max_{u_1(s), u_2(s) ... u_n(s)} \sum_{j=1}^n \alpha_j \left(\int_t^T f_j(s, x(s), u_j(s), u_{-j}(s)) ds \right) + \Psi(x(T)),$$

subject to the dynamics

$$\dot{x}(s) = g(s, x(s), u(s)), \quad x(t) = x_0.$$

In order to ensure that the Pareto optimal solution holds, the weight coefficients α must satisfy

$$\alpha_j W^{\alpha}(t, x_0) \geq V_j(t, x_0), \quad j = 1, ..., n,$$

where W^{α} is the Pareto optimal value function and V_j is the non-cooperative payoff for player j.

Chapter 3

Spectral methods

Prior to the description of the techniques employed in the numerical resolution of Optimal Control and Game Theory problems, a review of the interpolation methods and algorithms that will be employed in order to discretize the spatial variable is presented.

Spectral methods (see [3]) are a collection of techniques that can be employed to solve differential equations. Most of those methods are based only on a spatial discretization and they employ a suitable family of trial basis functions. The approximate solution is represented as a linear combination of trial functions (also called expansion or approximating functions, that are combination of the basis functions) weighted by test (or weight) functions.

Let us show how spectral methods are employed. Consider a differential equation

$$\frac{\partial u}{\partial t} = \mathcal{L}(u) \tag{3.1}$$

with an initial condition $u(\mathbf{x}, 0)$ and suitable boundary conditions. $\mathcal{L}(u)$ is an operator (linear or nonlinear) containing all the spatial derivatives of u (for the moment, we assume only a spatial dimension for simplicity).

Let us consider the expansion of the solution of (3.1), $u(x,t)$, on N trial functions $\phi_n(x)$, with coefficients $\hat{c}_n(t)$ $n = 0, ..., N-1$

$$u(x,t) \simeq \sum_{n=0}^{N-1} c_n(t)\phi_n(x) = u_N(x,t).$$

In general, the residual

$$\mathcal{R}_N(x,t) = \frac{\partial u_N}{\partial t} - \mathcal{L}(u_N)$$

will not vanish everywhere.

The approximation coefficients $\hat{c}_n(t)$ are obtained by selecting a set of test functions w_k and requiring

$$\int_a^b \mathcal{R}_N(x,t) w_n(x)\ dx = 0,$$

where $[a,b]$ is the spatial domain.

The choice of the trial functions depends on the characteristics of the problem. If the problem considered is periodic, the Fourier basis functions are commonly used. Chebyshev or Legendre polynomials are useful for bounded domains.

For unbounded domains (see section 2.6 of [3]) usual choices are Hermite polynomials in $(-\infty,\infty)$ and the Laguerre polynomials in $[0,\infty)$. Also, an expansion in a set of Jacobi polynomials can be applied when mapping an unbounded domain into a bounded one, or when truncating an unbounded domain.

Depending on the choice of test functions, some relevant spectral approaches are the Galerkin, the collocation and the tau methods. For the Galerking and tau schemes, the test functions are the same as the trial ones. The weigth functions must satisfy the boundary conditions in the case of Galerking methods. For tau methods, the previous condition is not necessarily true, but a supplementary set of equations is set to apply the boundary conditions.

In the collocation method, the weights are the Dirac delta functions centered at certain collocation points. The differential equation must be satisfied exactly at the collocation points. The collocation points for both the differential equations and the boundary conditions are usually the same as the physical grid points. The most effective choice for the grid points are those that correspond to quadrature formulas of maximum precision.

In the present work, a Chebyshev collocation approach is proposed. In the next section, definition and basic properties of the Chebyshev polynomials are presented.

3.1 Chebyshev polynomials

Definition 3.1. *Let*

$$T_n(x) = \cos(n \arccos(x)), \tag{3.2}$$

where n is a nonnegative integer and $x \in [-1, 1]$.

It is known (see [18]) that this function is a polynomial, referred as the Chebyshev polynomial of degree n

Proposition 3.2. *Chebyshev polynomials satisfy the following recursive property*

$$T_{n+2}(x) = 2xT_{n+1}(x) - T_n(x) \tag{3.3}$$

Proof. Consider the trigonometrical properties

$$\begin{aligned} \cos((n+2)\theta) &= \cos((n+1)\theta)\cos(\theta) - \sin((n+1)\theta)\sin(\theta) \\ \cos(n\theta) &= \cos((n+1)\theta)\cos(\theta) + \sin((n+1)\theta)\sin(\theta). \end{aligned} \tag{3.4}$$

The addition of the previous expressions gives

$$\cos((n+2)\theta) = 2\cos((n+1)\theta)\cos(\theta) - \cos(n\theta)$$

Taking $x = \cos(\theta)$ and employing the fact that that $T_1(x) = x$, gives the following expression

$$T_{n+2}(x) = 2xT_{n+1}(x) - T_n(x) \tag{3.5}$$

□

The previous recursive expression shows that (3.2) is indeed a polynomial of degree n. The first Chebyshev polynomials are, for $x \in [-1, 1]$

$$\begin{aligned} &T_0(x) = 1, &&T_1(x) = x, \\ &T_2(x) = 2x^2 - 1, &&T_3(x) = 4x^3 - 3x. \end{aligned} \tag{3.6}$$

Definition 3.3. *Let $N \in \mathbb{N}$. The $N+1$ Chebyshev nodes in $[-1, 1]$ correspond to the extrema of $T_n(x)$. The points $x \in [-1, 1]$ at which $|T_n(x)| = 1$ are called the extrema, and are given by*

$$\alpha^k = \cos\left(\frac{\pi k}{N}\right), \quad k = 0, ..., N.$$

Any point in $x \in [-1,1]$ can be expressed in an interval $[a,b]$ via the following affine transformation:

$$\tilde{x} = \frac{b-a}{2}x + \frac{b+a}{2}.$$

Therefore, the $N+1$ Chebyshev nodes in $[a,b]$ are defined by

$$\tilde{\alpha}_k = \frac{1}{2}\left((b+a) + (b-a)\cos\left(\frac{\pi k}{N}\right)\right), \quad k = 0, ..., N.$$

Some properties of Chebyshev polynomials are

- $|T_k(x)| \leq 1$, $-1 \leq x \leq 1$.
- The Chebyshev polynomials $T_n(x)$ form a complete orthogonal set on the interval $[-1,1]$ with a weight function $\frac{1}{\sqrt{1-x^2}}$.

$$\int_{-1}^{1} T_m(x)T_n(x)\frac{dx}{\sqrt{1-x^2}} = 0, \quad m \neq n$$

and for $m = n$

$$\int_{-1}^{1} T_n^2(x)\frac{dx}{\sqrt{1-x^2}} = \omega_n \frac{\pi}{2}, \tag{3.7}$$

where the previous coefficients ω_n are given by

$$\omega_n = \begin{cases} 2 & n = 0 \\ 1 & n \geq 1 \end{cases}$$

In this section, we exposed the basic Chebyshev polynomials properties that are needed to compute a Chebyshev interpolation scheme. For further results on Chebyshev polynomials see [18].

3.2 Chebyshev collocation approach

To solve the dynamics of a differential game, a good choice might be a collocation scheme based in Chebyshev polynomials. This choice is due to the fact that Chebyshev polynomials are a versatile set of trial functions, that supply very precise and numerically cheap approximations.

Let $\tilde{u}$ be a continuous function defined in $[a, b]$, and $\{\tilde{\alpha}_j\}_{j=0}^N$, the corresponding $N+1$ Chebyshev nodes in $[a, b]$.

Since it is easier to work in $[-1, 1]$, the corresponding function u in $[-1, 1]$ is defined as

$$u(x) = \tilde{u}(\tilde{x}), \quad \text{where} \quad \tilde{x} = \frac{b-a}{2}x + \frac{b+a}{2}, \quad x \in [-1, 1].$$

The interpolating polynomial, denoted by $I_N u$, is given by

$$I_N u = \sum_{n=0}^{N} \hat{c}_n T_n,$$

where T_n is the Chebyshev polynomial of degree n.

At the collocation points, the Chebyshev nodes α_j, the polynomial satisfies

$$u(\alpha_j) = I_N u(\alpha_j) = \sum_{n=0}^{N} \hat{c}_n T_n(\alpha_j), \quad j = 0, ..., N.$$

The $\hat{c}_n$ terms are called the *expansion coefficients* and they are given by the inverse relationship

$$\hat{c}_n = \frac{1}{\gamma_k} \sum_{j=0}^{N} u(x_j) T_n(x_j) w_j, \quad n = 0, ..., N, \tag{3.8}$$

where

$$\gamma_n = \sum_{j=0}^{N} T_n^2(x_j) w_j. \tag{3.9}$$

The test or weight functions are obtained by the Gauss-Lobato integration formula in $[-1, 1]$.

Gauss-Lobatto integration
Consider the polynomial

$$q(x) = p_{N+1}(x) + ap_N(x) + bp_{N-1}(x),$$

where a and b are chosen so that $q(-1) = q(1) = 0$.

Let $-1 = x_0 < x_1 < ... < x_N = 1$ be the $N+1$ roots of the polynomial $q(x)$, and let $\omega_0, ..., \omega_N$ be the solution of the linear system

$$\sum_{j=0}^{N} (x_j)^k \omega_j = \int_{-1}^{1} x^k \omega(x) dx, \qquad 0 \leq k \leq N.$$

Then the following equality holds

$$\sum_{j=0}^{N} p(x_j)\omega_j = \int_{-1}^{1} p(x)\omega(x) dx$$

for all $p \in \mathbb{P}_{2N-1}$.

With the Gauss-Lobato integration formula, the following collocation points and weights are obtained

$$x_j = \cos\left(\frac{\pi j}{N}\right), \qquad w_j = \begin{cases} \dfrac{\pi}{2N} & j = 0, N \\ \dfrac{\pi}{N} & j = 1, ..., N-1. \end{cases}$$

Note that the defined Chebyshev quadrature points are ordered from right to left. This breaks the general convention in which collocation points are ordered from left to right.

For the previous nodes x_j, the following must be satisfied

$$\begin{aligned} u(x_j) &= \sum_{n=0}^{N} \hat{c}_n T_n(x_j) \\ &= \sum_{n=0}^{N} \hat{c}_n \cos(n\ arcos(x_j)) = \sum_{n=0}^{N} \hat{c}_n \cos\left(n\left(\frac{\pi j}{N}\right)\right). \end{aligned}$$

Also, from (3.8) we get that

$$\hat{c}_n = \sum_{j=0}^{N} \frac{2}{N\omega_j\omega_n} \cos\left(\frac{\pi j n}{N}\right) u(x_j), \quad n = 0, ..., N,$$

where

$$\omega_l = \begin{cases} 2 & l = 0, N, \\ 1 & l = 1, ..., N-1. \end{cases}$$

We used the fact that combining (3.9) with the Chebyshev polynomials property (3.7) and the Gauss-Lobato integration formula, the normalization factors are given by

$$\gamma_j = \begin{cases} \pi & j = 0, N, \\ \frac{\pi}{2} & j = 1, ..., N-1. \end{cases}$$

This particular choice for the collocation points enables to use the *fast fourier transform* (FFT) algorithm in the evaluation of the function u.

The employed algorithm to determinate the expansion coefficients is the suggested in [3], as proposed in [5].

Algorithm 1:

1. Construct

   $z = [u(\alpha_0), u(\alpha_1), ..., u(\alpha_{N-1}), u(\alpha_N), ..., u(\alpha_1)]^T.$

2. Compute

   $$y = \frac{real(FFT(z))}{2N}.$$

3.

   $$\begin{cases} \hat{c}_0 = y(1) \\ \hat{c}_n = y(n+1) + y(2N - (n-1)) & 0 < n < N \\ \hat{c}_N = y(N) \end{cases}$$

3.3 Chebyshev multidimensional interpolation

Let $\{\alpha_k^j\}_{k=0}^{N_j}$ the $N_j + 1$ Chebyshev nodes for dimension j, $j = 1, ..., d$. The notation $\alpha_{\mathbf{n}} = (\alpha_{n_1}^1, \alpha_{n_2}^2, ..., \alpha_{n_d}^d)$, $0 \le n_j \le N_j$, refers to the d-dimensional Chebyshev nodes in $[-1, 1]^d$.

In the same way, $\tilde{\alpha}_{\mathbf{n}} = (\tilde{\alpha}_{n_1}^1, \tilde{\alpha}_{n_2}^2, ..., \tilde{\alpha}_{n_d}^d)$ are the Chebyshev nodes in an interval $\Pi = [a_1, b_1] \times [a_2, b_2] \times ... \times [a_d, b_d]$.

Let $\tilde{\mathbf{x}} = (\tilde{x}_1, ..., \tilde{x}_d)$, and $\tilde{\mathbf{u}}(\tilde{\mathbf{x}})$ be a continuous function defined in Π. The corresponding function $\mathbf{u}(\mathbf{x})$, $\mathbf{x} = (x_1, ..., x_d)$ in $[-1, 1]^d$ is defined as

$$\mathbf{u}(\mathbf{x}) = \tilde{\mathbf{u}}(\tilde{\mathbf{x}}), \quad \text{where} \quad \tilde{x}_j = \frac{b_j - a_j}{2} x_j + \frac{b_j + a_j}{2}, \quad j = 1, ..., d. \tag{3.10}$$

Let $\mathbf{N} = (N_1, ..., N_d)$. The $\mathbf{N}$-degree interpolation polynomial $I_{\mathbf{N}}\mathbf{u}$ is a d-dimensional polynomial that satisfies, at the Chebyshev nodes,

$$I_{\mathbf{N}}\mathbf{u}(\alpha_{\mathbf{n}}) = \mathbf{u}(\alpha_{\mathbf{n}}) = \tilde{\mathbf{u}}(\tilde{\alpha}_{\mathbf{n}}).$$

$I_{\mathbf{N}}\mathbf{u}$ is given by

$$\begin{aligned} I_{\mathbf{N}}\mathbf{u}(\mathbf{x}) &= \sum_{n_1, n_2, ..., n_d\ =0}^{N_1, N_2, ..., N_d} \hat{c}_{(n_1, n_2, ..., n_d)} T_{(n_1, n_2, ..., n_d)}(\mathbf{x}) \\ &= \sum_{n_1, n_2, ..., n_d\ =0}^{N_1, N_2, ..., N_d} \hat{c}_{(n_1, n_2, ..., n_d)} T_{n_1}(x_1) T_{n_2}(x_2) ... T_{n_d}(x_d), \end{aligned} \tag{3.11}$$

where the term $\hat{c}_{(n_1, n_2, ..., n_d)}$ is an expansion coefficient in $\mathbb{R}$, and $x_j \in [-1, 1]$ for $j = 1, ..., d$.

We use the following algorithm to determinate the coefficients $\hat{c}_{(n_1, n_2, ..., n_d)}$, proposed in [5].

First we define

- An array $\mathit{\Gamma}$ of dimension $(N_1 + 1) \times (N_2 + 1) \times ... \times (N_d + 1)$, containing the evaluation of u at the Chebyshev nodes

$$\mathit{\Gamma}(n_1 + 1, n_2 + 1, ..., n_d + 1) = \mathbf{u}(\alpha_{n_1}^1, \alpha_{n_2}^2, ..., \alpha_{n_d}^d).$$

- A dimension permutation operator $\mathcal{P}$. Let B an array of dimension $b \times a_1 \times ... \times a_m$. The array D such that $D = \mathcal{P}(B)$ has a size $a_1 \times ... \times a_m \times b$ and for any $0 \leq j_i \leq a_i$, $i = 1, ..., m$,

  $$D(j_1, ..., j_m, :) = B(:, j_1, ..., j_m). \tag{3.12}$$

  where we denote the vector $B(:, j_1, ..., j_m) = \{B(j, j_1, ..., j_m)\}_{j=1}^{b}$, and similar $D(j_1, ..., j_m, :) = \{D(j_1, ..., j_m, j)\}_{j=1}^{b}$.

The algorithm to obtain the coefficients $\hat{c}_{\mathbf{n}}$ of the interpolant is:

Algorithm 2:

```
1. $B_1 = \Gamma$.
2. For $i = 1 : d$
    • $\{m_1, m_2, ..., m_d\} = dim(B_i)$.
    • For $j_2 = 1 : m_2$, for $j_3 = 1 : m_3$, ..., for $j_d = 1 : m_d$
      $C_i(:, j_2, j_3, ..., j_d) = Algorithm1(B_i(:, j_2, j_3, ..., j_d))$.
    • $B_{i+1} = \mathcal{P}(C_i)$.
3. $\hat{c}_{(n_1, n_2, ..., n_d)} = B_{d+1}(n_1 + 1, n_2 + 1, ..., n_d + 1)$.
```

3.4 Chebyshev tensorial evaluation

In the previous section, an algorithm to obtain the coefficients of the interpolating polynomial was described. Suppose now that we need to evaluate this polynomial at certain values for each dimension. The number of query points in dimension j is denoted q_j. Let Θ be the set containing this points

$$\Theta = \{\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_d) \;/\; \tilde{x}_j \in \{\tilde{x}_j^1, ..., \tilde{x}_j^{q_j}\},\ \tilde{x}_j^k \in [a_j, b_j],\ 1 \leq k \leq q_j\}, \tag{3.13}$$

where $|\Theta| = q_1 q_2 ... q_d$.

The approximation of the values at the previous set of points is carried out with the interpolant $I_{\mathbf{N}}\mathbf{u}(\mathbf{x})$ in $[-1, 1]$, where the relation between $\mathbf{x}$ and $\tilde{\mathbf{x}}$ is given in (3.10).

The interpolation can be efficiently computed in a set of points like Θ by tensorial evaluation. First the *tensorial array operation* has to be defined, making use of the permutation operator defined in (3.12).

Definition 3.1. *Let A and B be two arrays of dimensions $(A)_{a \times a_1 \times a_2 \times ... \times a_m}$ and $(B)_{a \times b}$. We define the tensorial array operator $C = A \otimes B$ as the array $(C)_{a_1 \times a_2 \times ... \times a_k \times b}$ given by*

$$C(j_1, ..., j_m, :) = \mathcal{P}(B^t A(:, j_1, ..., j_m)),$$

where $B^t A(:, j_1, ..., j_m)$ is the usual product of matrix times a vector.

To implement the tensorial evaluation in Matlab, it is useful to employ function *multiprod*, that makes the tensorial operation in all variables at the same time in a very efficient way. It was implemented by Paolo de Leva (see [11]).

It should be mentioned that if multiprod is employed, it has to be imposed that $q_j > 1, j = 1, ..., m$. Multiprod does not recognise arrays of size $q_1 \times ... \times q_{i-1} \times 1 \times q_{i+1} \times ... \times q_m$, and collapses to an array $q_1 \times ... \times q_{i-1} \times q_{i+1} \times ... \times q_m$, something which may cause technical difficulties in the implementation due to the permute operator. It should also be mentioned that the latest versions of Matlab have the *pagemtimes* function, which performs a similar operation as *multiprod*.

Let

$$C = permute(multiprod(B'A), [2 : m\ 1]),$$

where *permute* is a command that rearranges array dimensions.

Let us see the algorithm presented in [5] to evaluate Chebyshev interpolation in set Θ ((3.13)).

The algorithm has two steps.

Algorithm 3:

Step 1: One dimensional Chebyshev evaluation

First, evaluate the Chebyshev polynomials in each dimension nodes.

For a fixed dimension $j = 1$, we need to evaluate the q_j nodes $\{\tilde{x}_j^k\}_{k=1}^{q_j}$. We denote by $\mu_j = \{x_j^k\}_{k=1}^{q_j}$ the previous nodes after the change of variables (3.10).

Making use of the recurrence relation (see (3.3))

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_n(x) = 2xT_{n-1}(x) - T_{n-2},\ n = 2, 3, ...,$$

we compute

$$\begin{aligned}
\mathbf{T}(\mu_1) &= \mathbf{T}(\mu_1)_{(N_1+1)\times q_1} = \left(T_n(x_1^k)\right)_{0\le n\le N_1,\ 1\le k\le q_1}, \\
\mathbf{T}(\mu_2) &= \mathbf{T}(\mu_2)_{(N_2+1)\times q_2} = \left(T_n(x_2^k)\right)_{0\le n\le N_2,\ 1\le k\le q_2}, \\
&\vdots \\
\mathbf{T}(\mu_d) &= \mathbf{T}(\mu_d)_{(N_d+1)\times q_d} = \left(T_n(x_d^k)\right)_{0\le n\le N_d,\ 1\le k\le q_d},
\end{aligned}$$

where $N_j + 1$ is the number of Chebyshev nodes in dimension j.

Each evaluation is stored in a two dimensional array.

Step 2: Multidimensional tensorial evaluation

The second step is to compute the tensorial product of the evaluation arrays obtained in Step 1.

This tensorial product provides the evaluation of $I_{\mathbf{N}}\mathbf{u}(\mathbf{x})$ at the whole set of points Θ.

Suppose that $(C)_{(N_1+1)\times(N_2+1)\times...\times(N_d+1)}$ is the array that contains the interpolation polynomial coefficients,i.e.,

$$C(n_1+1, n_2+1, ..., n_d+1) = \hat{c}_{(n_1,n_2,...,n_d)}.$$

After the following tensorial evaluation

$$I_{\mathbf{N}}\mathbf{u}(\Theta) = (...[(A \otimes \mathbf{T}(\mu_1)) \otimes \mathbf{T}(\mu_2)]...) \otimes \mathbf{T}(\mu_d),$$

we obtain an array of dimension $q_1 \times q_2 \times ... \times q_d$ containing the evaluation at the entire set Θ.

3.5 Differentiation

The derivative of a regular function u can be approximated from the Chebyshev expansion. This derivative is given by (see [3]),

$$u' = \sum_{n=0}^{\infty} \hat{d}_n T_n,$$

where

$$\hat{d}_n = \frac{2}{\omega_n} + \sum_{p=n+1,\ p+kodd} p\ \hat{c}_p, \quad k \geq 0,$$

$$\omega_k = \begin{cases} 2 & k = 0 \\ 1 & k \geq 1 \end{cases}$$

The previous expression is a consequence of the relation

$$2T_n(x) = \frac{1}{n+1} T'_{n+1}(x) - \frac{1}{n-1} T'_{n-1}(x).$$

However, the previous relation provides a more efficient way of computing the derivative.

Algorithm 4:

We know that for $n \geq N$, the coefficient $\hat{d}_n = 0$.

The coefficients of the derivative can be computed in decreasing order by the recursion relation

$$\omega_n \hat{d}_n = \hat{d}_{n+2} + 2(n+1)\hat{c}_{n+1}, \qquad 0 \leq n \leq N-1.$$

Chapter 4

Numerical methods in Game Theory

In this Section, the objective is to compute the Nash equilibrium of an infinite time horizon and non cooperative differential game with N players, where each of the players aims to maximize a benefit functional.

There are two very well known techniques in the numerical solution of Optimal Control Theory problems, which are commonly referred as *Value iteration* and *Policy iteration*. These techniques, which are also employed in Game Theory problems, are described in sections 4.1 and 4.2.

It is also described how a Chebyshev polynomial interpolation method can be combined with *Value and Policy iteration* techniques in order to obtain competitive algorithms which can provide numerical solutions of Game Theory problems.

Let us suppose that there are N players and that $\mathbb{X} \subset \mathbb{R}^N$ is the set of all possible states of the system. A particular state $\mathbf{x}$ is represented by a vector $\mathbf{x} = (x_1, x_2, ..., x_N) \in \mathbb{X}$.

Let $U_i \subset \mathbb{R},\ i = 1, ...N$ be the set of admissible actions of player i and let $\mathbb{U} = U_1 \times ... \times U_N$. The control variables of the players are represented by a vector $\mathbf{u} = (u_1, u_2, ..., u_N) \in \mathbb{U}$.

The objective of player $i = 1, ..., N$ is to maximize his payoff, that is, player i aims to maximize the value of

$$V_i(u_i, u_{-i}, \mathbf{x}_0) = \int_0^\infty f_i(\mathbf{x}, u_i, u_{-i})\ e^{-\rho t} dt \tag{4.1}$$

subject to

$$\begin{aligned} \text{s.t.} \quad & \dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}, u_i, u_{-i}) \\ & \mathbf{x}(0) = \mathbf{x}_0. \end{aligned} \tag{4.2}$$

where $\rho > 0$ is a given discount rate.

The work is focused on autonomous problems and stationary Markov-perfect Nash equilibria. Therefore, it is assumed that the state variable can always be observed and that, at any moment in time and space, the optimal player's decision will depend only on the particular state of the system, i.e. $u_i = u_i(\mathbf{x})$.

For a detailed proof of the following results see [7].

Definition 4.1. *Given $\mathbf{u}^s = (u_1^s, ..., u_N^s) \in \mathbb{U}$, a N-tuple of admissible strategies, it is said that u^s is a Markovian Nash equilibrium if, for all $\mathbf{x} \in \mathbb{X}$*

$$V_i(u_i^s, u_{-i}^s, \mathbf{x}) \geq V_i(u_i, u_{-i}^s, \mathbf{x}),\ \ i = 1, ..., N$$

and for all $u_i \in \mathbb{U}_i$.

The equations to solve the problem are given by [12, Theorem 4.1].

Theorem 4.2. *Let $(u_1^s, ..., u_N^s) \in \mathbb{U}$ be a N-tuple of admissible stationary strategies and assume that for $i = 1, ..., N$, there exist continuous differentiable functions*

$$W_i : \mathbb{X} \longrightarrow \mathbb{R}$$

such that the Hamilton-Jacobi-Bellman equations

$$\rho W_i(\mathbf{x}) = \max_{u_i \in \mathbb{U}_i} \left\{ f_i(\mathbf{x}, u_i, u_{-i}^s) + \nabla W_i(\mathbf{x})^T \mathbf{g}(\mathbf{x}, u_i, u_{-i}^s) \right\},\ \ i = 1, ...N, \tag{4.3}$$

are satisfied $\forall \mathbf{x} \in \mathbb{X}$.

Assume also that W_i is bounded or bounded below and that it holds the transversality condition

$$\limsup_{T\to\infty}\; e^{-\rho T} W_i(\mathbf{x}(T)) \leq 0$$

where $\mathbf{x}(t)$ is the solution of 4.2 with $u_i = u_i^s(\mathbf{x}(t)),\ i = 1, ..., N$.

If for $i = 1, ..., N$, $u_i^s(\mathbf{x})$ is an argument that maximizes the right hand of (4.3) for all $\mathbf{x} \in \mathbb{X}$, then $(u_1^s, ..., u_N^s)$ is a Markovian Nash Equilibrium.

It also holds that $V_i = W_i,\ i = 1, ..., N$.

To approximate the solution of the previous problem numerically, first a discrete-time reformulation is considered.

In the discrete-time infinite horizon version of the game, let $h > 0$ be a parameter and define $t_n = nh,\ n \in \mathbb{N}$. The discrete discount factor is given by $\beta_h = 1 - \rho h$.

Each player $i = 1, ...N$, tries to maximize

$$V_{i,h}(u_i, u_{-i}, \mathbf{x}_0) := h \sum_{n=0}^{\infty} \beta_h^n f_i(\mathbf{x}_n, u_{i,n}, u_{-i,n}), \tag{4.4}$$

$$\text{s.t.} \quad \mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{g}(\mathbf{x}_n, u_i, u_{-i}), \tag{4.5}$$

and where the controls correspond to $u_{*,n} = u_*(\mathbf{x}_n)$.

Assuming that (4.5) is well defined for any $\mathbf{x}_0 \in \mathbb{X}$ and all $(u_1, ..., u_n) \in \mathbb{U}$, the concept of Markovian Nash Equilibrium can be extended to the time-discrete version of the game.

Given a N-tupla $\boldsymbol{u}^{s,h} = \left(u_1^{s,h}, ..., u_N^{s,h}\right) \in \mathbb{U}$, it is a Markovian Nash Equilibrium of problem (4.4) if

$$V_{i,h}(u_i^{s,h}, u_{-i}^{s,h}, \mathbf{x}) \geq V_{i,h}(u_i, u_{-i}^{s,h}, \mathbf{x}),\ i = 1, ..., N, \tag{4.6}$$

for all $(u_1, ..., u_n) \in \mathbb{U}$ and all $\mathbf{x} \in \mathbb{X}$.

The discrete Bellman equations (see [13] and [16]) can also be obtained.

Theorem 4.3. *Let $\left(u_1^{s,h},...,u_N^{s,h}\right) \in \mathbb{U}$ be a N-tuple of admissible stationary strategies and assume that for $i = 1,...,N$, there exist continuous differentiable functions*

$$W_{i,h} : \mathbb{X} \longrightarrow \mathbb{R},$$

such that the Bellman equations

$$W_{i,h}(\mathbf{x}) = \max_{u_i \in \mathbb{U}_i} \left\{ h f_i(\mathbf{x}, u_i, u_{-i}^{s,h}) + \beta_h W_i \left(\mathbf{x} + h\mathbf{g}(\mathbf{x}, u_i, u_{-i}^{s,h}) \right) \right\}, \ i = 1, ...N, \tag{4.7}$$

are satisfied $\forall \mathbf{x} \in \mathbb{X}$.

Assume also that $W_{i,h}$ is bounded or bounded below and that it holds the transversality condition

$$\limsup_{n\to\infty} \beta_h^n W_{i,h}(\mathbf{x}(T)) \leq 0,$$

where $\mathbf{x}(t)$ is the solution of (4.5) with $u_i = u_i^{s,h}(\mathbf{x}(t)), \ i = 1, ..., N$.

If $u_i^{s,h}(\mathbf{x})$ is an argument that maximizes the right hand of (4.7) for $i = 1, ..., N$ and $\forall \mathbf{x} \in \mathbb{X}$, then $\left(u_1^{s,h},...,u_N^{s,h}\right)$ is a Markovian Nash Equilibrium and $V_{i,h} = W_{i,h}, \ i = 1, ..., N$.

Under certain regularity hypothesis (see [7]), it can be proved that the numerical solution of the time-discrete game is an ϵ-Nash solution of the continuous problem. If $\left(u_1^{s,h},...,u_N^{s,h}\right)$ is a Markov-Nash equilibrium of the discrete problem (4.4)-(4.5), it exists a constant $C > 0$ and a value $h_0 > 0$, such that $\forall h \leq h_0$ it holds that

$$V_i \left(u_i^{s,h}, u_{-i}^{s,h}, \mathbf{x} \right) \geq V_i \left(u_i, u_{-i}^{s,h}, \mathbf{x} \right) - Ch, \ i = 1, ..., N, \tag{4.8}$$

for all $(u_1, ..., u_n) \in \mathbb{U}$ and all $\mathbf{x} \in \mathbb{X}$.

In the following Sections, the main ideas of the *Value* and *Policy* techniques are sketched. It is also detailed how to implement both methods with a multidimensional Chebyshev interpolation.

4.1 Value iteration

In the *Value iteration* algorithm, the main idea is to employ, in each iteration and for each player, the previous value function and the previous controls to compute a better value function by means of the time-discrete Bellman equation.

Let I_k denote a k-degree multidimensional interpolation over domain $\mathbb{X}$ and let $\left\{\mathbf{x}_j = \left(x_1^j, ..., x_N^j\right)\right\}_{j=1,...,(k+1)^N} \subset \mathbb{X}$ be the corresponding collocation nodes. For simplicity in the notation, the same degree k has been taken for each dimension.

Let $h > 0$ and $\beta_h = 1 - \rho h$ the discrete discount factor.

Let $u_i^{[0]}$ and $V_i^{[0]},\ i = 1, ..., N$ be some initial candidates for the controls and the value functions.

At each iteration $[r+1]$, for each player $i = 1, ..., N$, a new strategy is computed for $j = 1, ..., (k+1)^N$

$$u_i^{[r+1]}(\mathbf{x}_j) = \underset{u_i \in \mathbb{U}_i}{\operatorname{argmax}} \left\{ h f_i(\mathbf{x}_j, u_i, u_{-i}^{[r]}) + \beta_h I_k V_i^{[r]} \left(\mathbf{x}_j + h\mathbf{g}(\mathbf{x}_j, u_i, u_{-i}^{[r]}) \right) \right\}.$$

The corresponding new value function for $j = 1, ..., (k+1)^N$ is

$$V_i^{[r+1]}(\mathbf{x}_j) = h f_i(\mathbf{x}_j, u_i^{[r+1]}, u_{-i}^{[r]}) + \beta_h I_k V_i^{[r]} \left(\mathbf{x}_j + h\mathbf{g}(\mathbf{x}_j, u_i^{[r+1]}, u_{-i}^{[r]}) \right).$$

The iterations continue until $\left|V^{[r+1]} - V^{[r]}\right| < \text{TOL}$, where TOL is a prescribed tolerance.

The main drawbacks of this method are that it may not converge or that it can fall in cycles of suboptimal solutions. Furthermore, this method presents a very slow error reduction and a large amount of iterations might be needed before reaching the prescribed tolerance.

One of the main benefits is that the computation has to be performed for each $\mathbf{x}_j,\ j = 1, ..., (k+1)^N$. At a fixed iteration, note that these computations are independent of each spatial node $\mathbf{x}_j$ from the others. This feature makes the method suitable for parallelization.

The algorithm presented below corresponds to the adaptation of the *Value iteration* with a multidimensional Chebyshev interpolation.

Value iteration algorithm

Step 0:

Let N be the number of players. Define an appropriate domain for the state and control spaces, where the problem will be numerically solved

$$\begin{aligned}[a_1, b_1] \times ... \times [a_N, b_N] \subset \mathbb{X},\\ [c_1, d_1] \times ... \times [c_N, d_N] \subset \mathbb{U}.\end{aligned}$$

Fix $N_x, N_u \in \mathbb{N}$, a tolerance TOL > 0 and a small time step $h > 0$.

For each player $i = 1, ..., N$, compute $\left\{\hat{x}^i_j\right\}_{j=0}^{N_x}$, $\left\{\hat{u}^i_j\right\}_{j=0}^{N_u}$, the Chebyshev nodes in each interval

$$\begin{aligned}\hat{x}^i_j &= \frac{1}{2}\left[\cos\left(\frac{\pi j}{N_x}\right)(b_i - a_i) + (b_i - a_i)\right],\ j = 0, ..., N_x,\\ \hat{u}^i_j &= \frac{1}{2}\left[\cos\left(\frac{\pi j}{N_u}\right)(d_i - c_i) + (d_i - c_i)\right],\ j = 0, ..., N_u.\end{aligned}$$

Define the sets of collocation points

$$\begin{aligned}\hat{\mathbf{X}} &= \left\{(\hat{x}^1_{j_1}, \hat{x}^2_{j_2}, ..., \hat{x}^N_{j_N}),\ j_i = 0, ..., N_x,\ i = 1, ..., N\right\},\\ \hat{\mathbf{U}} &= \left\{(\hat{u}^1_{j_1}, \hat{u}^2_{j_2}, ..., \hat{u}^N_{j_N}),\ j_i = 0, ..., N_u,\ i = 1, ..., N\right\},\end{aligned}$$

For simplicity in the notation, we employ $\hat{\mathbf{X}} = \left\{\hat{\mathbf{x}}_j,\ j = 1, ..., (N_x + 1)^N\right\}$.

Set $r = 0$. For each player $i = 1, ..., N$, initialize the iteration with some given $V^{N_x,[0]}_{h,i}(\hat{\mathbf{x}}_j)$ and $u^{[0]}_{h,i}(\hat{\mathbf{x}}_j)$, $j = 1, ..., (N_x + 1)^N$.

Step 1:

For each player $i = 1, ..., N$, compute $I_{N_x} V^{[r]}_{h,i}(\mathbf{x})$, the Chebyshev multidimensional interpolation polynomial such that

$$I_{N_x} V^{[r]}_{h,i}(\hat{\mathbf{x}}_j) = V^{N_x,[r]}_{h,i}(\hat{\mathbf{x}}_j),\ j = 1, ..., (N_x + 1)^N$$

employing the algorithms described in Chapter 3.

Step 2:

For each player $i = 1, ..., N$, compute the Chebyshev polinomials

$$G^i_{\hat{\mathbf{x}}_j}(u) = g_i(\hat{\mathbf{x}}_j, u, u^{[r]}_{h,-i}), \ j = 1, ..., (N_x + 1)^N,$$

employing the algorithms described in Chapter 3.

Each polynomial $j = 1, ..., (N_x + 1)^N$ is given by its coefficients. They can all be stored in a 2 dimensional array $\mathbf{G}$ of size $(N_u + 1) \times (N_x + 1)^N$, where $\mathbf{G}(:, j)$ contains the coefficients of polynomial $G^i_{\hat{\mathbf{x}}_j}(u)$.

The FFT algorithms employed admit *tensorial valuation*, i.e. several different interpolation polynomials of the same degree can be computed and evaluated at the same time. It is straightforward to adapt the Chebyshev interpolation algorithms of Chapter 3 so that all the polynomials $G^i_{\hat{\mathbf{x}}_j}(u), \ j = 1, ..., (N_x + 1)^N$ are computed at the same time and stored in array $\mathbf{G}$ which can be employed to evaluate simultaneously all of them.

As it will be shown in the numerical experiments, this gives an efficient implementation of the method.

Step 3:

For each player $i = 1, ..., N$ compute the Chebyshev polynomials

$$\mathcal{V}^{N_x,[r]}_{h,i,\hat{\mathbf{x}}_j}(u) = hf_i(\hat{\mathbf{x}}_j, u, u^{[r]}_{h,-i}) + \beta_h I_{N_x} V^{[r]}_{h,i}\left(\hat{\mathbf{x}}_j + hG^i_{\hat{\mathbf{x}}_j}(u)\right), \ j = 1, ..., (N_x + 1)^N.$$

As in the previous step, all the polynomials $\mathcal{V}^{N_x,[r]}_{h,i,\hat{\mathbf{x}}_j}(u), \ j = 1, ..., (N_x+1)^N$ can be computed at the same time.

Step 4:

For each player $i = 1, ..., N$ find the strategy which maximizes

$$u^{[r+1]}_{h,i}(\hat{\mathbf{x}}_j) = \underset{u \in [c_i, d_i]}{\operatorname{argmax}} \left\{ \mathcal{V}^{N_x,[r]}_{h,i,\hat{\mathbf{x}}_j}(u) \right\}, \quad j = 1, ..., (N_x + 1)^N.$$

To find the maximum, the Newton Raphson algorithm has been employed. If $F(u)$ denotes the function that we want to maximize, given an initial iterant u_i^0, the recursive iteration

$$u_i^{l+1} = u_i^l - \frac{F'(u_i^l)}{F''(u_i^l)}, \quad l = 0, 1, 2, ...$$

is followed until a certain tolerance is achieved.

To employ the Newton-Raphson method, we need to evaluate and compute the Chebyshev derivatives. Both the iteration of the Newton-Raphson method and the algorithms described in Chapter 3 can be adapted to compute the derivatives and maximums for all $\mathcal{V}_{h,i,\hat{\mathbf{x}}_j}^{N_x,[r]}(u),\ j = 1, ..., (N_x + 1)^N$ at the same time.

<u>Step 5:</u>

For each player $i = 1, ..., N$, define

$$V_{h,i}^{N_x,[r+1]}(\hat{\mathbf{x}}_j) = \mathcal{V}_{h,i,\hat{\mathbf{x}}_j}^{N_x,[r]}\left(u_{h,i}^{[r+1]}(\hat{\mathbf{x}}_j)\right),\ j = 1, ..., (N_x + 1)^N.$$

If we are not below the prescribed tolerance

$$\left|V_{h,i}^{N_x,[r+1]}(\hat{\mathbf{x}}_j) - V_{h,i}^{N_x,[r]}(\hat{\mathbf{x}}_j)\right| < TOL,\ j = 1, ..., (N_x + 1)^N,$$

set $r = r + 1$ and return to Step 1. Otherwise stop.

The last iterate $u_{h,i}^{[r]}(\hat{\mathbf{x}}_j),\ j = 1, ..., (N_x + 1)^N$ is taken as the numerical solution of the game.

It should be mentioned that function $\mathbf{g}$ and functions $f_i,\ i = 1, ..., N$ could be interpolated in $\mathbb{X} \times \mathbb{U}$ just once in Step 0 and evaluated when necessary. This could probably improve the efficiency of the algorithm.

Parallelization of the Value Iteration Algorithm

Value iteration algorithm requires a significant number of iterations to converge, and this quantity raises as the number of players is increased or the spatial grid is refined. For this reason, a parallelization of the process might be considered.

Steps 2-4 require the computation of polynomials for each state node $\hat{\mathbf{x}}_j,\ j = 1, ..., (N_x + 1)^N$. As mentioned before, these computations can all be performed at the same time, but note that they are independent for each $\hat{\mathbf{x}}_j,\ j = 1, ..., (N_x + 1)^N$.

Therefore set $\hat{\mathbf{X}} = \left\{(\hat{x}^1_{j_1}, \hat{x}^2_{j_2}, ..., \hat{x}^N_{j_N}),\ j_i = 0, ..., N_x,\ i = 1, ..., N\right\}$ can be divided in different subsets $\hat{\mathbf{X}}_m,\ m = 1, ..., N_b$ of N_f elements each one, such that $N_b \cdot N_f = (N_x + 1)^N$.

The calculus involved in Steps 2-4 can be performed in several different processors simultaneously and the information can be reassembled before Step 5. For a large number of players or a large value of N_x, these technique is more computational efficient.

4.2 Policy iteration

In the *Policy iteration* algorithm, the main idea is to employ, in each iteration and for each player, the previous controls to compute a polynomial such that it satisfies the time-discrete Bellman equation at all the collocation nodes of the state space. Afterwards, with that polynomial, a better control is obtained through the time-discrete Bellman equation.

Again, let I_k denote a k-degree multidimensional interpolation over domain $\mathbb{X}$ and let $\left\{\mathbf{x}_j = \left(x_1^j, ..., x_N^j\right)\right\}_{j=1,...,(k+1)^N} \subset \mathbb{X}$ be the corresponding collocation nodes.

Let $h > 0$ and $\beta_h = 1 - \rho h$ be the discrete discount factor. Let $u_i^{[0]}$, $i = 1, ..., N$, be some initial candidate for the controls.

At each iteration $[r]$ and for each player $i = 1, ..., N$, it is computed a polynomial $I_k V_i^{[r]}(\mathbf{x})$ such that, for all $j = 1, ..., (k+1)^N$, it holds

$$I_k V_i^{[r]}(\mathbf{x}_j) = h f_i(\mathbf{x}_j, u_i^{[r]}, u_{-i}^{[r]}) + \beta_h I_k V_i^{[r]} \left(\mathbf{x}_j + h\mathbf{g}(\mathbf{x}_j, u_i^{[r]}, u_{-i}^{[r]})\right),$$

It is important to remark that the previous condition defines a linear system, where the unknowns are the polynomial coefficients.

The update policy is given for $j = 1, ..., (k+1)^N$ by

$$u_i^{[r+1]}(\mathbf{x}_j) = \underset{u_i \in \mathbb{U}_i}{\operatorname{argmax}} \left\{ h f_i(\mathbf{x}_j, u_i, u_{-i}^{[r]}) + \beta_h I_k V_i^{[r]} \left(\mathbf{x}_j + h\mathbf{g}(\mathbf{x}_j, u_i, u_{-i}^{[r]})\right)\right\}.$$

The iterations continue until $\left|V^{[r+1]} - V^{[r]}\right| <$ TOL, where TOL is a prescribed tolerance.

The main drawbacks of this method is that a good initial guess for $u_i^{[0]}$ might be necessary and that the computational cost of solving the linear system grows with a large amount of players or collocation nodes.

The main benefits are that it is well known that this algorithm has a quadratic rate of convergence towards the solution for $N = 1$ (control problems). This behaviour seems to remain in Game Theory problems ($N > 1$).

The numerical experiments show that a much smaller number of iterations is needed to reach the prescribed tolerance with *Policy iteration* than with *Value iteration.*

The algorithm presented below corresponds to the adaptation of the *Policy iteration* with a multidimensional Chebyshev interpolation.

Policy iteration algorithm:

Step 0:

Let N be the number of players. Define an appropriate domain for the state and control spaces, where the problem will be numerically solved

$$[a_1, b_1] \times ... \times [a_N, b_N] \subset \mathbb{X},$$
$$[c_1, d_1] \times ... \times [c_N, d_N] \subset \mathbb{U}.$$

Fix $N_x, N_u \in \mathbb{N}$, a tolerance $\mathrm{TOL} > 0$ and a small time step $h > 0$.

For each player $i = 1, ..., N$, compute $\left\{\hat{x}_j^i\right\}_{j=0}^{N_x}$, $\left\{\hat{u}_j^i\right\}_{j=0}^{N_u}$, the Chebyshev nodes in each interval

$$\hat{x}_j^i = \frac{1}{2}\left[\cos\left(\frac{\pi j}{N_x}\right)(b_i - a_i) + (b_i - a_i)\right],\ j = 0, ..., N_x,$$
$$\hat{u}_j^i = \frac{1}{2}\left[\cos\left(\frac{\pi j}{N_u}\right)(d_i - c_i) + (d_i - c_i)\right],\ j = 0, ..., N_u.$$

Define the sets of collocation points

$$\hat{\mathbf{X}} = \left\{(\hat{x}_{j_1}^1, \hat{x}_{j_2}^2, ..., \hat{x}_{j_N}^N),\ j_i = 0, ..., N_x,\ i = 1, ..., N\right\},$$
$$\hat{\mathbf{U}} = \left\{(\hat{u}_{j_1}^1, \hat{u}_{j_2}^2, ..., \hat{u}_{j_N}^N),\ j_i = 0, ..., N_u,\ i = 1, ..., N\right\},$$

For simplicity in the notation, we employ $\hat{\mathbf{X}} = \left\{\hat{\mathbf{x}}_j,\ j = 1, ..., (N_x + 1)^N\right\}$.

For each player $i = 1, ..., N$, initialize the iteration with some given $u_{h,i}^{[0]}(\hat{\mathbf{x}}_j)$, $j = 1, ..., (N_x + 1)^N$.

For each player $i = 1, ..., N$, compute the Chebyshev polinomials

$$G_{\hat{\mathbf{x}}_j}^i(u) = g_i(\hat{\mathbf{x}}_j, u, u_{h,-i}^{[0]}),\ j = 1, ..., (N_x + 1)^N,$$

employing the algorithms described in Chapter 3.

As in the Value iteration algorithm, they can all be stored in a 2 dimensional array $\mathbf{G}$.

Compute the Chebyshev multidimensional polynomial $I_{N_x} V_{h,i}^{[0]}(\mathbf{x})$ such that, $\forall j = 1, ..., (N_x+1)^N$, it holds

$$I_{N_x} V_{h,i}^{[0]}(\hat{\mathbf{x}}_j) = h f_i(\hat{\mathbf{x}}_j, u_{h,i}^{[0]}, u_{h,-i}^{[0]}) + \beta_h I_{N_x} V_{h,i}^{[0]} \left(\hat{\mathbf{x}}_j + h G_{\hat{\mathbf{x}}_j}^i (u_{h,i}^{[0]}) \right).$$

If set $\{p_{i_1,...,i_N},\ i_j = 0, ..., N_x,\ j = 1, ..., N\}$ corresponds to the coefficients of polynomial $I_{N_x} V_{h,i}^{[0]}(\mathbf{x})$, the previous condition defines a linear system where:

- The $p_{i_1,...,i_N}$ are the unknowns.
- The elements of the matrix of the system can be computed in terms of products and sums of Chebyshev polynomials
- The independent term is given by $h f_i(\hat{\mathbf{x}}_j, u_{h,i}^{[0]}, u_{h,-i}^{[0]}),\ j = 1, ..., (k+1)^N$.

Set $r = 0$.

Step 1:

For each player $i = 1, ..., N$ compute the Chebyshev polynomials

$$\mathcal{V}_{h,i,\hat{\mathbf{x}}_j}^{N_x,[r]}(u) = h f_i(\hat{\mathbf{x}}_j, u, u_{h,-i}^{[r]}) + \beta_h I_{N_x} V_{h,i}^{[r]} \left(\hat{\mathbf{x}}_j + h G_{\hat{\mathbf{x}}_j}^i (u) \right),\ j = 1, ..., (N_x+1)^N.$$

As in the *Value iteration*, all this polynomials can be computed at the same time.

Step 2:

For each player $i = 1, ..., N$ find the strategy which maximizes

$$u_{h,i}^{[r+1]}(\hat{\mathbf{x}}_j) = \underset{u \in [c_i, d_i]}{\operatorname{argmax}} \left\{ \mathcal{V}_{h,i,\hat{\mathbf{x}}_j}^{N_x,[r]}(u) \right\},\quad j = 1, ..., (N_x+1)^N.$$

Again, the Newton Raphson algorithm has been employed.

Step 3:

For each player $i = 1, ..., N$, compute the Chebyshev polinomials

$$G^i_{\hat{\mathbf{x}}_j}(u) = g_i(\hat{\mathbf{x}}_j, u, u^{[r+1]}_{h,-i}),\ j = 1, ..., (N_x+1)^N,$$

employing the algorithms described in Chapter 3.

Compute the Chebyshev multidimensional polynomial $I_{N_x} V^{[r+1]}_{h,i}(\mathbf{x})$ such that, $\forall j = 1, ..., (N_x+1)^N$, it holds

$$I_{N_x} V^{[r+1]}_{h,i}(\hat{\mathbf{x}}_j) = h f_i(\hat{\mathbf{x}}_j, u^{[r+1]}_{h,i}, u^{[r+1]}_{h,-i}) + \beta_h I_{N_x} V^{[r+1]}_{h,i}\left(\hat{\mathbf{x}}_j + h G^i_{\hat{\mathbf{x}}_j}(u^{[r+1]}_{h,i})\right).$$

If we are not below the prescribed tolerance

$$\left| I_{N_x} V^{[r+1]}_{h,i}(\hat{\mathbf{x}}_j) - I_{N_x} V^{[r]}_{h,i}(\hat{\mathbf{x}}_j) \right| < TOL,\ j = 1, ..., (N_x+1)^N,$$

set $r = r + 1$ and return to Step 1. Otherwise stop.

The last iterate $u^{[r]}_{h,i}(\hat{\mathbf{x}}_j),\ j = 1, ..., (N_x+1)^N$ is taken as the numerical solution of the game.

Chapter 5

A numerical example

In this chapter, the proposed numerical methods are applied to the multi-regional transboundary pollution game presented in [9].

Consider a planar region which is divided in several subregions. Different regions may share or not a common boundary. Each subregion determines the value of a control variable, emissions, and a state variable, which is the average of the pollution in the corresponding subregion.

The model studies the pollution flows through the different subregions. As some subregions present common boundaries, the pollution does not remain only in the emitting region. Instead, pollution of one region may flow to adjacent regions, raising their pollution levels. Some physical conditions are included in the model. For example, when a region is placed next to the coast, a part of the pollution diffuses through the coast boundary.

Each subregion increases its income increasing the emission rate. However, the environmental damage has also a cost for each subregion. Therefore, the objective function for each subregion is to maximize a benefit which depends both on the emissions and the environmental damage measured through the pollution level.

Each subregion aims to maximize his own objective function. Hence the model is a N-player non-cooperative differential game. The objective is to find an stationary Markov-perfect Nash equilibria.

The model formulation is as follows. Let Ω be a planar region, divided in N subregions Ω_i, $i = 1, ..., N$, satisfying

$$\bar{\Omega} = \bigcup_i \bar{\Omega}_i, \qquad \Omega_i \cap \Omega_j = \varnothing, \qquad i \neq j.$$

Let us denote by u_i the control variable of player i, which represents the emission rate. Let also $\mathbf{p} = (p_1, p_2, ..., p_N)$ be the state variable, representing the pollution stock in each region. The control will depend on the pollution stock, i.e., $u_i = u_i(\mathbf{p})$.

A linear quadratic model is proposed. The state dynamics are linear, whether the payoff functions are quadratic in the state and the controls. For each player, $i = 1, ..., N$ the benefit and damage functions are given by

$$B_i(u_i) = u_i\left(A_i - \frac{u_i}{2}\right), \qquad D_i(p_i) = \frac{\varphi_i}{2}p_i^2,$$

where $A_i,\ \varphi_i,\ i = 1, ..., N$ are positive constants.

The problem for each player is to maximize, for an infinite time horizon, the following functional

$$\begin{aligned} J_i(u_i, u_{-i}, \mathbf{p}_0) &= \int_0^\infty e^{-\rho t}\left(B_i(u_i) - D_i(p_i)\right)\, dt \\ &= \int_0^\infty e^{-\rho t}\left(u_i(A_i - \frac{u_i}{2}) - \frac{\varphi_i}{2}p_i^2\right)\, dt. \end{aligned} \tag{5.1}$$

The dynamics of the stock of pollution are given by a parabolic partial differential equation depending on the spatial continuous variables. After some simplifying assumptions (see [9] for the details), the dynamics of the state variables is given by

$$\frac{dp_i}{dt} = \sum_{j=1}^{N} k_{ij}p_j - c_i p_i + \beta_i u_i, \quad i = 1, ..., N,$$

with an initial condition

$$p_i(0) = p_i^0, \quad i = 1, ..., N.$$

The coefficients of the previous dynamics have the following physical interpretation:

- $c_i p_i$ measures the natural decay of pollutant, where $c_i \geq 0$.
- $\beta_i u_i$ is the source term, where $\beta_i > 0$.
- k_{ij} is a diffusion term. It represents how the pollution flows between regions. Coefficients k_{ij} can be stored as a matrix $K = (k_{ij})$ where $k_{ij} = 0$ if there is no common border between subregions Ω_i and Ω_j. Term $k_{ii} = -\sum_{j \neq i} k_{ij}$. Matrix K defines a graph with one node for each subregion.

The rest of the Chapter is organized as follows.

First, an example will be analytically and numerically solved in a two-region scenario. Other examples, with different spatial distributions and number of players, will be also computed in order to show how different geographical characteristics lead to different solutions. Afterwards, employing the analytical solution of the first example, an empirical error analysis of the methods proposed in Chapter 4 will be carried out.

In [9], a linear-spline based Value iteration algorithm is proposed to find the Markovian Nash equilibrium of the game. A performance comparison of the methods proposed in this work between then and with respect to the spline-based method employed in [9] will also be made.

Finally, we mention that other approaches could be considered. In [8], the solution of the Bellman equations is approximated by a fixed point iteration using a collocation method based on piecewise cubic Hermite interpolation.

5.1 Two regions scenario

5.1.1 Analytical solution

The problem for a two player scenario is given for $i = 1, 2$, by

$$V_i(\mathbf{p}^0) = \max_{u_i} J_i(u_i, u_{-i}, \mathbf{p}_0) = \max_{u_i} \int_0^\infty e^{-\rho t} \left(u_i(A_i - \frac{u_i}{2}) - \frac{\varphi_i}{2} p_i^2 \right) dt, \tag{5.2}$$

$$\begin{aligned} &s.t. \quad \dot{p}_i = k_{ij} p_j + k_{ii} p_i + -c_i p_i + \beta_i u_i, \\ &p_1(0) = p_1^0, \quad p_2(0) = p_2^0. \end{aligned} \tag{5.3}$$

Let us define

$$f_i(u_i, \mathbf{p}) = u_i \left(A_i - \frac{u_i}{2} \right) - \frac{\varphi_i}{2} p_i^2, \; i = 1, 2$$

and

$$g_i(u_i, \mathbf{p}) = k_{ij} p_j + k_{ii} p_i - c_i p_i + \beta_i u_i, \; i, j = 1, 2, \; i \neq j.$$

Let the control u_i that maximizes the objective function be $\mathcal{U}_i^*(p_i, p_j)$.

Applying Hamilton-Jacobi-Bellman and after some calculus (see [9] for the details),

$$\mathcal{U}_i^*(p_i, p_j) = A_i + \beta_i \frac{\partial V_i}{\partial p_i}, \; i = 1, 2, \; i \neq j$$

and the partial differential equation to solve for $i, j = 1, 2$, $j \neq i$, is given by

$$\rho V_i(\mathbf{p}) = f_i(\mathcal{U}_i^*, \mathbf{p}) + \frac{\partial V_i}{\partial p_i} \, g_i(\mathcal{U}_i^*, \mathbf{p}) + \frac{\partial V_i}{\partial p_j} \, g_j(\mathcal{U}_j^*, \mathbf{p}). \tag{5.4}$$

The linear quadratic model suggest a quadratic solution, taking the following form

$$V_i(p_i, p_j) = \frac{1}{2} a_{ii} p_i^2 + b_{ii} p_i + d_i + \frac{1}{2} a_{ij} p_j^2 + b_{ij} p_j + e_i p_i p_j, \quad i, j = 1, 2, \quad i \neq j.$$

From (5.4) we get the following twelve Ricatti equations

$$a_{ii}^2\beta_i^2 + 2e_i(k_j i + \beta_j^2 e_j) - a_{ii}(2c_i - 2k_{ii} + \rho) - \phi_i = 0, \quad \text{(i)}$$

$$A_i a_{ii}\beta_i + A_j e_j \beta_j + b_{jj} e_i \beta_j^2 + b_{ij}(k_{ji} + \beta_j^2 e_j) - b_{ii}(c_i - k_{ii} - \beta_i^2 a_{ii} + \rho) = 0, \quad \text{(ii)}$$

$$(A_i + \beta_i b_{ii})^2 + 2\beta_j b_{ij}(A_j + \beta_j b_{jj}) - 2\rho d_i = 0, \quad \text{(iii)}$$

$$e_i(2k_{ij} + \beta_i^2 e_i) - a_{ij}(2c_j - 2k_{jj} - 2\beta_j^2 a_{jj} + \rho) = 0, \quad \text{(iv)}$$

$$\beta_i e_i(A_i + \beta_i b_{ii}) + \beta_j a_{ij}(A_j + \beta_j b_{jj}) + b_{ii}k_{ij} - b_{ij}(c_j - k_{jj} - \beta_j^2 a_{jj} + \rho) = 0, \quad \text{(v)}$$

$$e_i(c_i + c_j - k_{ii} - k_{jj} - a_{ii}\beta_i^2 - a_{jj}\beta_j^2 + \rho) - a_{ii}k_{ij} - a_{ij}(k_{ji} + \beta_j^2 e_j) = 0. \quad \text{(vi)}$$

$i, j = 1, 2,\ i \neq j.$

Therefore, we obtain

$$\mathcal{U}_i^*(p_i, p_j) = A_i + \beta_i \frac{\partial V_i}{\partial p_i}(p_i, p_j) = A_i + \beta_i \left(a_{ii}p_i + b_{ii} + e_i p_j\right), \quad (5.6)$$

and substituting in (5.3)

$$\dot{p}_i = p_i(k_{ii} - c_i + \beta_i^2 a_{ii}) + p_j(k_{ij} + \beta_i^2 e_i) + \beta_i A_i + \beta_i^2 b_{ii} \quad i, j = 1, 2,\ i \neq j. \quad (5.7)$$

The previous expression gives a system of linear differential equations and the solution is

$$\begin{aligned} p_1(t) &= C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t} + p_1^{SS}, \\ p_2(t) &= C_3 e^{\lambda_1 t} + C_4 e^{\lambda_2 t} + p_2^{SS}, \end{aligned} \quad (5.8)$$

where λ_1 and λ_2 are the eigenvalues of the matrix associated to the system, $C_i \in \mathbb{R},\ i = 1, .., 4$ and p_1^{SS}, p_2^{SS} are the steady state of the pollution stock in each region. In order to guarantee the existence of a stationary state, the eigenvalues λ_1 and λ_2 have to be negative.

Two examples for 2 players are proposed. In the first one, the transport of pollution is symmetric in the two regions, and there is not a border acting like a sink of pollution. In the second example, one of the borders is a sink of pollution. This sink of pollution is physically interpreted as a boundary with a coast or a big region with stock of pollution equal to zero. It can be added to the model with a term $k_{i0},\ i = 1, 2$.

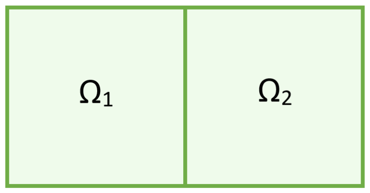
In order to focus on the spatial aspect of the two examples, equal parameters are assumed for both regions

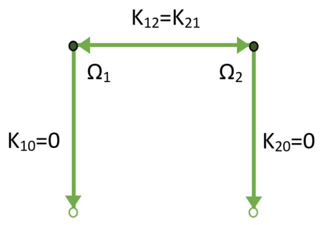
$$A = 0.5, \quad \phi = 1, \quad \beta = 1, \quad c = 0.5, \quad \rho = 0.01.$$

Example 1: Two regions isolated from outside

In this example, two isolated regions are considered.

The next scheme represents the geographical position of the regions Ω_1 and Ω_2. Also, this representation can be condensed in a graph. The pollution flux with external regions is expressed by means of the coefficients k_{i0}, $i = 1, 2$. In this case $k_{10} = k_{20} = 0$.





The matrix K containing the pollution flux between regions is

$$K = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The symmetry of matrix K allows to consider symmetric strategies and value functions. Therefore, following the analytical solution detailed on the previous section, $a_{jj} = a_{ii}$, $b_{jj} = b_{ii}$, $d_j = d_i$, $a_{ji} = a_{ij}$, $b_{ji} = b_{ij}$, $e_i = e_j$.

The Ricatti equations system is reduced to six equations with unknowns a_{ii}, b_{ii}, d_i, a_{ij}, b_{ij}, e_i.

From (ii) to (vi) the following expressions are obtained

$$b_{ii} = \frac{b_{ij} + A\beta(a_{ii} + e_i) + \beta^2 b_{ij} e_i}{1 + c - \beta^2 a_{ii} - \beta^2 e_i + \rho}, \tag{5.9a}$$

$$d_i = \frac{(A + \beta b_{ii})(A + \beta b_{ii} + 2\beta b_{ij})}{2\rho}, \tag{5.9b}$$

$$a_{ij} = \frac{(2 + \beta^2 e_i)e_i}{2(1 + c) - 2\beta^2 a_{ii} + \rho}, \tag{5.9c}$$

$$b_{ij} = \frac{A\beta(a_{ij} + e_i) + b_{ii}(1 + \beta^2 a_{ij} + \beta^2 e_i)}{1 + c - \beta^2 a_{ii} + \rho}, \tag{5.9d}$$

$$e_i = \frac{a_{ii} + a_{ij}}{2(1 + c) - 2\beta^2 a_{ii} - \beta^2 a_{ij} + \rho}. \tag{5.9e}$$

After several calculus (see [9]), six solutions exist for the system, but only one guarantees that the eigenvalues of the matrix associated to the system of differential equations (5.7) are negative. This solution is

$$a_{ii} = -0.354746,\ e_i = -0.108611,\ b_{ii} = -0.172108,$$
$$a_{ij} = -0.0552293,\ b_{ij} = -0.121105,\ d_i = 1.40474.$$

The optimal emission strategies are

$$\begin{bmatrix} u_1(p_1,p_2) \\ u_2(p_1,p_2) \end{bmatrix} = L \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \begin{bmatrix} 0.32789 \\ 0.32789 \end{bmatrix} \quad \text{where} \quad L = \begin{bmatrix} -0.35475 & -0.10861 \\ -0.10861 & -0.35475 \end{bmatrix}.$$

Functions u_1 and u_2 are the feedback Nash equilibrium strategies. They depend negatively on the state variable of the problem, the pollution stock. Moreover, the effect of the own pollution is greater, in absolute terms, than the effect of the pollution of the other region.

Due to the symmetric character of the problem and the identical parameter values, the optimal policy for both players is symmetric,

$$u_1(p_1,p_2) = u_2(p_2,p_1)$$

.

The solution of (5.7) gives the time paths of the pollution stock along the equilibrium strategy. They are given by

$$p_1(t) = \left((p_1^0 - p_1^{SS}) - (p_2^0 - p_2^{SS})\right)\frac{e^{\lambda_1 t}}{2} + \left((p_1^0 - p_1^{SS}) + (p_2^0 - p_2^{SS})\right)\frac{e^{\lambda_2 t}}{2} + p_1^{SS},$$
$$p_2(t) = \left((p_2^0 - p_2^{SS}) - (p_1^0 - p_1^{SS})\right)\frac{e^{\lambda_1 t}}{2} + \left((p_1^0 - p_1^{SS}) + (p_2^0 - p_2^{SS})\right)\frac{e^{\lambda_2 t}}{2} + p_2^{SS}.$$

where $\lambda_1 = -2.7461$, $\lambda_2 = -0.96336$ are the eigenvalues, the steady-state pollution stock values are $p_1^{SS} = p_2^{SS} = 0.340365$ and $p_i^0 = p_i(0)$, $i = 1, 2$ are the initial values of the pollution stocks.

If $p_1^0 = p_2^0$, and given that $p_1^{SS} = p_2^{SS}$, both paths are identical over time

$$p_1(t) = p_2(t) = (p_1^0 - p_1^{SS})e^{\lambda_2 t} + p_1^{SS}.$$

In figure (5.2), the particular case $p_1^0 = p_2^0 = 0.1$ is presented.

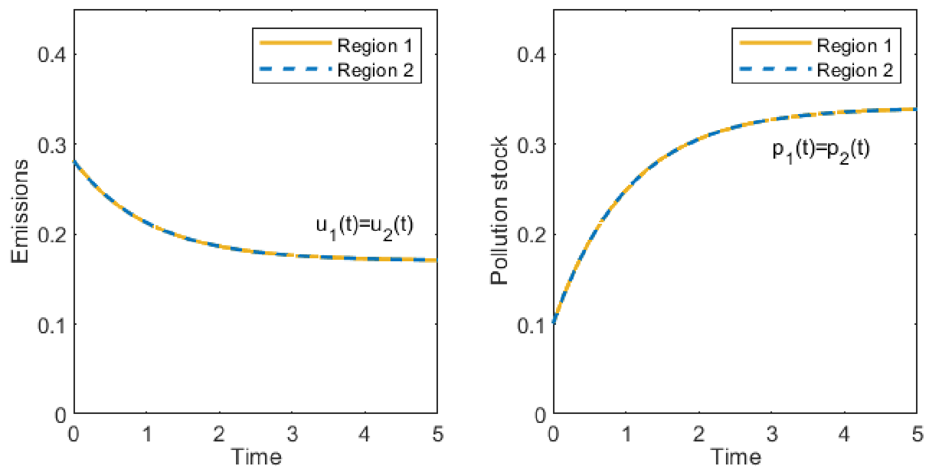


Figure 5.2: Emissions and pollution stock along the equilibrium.

As expected, the symmetrical character of the example provides identical trajectories for both players. The pollution levels increase from the initial pollution stock to the steady state value p^{SS}. Correspondingly, the emissions decrease with time until the control steady state value is reached.

Analytical solutions will be employed below in the Error analysis. The rest of the figures are from results obtained with the proposed numerical methods.

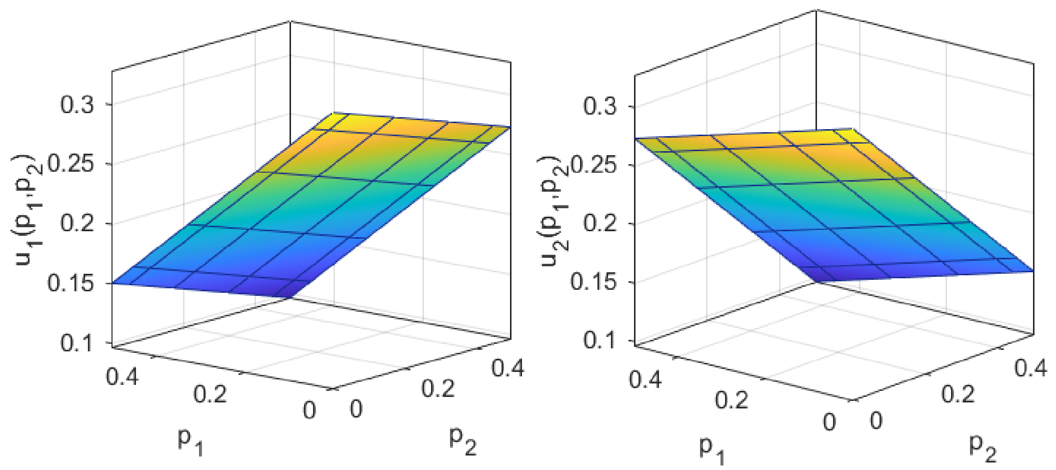


Figure 5.3: Regions' feedback strategies in $[0, 0.5] \times [0, 0.5]$

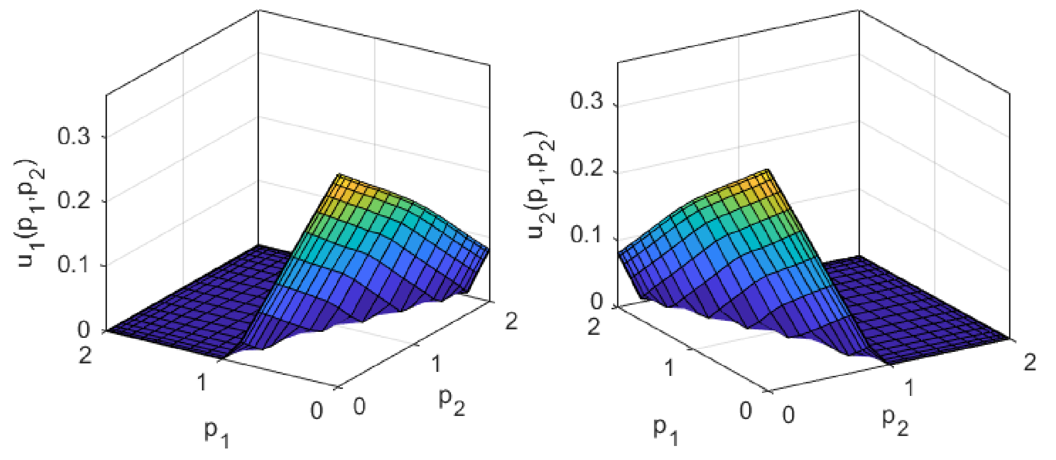


Figure 5.4: Regions' feedback strategies in $[0, 2] \times [0, 2]$.

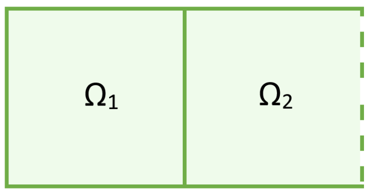
In Figures 5.3 and 5.4, the computed feedback strategies for regions Ω_1

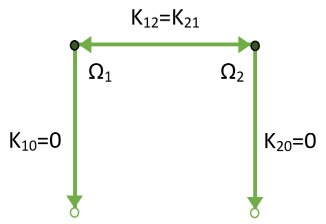
and Ω_2 are represented. The optimal emissions are obtained as a function of the pollution stocks p_1 and p_2. It can be appreciated that the emissions negatively depend on the pollution stock level of both regions.

The feedback strategies are presented for two different computational domains. In Figure 5.3, the pollution stock lies in domain $[0, 0.5] \times [0, 0.5]$, and feedback strategies are $\mathcal{C}^\infty$. However, in Figure 5.4 domain $[0, 2] \times [0, 2]$ has been employed and the solution is piecewise continuous. Emissions take value zero for high pollution stock values. It is worth to mention now that this different regularity in the computational domain leads to different spatial error convergence as it will be seen in Section 5.3.

Example 2: Two regions with different geographical neighbourhoods

In this example, one of the regions presents a border acting like a sink of pollution. Then, the problem is not symmetric for both players.





The pollution diffusion is

$$K = \begin{bmatrix} -1 & 1 \\ 1 & -2 \end{bmatrix}.$$

There exists a flux of pollution from region Ω_2 to the sink but not from Ω_1 to the sink. In this case $k_{10} = 0$ and $k_{20} = 1$.

The analytical solution for this example can also be computed (see [9]). The obtained optimal emission strategies are

$$\begin{bmatrix} u_1(p_1, p_2) \\ u_2(p_1, p_2) \end{bmatrix} = L \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \begin{bmatrix} 0.352463 \\ 0.437588 \end{bmatrix} \quad \text{where} \quad L = \begin{bmatrix} -0.344173 & -0.081391 \\ -0.051392 & -0.209672 \end{bmatrix}.$$

The eigenvalues are $\lambda_1 = -1.24801$, $\lambda_2 = -3.30584$ and the steady-state pollution stock values are

$$p_1^{SS} = 0.328921, \quad p_2^{SS} = 0.276641.$$

The different solutions between the two regions lies in their geographical aspects.

As in the previous example, we assume $p_1^0 = p_2^0 = 0.1$.

Figure 5.6 represents the evolution of the pollution stock and the emissions along the optimal time-path. Region Ω_2 presents a border acting like a sink of pollution. Consequently, the steady-state stock of pollution for region Ω_2 is lower than the steady-state value for region Ω_1.

Moreover, as the figure presents, the optimal emissions in region Ω_2 are always higher than in Ω_1, even Ω_1 generates a lower pollution stock. This means that an important flow of pollution is moving from region Ω_2 through the boundary to sink Ω_0.

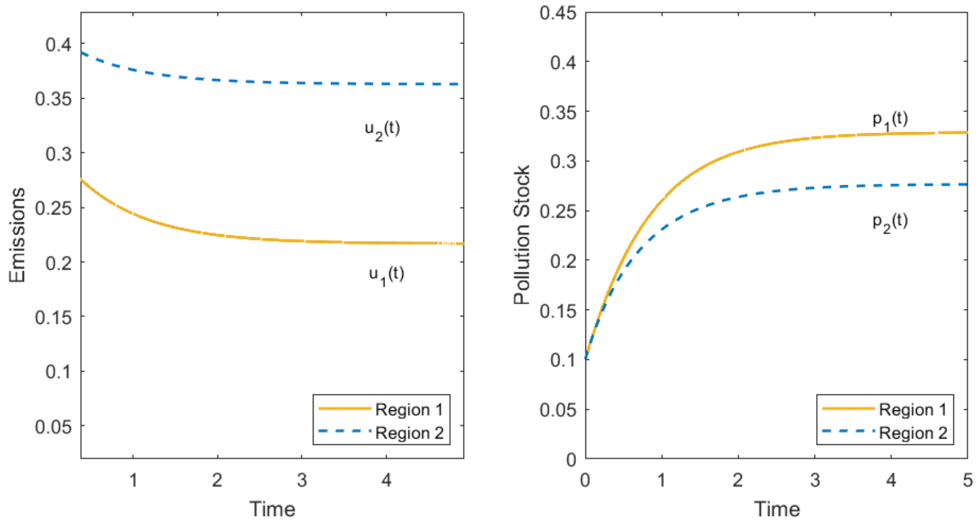


Figure 5.6: Emissions and pollution stock along the equilibrium.

Figure 5.7 shows the computed feedback strategies in function of the pollution stock levels. It can be appreciated that emissions in region Ω_2 are higher than in Ω_1.

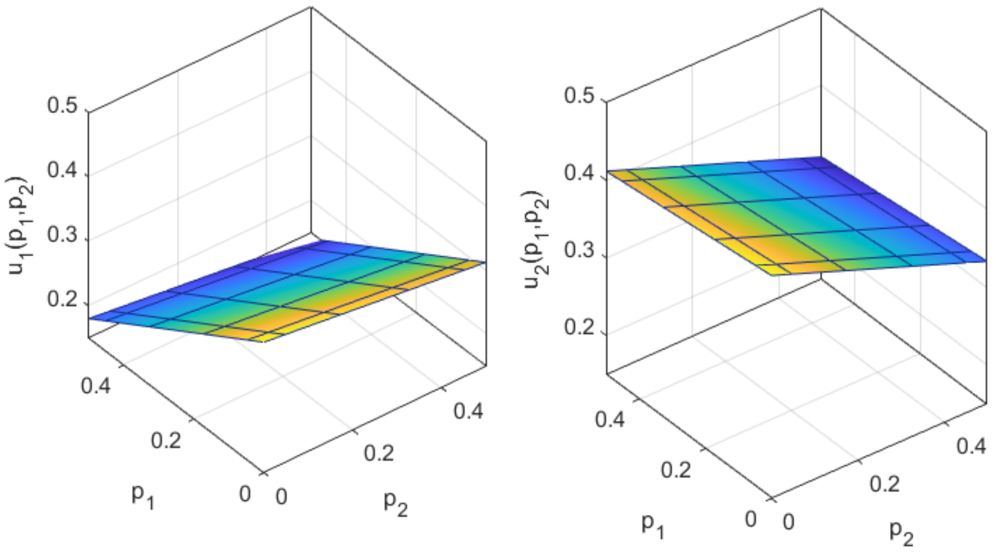


Figure 5.7: Regions' feedback strategies.

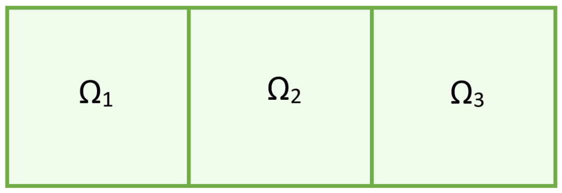
The different results obtained in *Example 1* and *Example 2* highlights the importance of including the spatial aspect in the transboundary pollution model. This idea will be reinforced with the following three-region scenario examples.

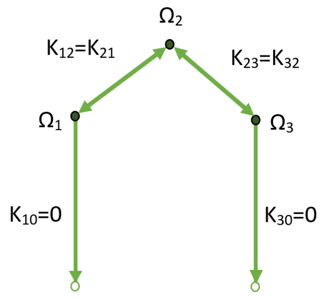
5.2 Three regions scenario

Example 3: Three regions isolated from outside

Let us consider $k_{10} = k_{20} = k_{30} = 0$, that is, three regions isolated from outside.

In this example, a flux of pollution appears from each region towards his neighbours. Pollution is passing from the central region Ω_2 to the sided regions Ω_1 and Ω_3, and also from the sides to the center region. Region Ω_1 and Ω_3 present symmetrical conditions.





The diffusion matrix K is

$$K = \begin{bmatrix} -1 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -1 \end{bmatrix}.$$

Figure 5.9 shows the trajectories for the emissions and the stock of pollution along the equilibrium strategy. The initial pollution stocks are set to 0.1.

As expected, regions Ω_1 and Ω_3 present the same trajectories for the emissions and pollution stock. Region Ω_2 benefits from its center position, profiting of the lower emissions of Ω_1 and Ω_3. With higher emissions, region Ω_2 presents almost the same pollution stock as the other regions.

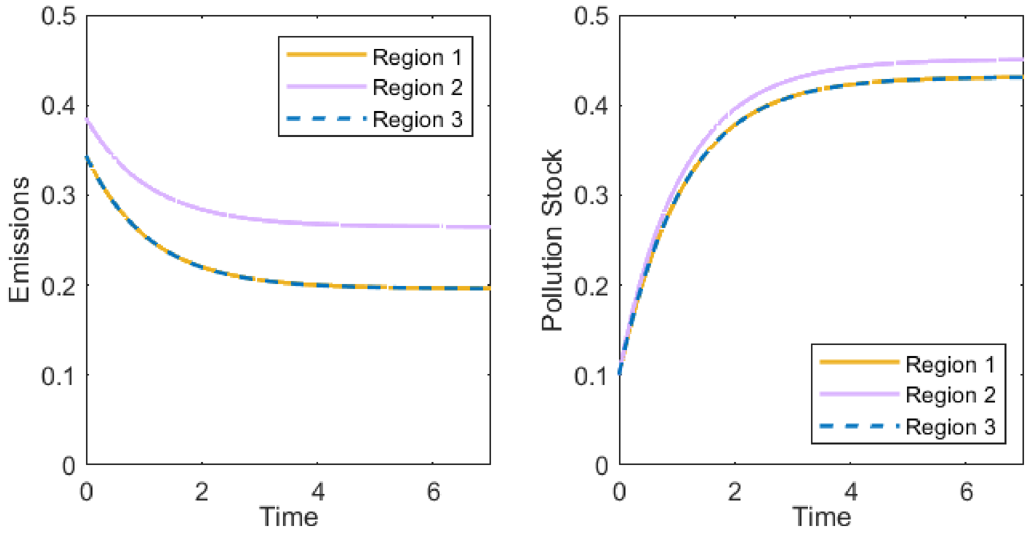
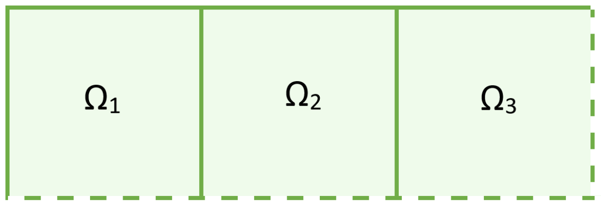


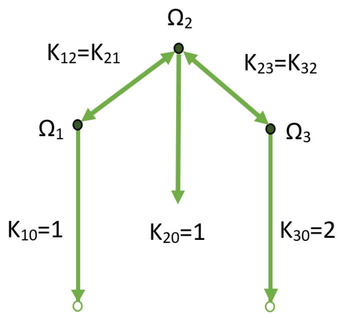
Figure 5.9: Emissions and pollution stock along the equilibrium.

Example 4: Three regions with different geographical neighbourhoods

In this example, a coast region is considered. The region is divided in three subregions that present different geographical conditions. Regions Ω_1 and Ω_2 present only one coast side, while Ω_3 present two coasts.

The flux of pollution through the coast is set as $k_{10} = k_{20} = 1$ and $k_{30} = 2$.





The matrix K containing the flux of pollution is

$$K = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -3 & 1 \\ 0 & 1 & -3 \end{bmatrix}.$$

Figure 5.11 represents the emissions and pollution stock along the equilibrium strategy. As expected, the three regions present different trajectories.

It should be noted that the coast side produces an important decreasing of the pollution rate. Therefore, the optimal emission strategies are much higher than in the *Example 3*. Region Ω_3 presents two coasts, something that allows this region to have higher emission levels.

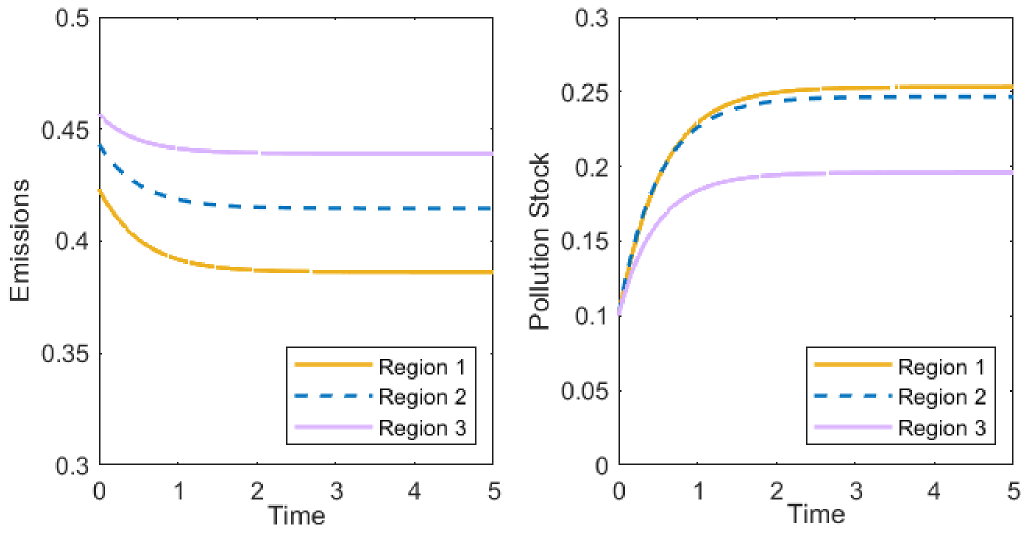


Figure 5.11: Emissions and pollution stock along the equilibrium.

In the exposed three-player examples, the importance of taking into account geographical characteristics in the transboundary pollution model is again enhanced.

5.3 Error analysis

Since the numerically computed solutions are those of a time-discrete game with a space discretization, numerical errors might have appeared.

In this Section, the spatial and temporal discretization errors are analysed empirically for *Example 1*, thanks to the fact that we have analytical solutions. For some theoretical results on the approximation error committed with fully discrete problems, see [10].

The error will be analysed in a fixed approximation domain for different computational domains, the domain where the solution is numerically computed.

The approximation domain for the analysis will be $[0, 0.5] \times [0, 0.5]$.

We fix a grid of equidistant nodes in the approximation domain D

$$D = \{(x_i, x_j),\ x_i = 0.05i,\ x_j = 0.05j,\ i, j = 0, ..., 10\}.$$

Since in *Example 1* the solutions were symmetric, the values that will be compared will be the ones obtained computing

$$V_1(\mathbf{p}^0) = \int_0^\infty e^{-\rho t} \left(u_1(A_1 - \frac{u_1}{2}) - \frac{\varphi_1}{2} p_1^2 \right)\, dt, \tag{5.10}$$

subject to the dynamics of the system with the parameter values employed in *Example 1.* The controls employed in computing the previous functional will be the numerical/analytical controls obtained.

The numerical controls u_n are obtained through the interpolation of the solution obtained in the computational domain. The error measured will be the root of the mean square error between the analytical and the numerical solution (u_a).

$$RMSE = \sqrt{\frac{1}{|D|} \sum_{\mathbf{p} \in D} \left(V_1(\mathbf{p}; u_n) - V_1(\mathbf{p}; u_a)\right)^2}.$$

Two different computational domains are considered, $[0, 0.5] \times [0, 0.5]$ and $[0, 2] \times [0, 2]$.

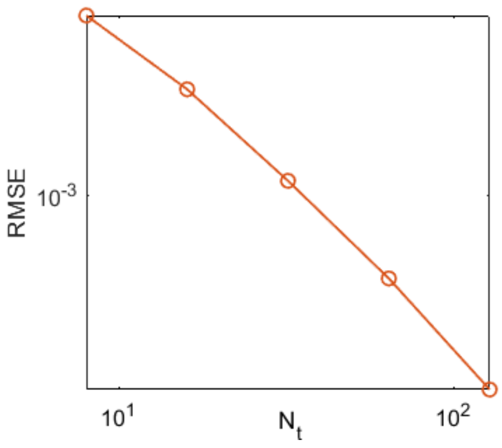
In $[0, 0.5] \times [0, 0.5]$, the value function is $\mathcal{C}^\infty$. However, in $[0, 2] \times [0, 2]$ is only piecewise regular. In Spectral methods, the order of the method depends on the regularity of function to be interpolated (see [3, Ch 2]). Specifically, it depends on how fast the expansion coefficients decay to zero.

5.3.1 Temporal error in $[0, 0.5] \times [0, 0.5]$

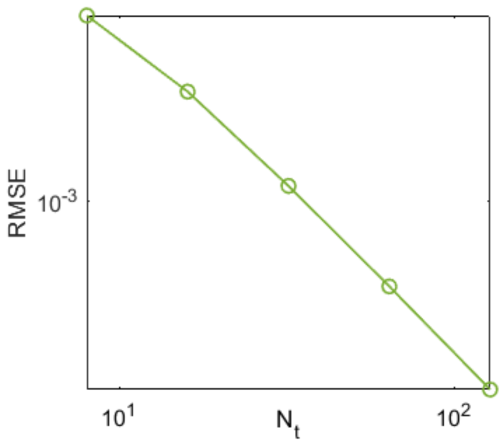
In this section, the temporal error is studied for the Value iteration (Chebyshev and splines-based) and Policy iteration.

The idea to study the temporal error is to take a sufficiently refined spatial discretization (N_x large), so that the spatial error does not interfere in the temporal analysis. The error will be computed for different values of $h > 0$, which is the parameter employed to define de discrete-time game of the numerical approximation.

Let $h = \frac{1}{N_t}$. The numerical solutions will be computed doubling the number N_t, i.e., h is halved in each of the successive computations.



(a) Value iteration



(b) Policy iteration

Figure 5.12: Temporal error in logarithmic scale, domain $[0, 0.5] \times [0, 0.5]$.

Figure 5.12 shows the RMSE for each choice of N_t, where both axis are represented in logarithmic scale. A linear tendency of the behaviour of the error as we increase the value of N_t can be appreciated. The slope is close to one.

The behaviour of the temporal error is almost identical for the Value iteration and the Policy iteration algorithms. For the splines-based Value iteration method the same result is also obtained.

The behaviour of the temporal error is similar in both computational domains.

For the previous analysis, the number of iterations were stored. The following tables show these values for each time discretization steps.

	Value iteration				
N_t	8	16	32	64	128
Number of iterations	9613	18079	33911	63344	117765

Table 5.1: Iterations for each time step, Value iteration.

	Policy iteration				
N_t	8	16	32	64	128
Number of iterations	140	140	140	140	140

Table 5.2: Iterations for each time step, Policy iteration.

The previous results show that many less iterations are needed in the policy method. This is a well known fact in Control Problems and seems to hold in Game Theory problems.

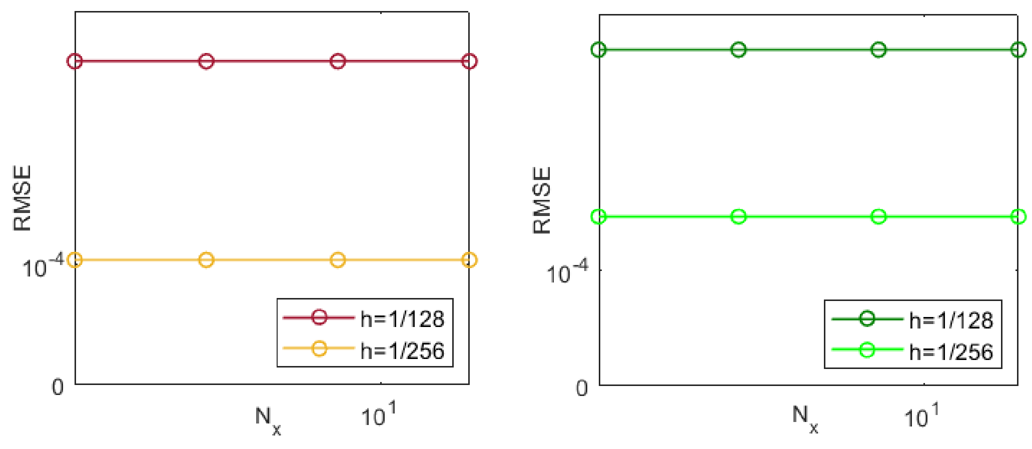
The results obtained for the splines-based Value method were very similar to the Chebyshev based Value method.

5.3.2 Spatial error in $[0, 0.5] \times [0, 0.5]$

The spatial error is analysed in the same way as the temporal error. Taking a large value of N_t, and the number of spatial nodes N_x is successively multiplied by two.

The analytical solution of the problem restricted to domain $[0, 0.5] \times [0, 0.5]$ is a polynomial. Therefore, there should not be any numerical error related with the spatial discretization in this domain. The error should depend just on the size of h.

Figure 5.13 represents, for two different time steps, the error committed in function of the spatial nodes. As expected, the result is that the spatial error does not depend on the space discretization. Then, a constant error is obtained, where the magnitude depends on the time step.



(a) Value iteration (b) Policy iteration

Figure 5.13: Spatial error in logarithmic scale, domain $[0, 0.5] \times [0, 0.5]$.

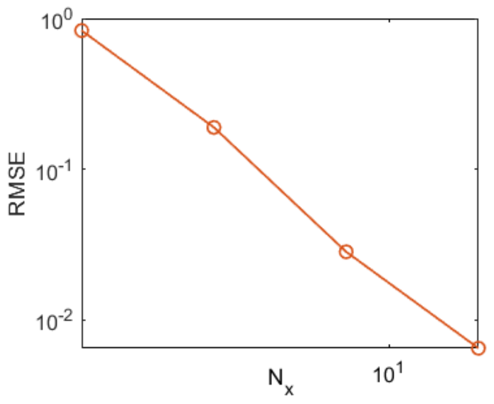
5.3.3 Spatial error in $[0, 2] \times [0, 2]$

The previous situation, in which there was no spatial error, changes when the solution is not a polynomial.

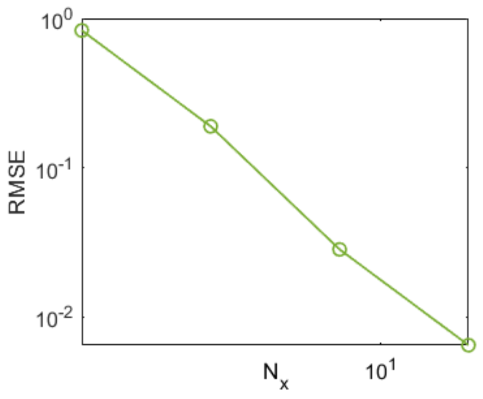
In the computation domain $[0, 2] \times [0, 2]$ the value function is piecewise regular.

Therefore, an error related with the spatial discretization should be appreciated. In Figure 5.14, the spatial error is represented for the Value iteration and Policy iteration algorithms. The same tendency is obtained when analysing the spatial error in $[0, 2] \times [0, 2]$ for the spline-Value iteration method.

A linear tendency of the error on the spatial discretization is perceived. The slope of this linear tendency is close to two.



(a) Value iteration



(b) Policy iteration

Figure 5.14: Spatial error in logarithmic scale, domain $[0, 2] \times [0, 2]$.

The order of error convergence is identical for the three different methods: Value iteration, Policy iteration and splines-Value iteration. The difference between the three methods lies in the computational cost.

5.4 Performance analysis

Computational performance of the numerical Value iteration and Policy iteration algorithms is analysed in this section.

The computational time and the error committed is stored for different values of the space discretization, time step and algorithm toleration. Figure 5.15 shows the error in function of the computational time for Policy iteration algorithm. The lower convex hull is also represented for the obtained points.

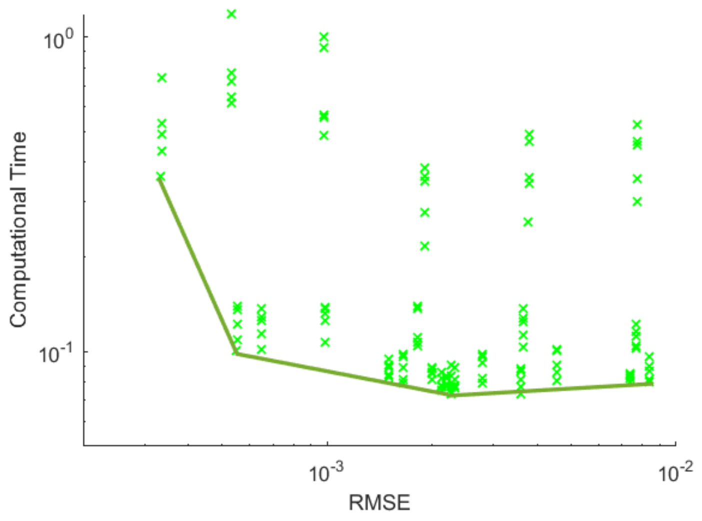


Figure 5.15: Error and computational time for Policy iteration.

Taking also the convex hull for Value iteration and spline-Value iteration, we obtain error-time curves that can be used to compare the three methods.

For a certain error value, this curve shows the least time needed to reach that precision. The same way, for a given computational time the curve indicates the best error rate that can be obtained.

In Figure 5.16, convex hulls for the three algorithms is presented.

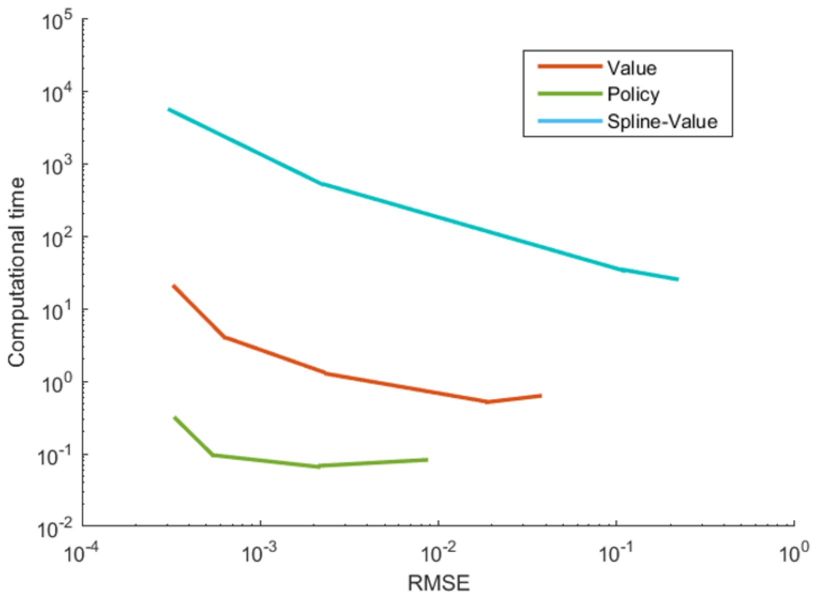


Figure 5.16: Error-time curves of the three algorithms.

It can be appreciated that the best algorithm in computational terms is Policy iteration. Both Chebyshev algorithms proposed in the present work outperform the spline-Value iteration algorithm computation time.

Chapter 6

Conclusions

In the present work, Chebyshev polynomial interpolation is applied to the Value iteration and Policy iteration algorithms. These algorithms are employed to approximate the Markovian Nash equilibria of multi-player differential games.

The general conclusion is that the Chebyshev interpolation scheme is a precise and computationally fast method. It allows to solve efficiently the maximization problem required by the previous algorithms.

A transboundary pollution model proposed in [9] has been studied as a numerical example. For a two-region scenario, the analytical solution has been compared to the numerical approximation obtained through different methods.

Value iteration and Policy iteration numerical results have been compared to the spline-based Value iteration scheme proposed in [9] and they have proved competitive.

All the methods present similar error behaviour. The main difference lies in the computational cost. The results extracted from the performance analysis is that Chebyshev-based Value algorithm require much less computational time than the spline-Value iteration. Moreover, the Policy iteration method requires very few iterations to converge to the solution.

Game theory is a broadly employed tool, and this methods can be adapted to problems in other fields, like economics and management science [12], industrial organization [4] or marketing [14], among others.

One of the possible extensions of this work could be the study of numerical methods to solve differential games with multiple control variables, as in [6], where emissions and investment in clean technologies are the two controls considered in the problem. The adaptation and efficient implementation of these methods to more general problems is a subject of interest. For example, the main computational cost of the Policy iteration method is the numerical computation of the polynomial coefficients. Maybe iterative methods for computing them could be considered. Another field of interest would be a theoretical analysis of the error behaviour of Chebyshev-based methods in Game Theory.

Bibliography

[1] Başar, T., Haurie, A., and Zaccour, G. *Nonzero-sum differential games*. Springer, Germany, Aug. 2018, pp. 61–110.

[2] Bauso, D. *Game Theory with Engineering Applications*. Society for Industrial and Applied Mathematics, 2016.

[3] Canuto, C., Hussaini, Y., Quarteroni, O. M. A., and Zang, T. *Spectral Methods: Fundamentals in Single Domains*, 2006. corr. 4th printing 2010 ed. ed. Springer, 2006.

[4] Colombo, L., and Labrecciosa, P. *Differential Games in Industrial Organization*. Springer International Publishing, Cham, 2017, pp. 1–46.

[5] de Frutos, J., and Gatón, V. Chebyshev reduced basis function applied to option valuation. *Computational Management Science 14*, 4 (2017), 465–491.

[6] De Frutos, J., Gatón, V., López-Pérez, P., and Martín-Herrán, G. Investment in cleaner technologies in a transboundary pollution dynamic game: A numerical investigation. *Dynamic Games and Applications* (04 2022), 1–31.

[7] de Frutos, J., Gatón, V., and Novo, J. On discrete-time approximations to infinite horizon differential games, 2021.

[8] de Frutos, J., and Martín-Herrán, G. Does Flexibility Facilitate Sustainability of Cooperation Over Time? A Case Study from Environmental Economics. *Journal of Optimization Theory and Applications 165*, 2 (2014), 657–677.

[9] de Frutos, J., and Martín-Herrán, G. Spatial effects and strategic behavior in a multiregional transboundary pollution dynamic game. *Journal of Environmental Economics and Management 97*, C (2019), 182–207.

[10] de Frutos, J., and Novo, J. Optimal bounds for numerical approximations of infinite horizon problems based on dynamic programming approach, 2021.

[11] de Leva, P. Multiprod algorithm., 07 2010.

[12] Dockner, E. J., Jorgensen, S., Long, N. V., and Sorger, G. *Differential Games in Economics and Management Science.* Cambridge University Press, 2000.

[13] Haurie A., Krawczyk J.B., Z. G. *Games and dynamic games.* World Scientific, Singapore, 2011.

[14] Jørgensen, S. *Marketing.* Springer International Publishing, Cham, 2016, pp. 1–41.

[15] Kirk, D. *Optimal Control Theory: An Introduction.* Dover Publications, 2004.

[16] Krawczyk, J.B., P. V. *Handbook of Dynamic Game Theory.* Bacar T., Zaccour G., eds., Springer Nature, 2018.

[17] Navarro, J. P., and Tena, E. C. *Teoría de juegos.* Pearson Educación, 2003.

[18] Rivlin, T. *Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory.* Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. Wiley, 1990.

[19] Seierstad, A. Optimal control theory with economic applications / atle seierstad, and knut sydsaeter, 1999.

[20] Yeung, D. W., and Petrosyan, L. A. *Nontransferable Utility Cooperative Dynamic Games.* Springer International Publishing, Cham, 2017, pp. 1–38.