university of groningen

faculty of science and engineering

Interventions using optimal control for social welfare maximization in network games

Master's Research Project — WMIE901-30

Industrial Engineering & Management

Supervisors:
Dr. N. (Nima) Monshizadeh Naini
Dr. A.K. (Ashish) Cherukuri
PHD M. (Mehran) Shakarami

Author:
J.D. (Jan Douwe) Breeuwsma - S3229262

March 30, 2022

Abstract

Keywords — Network games, social welfare, optimal control, admissible controls, penalty method

In this research, optimal control theory was applied as a means of social welfare maximization by a regulator in a linear quadratic network game. After setting the conditions for achieving a social welfare state in the network game, we derive the optimal intervention policy for two situations: one with a constraint on the admissible control set, and one where the intervention is penalized through an additional penalty term in the players' cost functions. By introducing an adaptation of the forward-backward sweep method used for numerical simulations, convergence to the social welfare state could be ensured for different values of the penalty parameter α (denoting the amount of penalization on the amount of intervention). In the following experiment, we noticed that, compared to existing intervention policies, the newly derived policy could not provide lower costs for the regulator. Therefore, the optimal control problem was altered, and new numerical results were obtained showing improved results where fast convergence was achieved, and lower regulator costs could be achieved. Hence, we showed that optimal control for social welfare maximization in such network games is indeed possible. Finally, recommendations for further research are given to strengthen the argument that the application of the derived intervention policies can truly be called optimal.

Preface

In front of you lies the report of the Master's Research Project on the use of optimal control theory for social welfare maximization in network games. This report was written for the Smart Manufacturing Systems research group of the University of Groningen as a part of finalizing my Master's in Industrial Engineering and Management.

Hereby, I would like to thank all those who have contributed to this research. Specifically, I would like to thank my first supervisors Dr N. Monshizadeh Naini and PHD M. Shakarami for their excellent guidance during the project and allowing me to conduct this research. Although changing conditions regarding COVID-19 regulations or busy schedules on both sides posed some challenges, I found the discussions we had regarding both the research in general, and the more specific problems that I encountered to be very helpful. Furthermore, I would like to thank my second supervisor Dr A.K. Cherukuri for his feedback on the final report and presentation. And last but not least, I would like to thank my fellow students, friends, and family for their ongoing support during the project.

Contents

Nomenclature

FBSM Forward-backward sweep method

PMP Pontryagin's Minimum Principle

PoA Price of Anarchy

RK4 Runge-Kutta 4

List of Figures

Chapter 1

Introduction

1.1 Research context

Within many different social or economic settings, the actions of individual decision-makers are influenced by the actions of their peers. In the literature, network games have emerged as a powerful framework for studying and modelling such settings, also covering interactions between populations with a large number of agents (Parise and Ozdaglar 2021). This framework, in addition, has applications in engineering settings such as optimization of supply chains(Vasnani et al. 2019), traffic flows (Bui and Jung 2018), and the coordinated charging of electric vehicles (Ma, Callaway, and Hiskens 2011). Here, network games differentiate from regular game theory since, in network games, the payoffs of individual agents is not based on the strategy of all other players, but only on those of the agents in the player's direct neighbourhood. In this research, we will look at network games from the perspective of a central regulator trying to steer the behaviour of agents in a network towards a social optimum. This game is considered to be non-cooperative since initially, the players only aim to minimize their individual cost functions (Khan and Sun 2002). This selfish competitive behaviour of the players causes for a Nash equilibrium where the solution is not socially optimal for all players. In literature, this loss of efficiency is measured by the so-called price of anarchy (PoA), being the utility ratio between the worst possible Nash solution and the social optimum (Başar and Zhu 2011; Deori, Margellos, and Prandini 2018; Zhu and Başar 2010).

For regulators, methodologies for setting up incentives for players to alter their behaviour and decrease the PoA often require private information of the individual agents (Başar and Zaccour 2018). In competitive environments, however, it could happen that private information such as players' possible actions, strategies, and objective functions is not publicly available. A methodology not requiring such detailed information applies to control theoretic tools to design intervention mechanisms, such as is done in Shakarami, Cherukuri, and Monshizadeh (2021). Here, the regulator observes the actions of the players over time, therefore not requiring their private information. In this sense, the problem can be regarded as a feedback control problem, with the interventions performed by the regulator being the control input, and the social welfare of the network being the desired outcome.

1.2 Problem analysis

In this research, we will consider a regulator trying to maximize the social welfare of a network with players having linear quadratic utility functions. This utility structure implies that players' cost functions will have both an individualized component, and a component that reflects the local interaction of the players, meaning that each player can be affected differently by different

neighbours (Corbo, Calvó-Armengol, and Parkes 2007).

With this model, the regulator's problem will be to maximize the total utility of the network, also known as social welfare, by designing a suitable intervention policy. In Shakarami, Cherukuri, and Monshizadeh (2021), two intervention protocols that achieve social welfare have already been discussed. These static and dynamic interventions steer the network to a social optimal state, depending on the available information on the players' parameters. However, in some cases, the regulator can only apply a limited amount of intervention and is therefore subjected to budget constraints. With that, the problem can be turned into a constrained optimization problem, where the optimal intervention protocol can be formulated using the optimality conditions found by applying Pontryagin's minimum principle (PMP) (Geering 2007). With this technique, the maximum amount of intervention is set as a *hard constraint*. Here, the derived expression for the optimal controller will usually take the form of a switching function, causing the intervention applied by the regulator to be either 0% or 100%. Since in some applications the regulator could also wish to apply an intervention of a different size than 100%. Therefore, the possibility arises for the regulator to reformulate the budget constraint using a penalty term that can be added to the utility function of the individual players. By doing so, an adjustable penalty parameter is introduced to represent the amount of influence the regulator has on the network, thus making the budget constraint more a *soft constraint* (Dolgopolik and Fominyh 2019). To summarize the above, the problem statement for this research becomes *within network games with linear quadratic cost functions, budget constraints concerning the amount of intervention a regulator can apply open up the possibility of the application of optimal control theory for deriving optimal intervention policies. In these optimal control problems, the budget constraints can be formulated both as a hard constraint and as a soft constraint. Due to the novelty of this application, however, the proof is still required on whether this so-called optimal controller can indeed reach social welfare and whether it can truly be called optimal compared to existing policies.*

1.3 Research goal

Given the problem statement from the previous section, the goal of this project will be to *assess the effectiveness of optimal control theory for maximizing social welfare in network games having linear quadratic utilities, taking into account regulator budget constraints formulated as both hard and soft constraints.*

1.3.1 Research questions

To achieve the goal of this research, research questions have been devised which split up the following main research question: *How does the optimal control solution with control constraints compare to the solution where a penalty function is used and what is the effect of the different controllers on reaching a social welfare solution?*

RQ-1 What are the convergence properties of the network game with linear quadratic cost functions?

RQ-2 How is the optimal control problem in a linear quadratic network game solved in both the constrained and unconstrained case, and how to ensure that this solution converges to the social optimum?

RQ-3 What is the effect of different sizes of interventions on the solution of the penalized optimal control problem?

RQ-4 How does the derived optimal intervention policy compare to static and dynamic feedback intervention policies?

1.4 Outline of the report

Based on the research questions above, the report is outlined as follows. First, chapter 2 will be used to introduce the theories involved with network games, optimal control, and the necessary numerical methods to clarify the complexities involved in the problem. Subsequently, in this chapter, we will also analyse the used network game model and derive the conditions for the social welfare solution. Next, chapter 3 is used to elaborate on the derivation of the designed optimal intervention policies and give the necessary conditions required for an optimal solution for both the constrained and the penalized case. Along with that, this chapter will show how the optimal intervention policy finds its connection with the social welfare maximization problem. Then, in chapter 4, experiments for numerical simulations are proposed and performed to confirm whether the obtained intervention policies indeed steer the network to a social welfare solution and whether the policies can indeed be called optimal compared to previous literature. Finally, chapter 5 is used to reflect on the results, describe the limitations of the research, and give recommendations for further research on this topic.

Chapter 2

Theoretical background

2.1 Network games and interventions

2.1.1 Modelling the network game

In this research, we consider network games with a population of $\mathcal{I} = \{1, ..., n\}$ players interacting with a central regulator, as well as with each other (Parise and Ozdaglar 2021). For such a given network of interactions between players, $P \in \mathbb{R}^{n \times n}$ denotes the adjacency matrix, where each node $P_{ij} \in [0, 1]$ denotes the amount of influence player j has on player i. In these networks, no self loops should occur, implying that $P_{ii} = 0$ for all i. Additionally, we use the notion $P_{ij} = P_{ji}$ to say that the network under consideration is directed, and otherwise it is undirected. The set of neighbours of player i is denoted by the set $\mathcal{N}_i = \{j \in \mathcal{I} | P_{ij} > 0\}$ Within the network, each player i chooses a strategy $x_i \in \mathcal{X}_i$ to minimize a cost function. Here, $\mathcal{X}_i$ denotes the strategy space of player i, usually being a finite set in $\mathbb{R}^n$. Here, each player i has a cost function $U_i(x_i, z_i(x))$, which depends on the player's own strategy $x_i \in \mathbb{R}$ and the *network aggregate*

$$z_i(x) = \sum_{j \in \mathcal{N}_i} = P_{ij} x_j, \tag{2.1}$$

with $x = [x_1; ...; x_i]$. In this research, we will consider the minimization of linear quadratic cost functions, taking the form

$$W_i(x_i, z_i(x)) = \frac{1}{2} x_i^2 - x_i(a z_i(x) + b_i), \tag{2.2}$$

where $a \in \mathbb{R}$ captures the impact of the network aggregate $z_i(x)$ on the cost of player i, and $b_i \in \mathbb{R}$ denotes the player's *standalone marginal return*. Moreover, in the case that $a > 0$, we say that this is a game of strategic complements, where players' actions have positive effects on their neighbours' cost functions. In the other case, when $a < 0$, we are talking about a game of strategic substitutes and the actions of each player i have a negative spillover effect on its neighbours (Currarini and Feri 2015). As can be seen in equation (2.2), the strategy of player i has a quadratic effect on the cost, and a linear effect on the benefits. In that way, this simple parametric form allows for a tractable analysis of the impact of interventions on the network structure.

2.1.2 Interventions

To decrease the PoA of the network, the regulator wishes to influence the players' actions to steer them towards social welfare. From the structure of the linear-quadratic cost function in equation (2.2), it can be seen that a player's action x_i creates a standalone marginal return

that is independent of the actions of the other players. In that sense, the costs of an individual player depending on how strongly the standalone marginal return is affected by the actions of other players, which is denoted by the term $az_i(x)$ in equation (2.2) (Galeotti, Golub, and Goyal 2020). With that, the regulator seeks to alter the standalone marginal returns of individual players. Hence, the term $x_i u_i$ is subtracted from the individual player's cost function such that it becomes:

$$U_i(x_i, z_i(x), u_i) = W_i(x_i, z_i(x)) - x_i u_i = \frac{1}{2}x_i^2 - x_i(az_i(x) + b_i + u_i), \tag{2.3}$$

where u_i denotes the amount of intervention the regulator performs regarding the cost function of player i. Furthermore, in the case of strategic complements, if the regulator increases the marginal returns of a player i, this will have a positive spillover effect on its neighbours, hence lowering the value of their cost functions as well. In the case of strategic substitutes, however, increasing the marginal returns will cause more costs for player i's neighbours, discouraging them from exerting action. In this research, the function for the intervention will be derived as a solution to an optimal control problem where the aggregate of the individual players' cost functions is minimized.

2.1.3 Player dynamics

Given the amount of intervention performed by the regulator, each player in the network game chooses its own strategy to minimize the individual cost function equation (2.3). In order to capture the behaviour of the players over time, we consider the following pseudo-gradient dynamics:

$$\dot{x}_i(t) = -\frac{\partial U_i}{\partial x_i}(x_i(t), z_i(x(t)), u_i(t)), \quad \forall i \in \mathcal{I}, \tag{2.4}$$

where it should be noticed that the sign before the partial derivative over U_i is negative due to the convex nature of the cost function (De Persis and Grammatico 2019). Subsequently, by including the definition of the network aggregate from equation (2.1), together with the fact that $P_{ii} = 0$, the dynamics above can be rewritten as:

$$\dot{x}_i(t) = -x_i(t) + a\sum_{j\in\mathcal{I}} P_{ij}x_j(t) + b_i + u_i(t). \tag{2.5}$$

Ultimately, the set of strategies where each player plays their best response to the other players' strategies is called the *Nash equilibrium*, thus implying that no player has a profitable deviation from the current strategy for all $x_i \in \mathcal{X}_i$. In the case where the regulator does not intervene, i.e., $u_i(\cdot) \equiv 0$, the equilibrium of the dynamics equation (2.5) coincides with finding the action profile $\bar{x}$ satisfying:

$$\bar{x}_i \in \arg\ \min_{y\in\mathbb{R}} U_i(y, z_i(x), 0), \quad \forall i \in \mathcal{I}, \tag{2.6}$$

(Galeotti, Golub, and Goyal 2020). Then after rewriting, the equilibrium in the network game where the regulator does not intervene and the players thus choose selfishly becomes:

$$\bar{x} = (I - aP)^{-1}b. \tag{2.7}$$

In the following section, the method of deriving the optimal intervention policy will be introduced, helping the regulator to maximize social welfare.

2.1.4 The social welfare problem

For this research, the network game model introduced in section 2.1.1 is considered from the regulator's point of view, implying that we take into account the sum of the individual cost functions (equation (2.3)). Hence, we get the cost function

$$U(x(t), u(t)) = \frac{1}{2}x(t)^T x(t) - x(t)^T(aPx(t) + b + u(t)), \quad t \in [0, T], \tag{2.8}$$

where $x = \text{col}(x_i)$, $u = \text{col}(u_i)$, $\forall i \in \mathcal{I}$, and the fact that $P_{ii} = 0$ for all i allows us to write $z(x) = Px$. Accordingly, the differential equation describing the action profile of the players is derived by taking the sum of equation (2.5) and therefore given by:

$$\dot{x}(t) = (-I + aP)x(t) + b + u(t), \quad x(0) = x_0, \quad t \in [0, T], \tag{2.9}$$

where $x(t) = \text{col}(x_i(t))$, $b = \text{col}(b_i)$, and $u(t) = \text{col}(u_i(t))$.

In section 2.1.3, we looked at the dynamics from the players' point of view, where the intervention signal $u_i(t)$ was assumed to be zero. From the regulator's point of view, however, this intervention signal has to be properly designed to achieve a social welfare solution, which is defined as:

$$x_{opt} \in \arg\ \min_{y \in \mathbb{R}^n} \sum_{i \in \mathcal{I}} W_i(y_i, z_i(y)), \tag{2.10}$$

where the expression of $W_i(y_i, z_i(y))$ is also given in equation (2.2). In Shakarami, Cherukuri, and Monshizadeh (2021), the necessary condition for the existence of a unique social optimum x_{opt} is given using the following lemma:

Lemma 1. *(Shakarami, Cherukuri, and Monshizadeh 2021, Lemma II.1) The social welfare maximization problem 2.10 has a unique solution if and only if*

$$\max_{i \in \mathcal{I}} a\lambda_i(P + P^T) < 1, \tag{2.11}$$

where λ_i denotes the i-th eigenvalue of $P + P^T$.

Resulting from the lemma above, we make the following assumption before parameterizing the model:

Assumption 1. *The adjacency matrix $P \in \mathbb{R}^{n \times n}$ and the parameter $a \in \mathbb{R}$ satisfy $\max_{i \in \mathcal{I}} a\lambda_i(P + P^T) < 1$.*

Following assumption 1, we notice that the social welfare function on the right hand side of equation (2.10) is strictly convex (since $\frac{\partial^2 U}{\partial x^2} = -I + a(P + P^T) < \mathbf{0}$), and thus the unique solution for the social welfare maximization problem is given by

$$x_{opt} = (I - a(P + P^T))^{-1} b. \tag{2.12}$$

2.2 Calculus of variations and optimal control

2.2.1 Formulation of the optimal control problem

In order to get a better understanding of how the optimal control problem is formulated, methods from the *calculus of variations* will be used. Within this framework, one tries to find the minimum of a cost function by minimizing its integral (Zwart et al. 2012). By taking the integral of the function, one can calculate the area beneath it, making this technique useful for time dependant functions, such as the actions of the players in equation (2.3), for instance. In the formulation of the optimal control problem, we consider the (nonlinear) dynamics of the system to be controlled as a *dynamic constraint*

$$\dot{x} = f(x, u), \tag{2.13}$$

with state vector $x \in \mathbb{R}^n$ and input vector $u \in \mathbb{R}^m$. Next to the dynamic constraint, in this research, we will also consider an *admissible control constraint*, representing the maximum amount of intervention that can be applied by the regulator. Moreover, the set of admissible controls is denoted by $\mathcal{U} = \{u \in \mathbb{R}^n : \ |u_i| \leq u_{max}, \forall i \in \mathcal{I}\}$, for some maximum intervention

$u_{max} > 0$. Subjected to these constraints, the cost criterion to be minimized over the inputs $u(\cdot)$ becomes

$$J(x_0, u) = S(x(T)) + \int_0^T L(x(t), u(t))dt, \tag{2.14}$$

where the function $S(x(T))$ is usually called the *end-* or *final cost* and $L(x(t), u(t))$ is called the *running cost*. Subsequently, *the optimal control problem is then to find, for a given initial condition x_0, an input function $u^*(t), t \in [0, T]$, for which the cost criterion (2.14) is minimized* (Zwart et al. 2012).

In order to solve such constrained optimization problems, a commonly used technique is to introduce *Lagrangian multipliers* to derive the necessary conditions for the optimal solution. Using Pontryagin's minimum principle (PMP) (Zwart et al. 2012), these conditions for the optimal control $u^*(t)$ can be derived analytically by evaluating the Hamiltonian of the running cost function shown in equation (2.14) and the dynamic constraint given in equation (2.13). Explicitly, the Hamiltonian function is given as

$$H(x, \lambda, u) = \lambda^T f(x, u) + L(x, u), \tag{2.15}$$

where $\lambda(t) \in \mathbb{R}^n$ plays the role of the Lagrangian multiplier, also called the *adjoint variable*, next to the state variable $x \in \mathbb{R}^n$. Accordingly, based on PMP, the following theorem is used to derive the first order necessary conditions for solving the optimal control problem.

Theorem 2. *(Zwart et al. 2012, Theorem 27) Suppose $u^*(\cdot) \in \mathcal{U}$ is a solution of the optimal control problem 2.14, and $x^*(\cdot)$ the resulting optimal state trajectory. Then there exists a function $\lambda^* : [0, T] \to \mathbb{R}^n$ such that*

$$\dot{x}^* = \frac{\partial H}{\partial \lambda}(x^*, \lambda^*, u^*)^T, \quad x^*(0) = x_0, \tag{2.16a}$$

$$\dot{\lambda}^* = -\frac{\partial H}{\partial x}(x^*, \lambda^*, u^*)^T, \quad \lambda(T)^* = \frac{\partial S}{\partial x}(x^*(T))^T, \tag{2.16b}$$

$$\frac{\partial H}{\partial u}(x^*, \lambda^*, u^*)^T = 0 \tag{2.16c}$$

and

$$H(x^*, \lambda^*, u^*) = \min_{v \in \mathcal{U}} H(x^*, \lambda^*, v), \quad \forall t \in [0, T], \forall v \in \mathcal{U}. \tag{2.17}$$

Hence, with equation (2.17), we can derive the expression for the optimal control $u^*(t)$ that is within the set of admissible controls $\mathcal{U}$. Additionally, it should be noted that when there are no bounds on the control, solving equation (2.16c) should suffice for deriving an expression for $u^*(t)$. Furthermore, by solving the equations for the dynamics of the state (2.16a) and co-state (2.16b) variables, and including the initial condition $x^*(0)$ and *transversality condition* $\lambda^*(T)$, numerical results can be obtained for a particular controlled network.

2.2.2 Penalization of constraints

In the optimal control problem formulated in section 2.2, it can be seen that the set of admissible controls $\mathcal{U}$ is formulated as a constraint to which the cost function is subjected, thus making it a *hard constraint*. In order to make the control constraint less strict, it is added to the cost function as a penalty function. By formulating the constraint as a *soft constraint*, the problem becomes practically unconstrained and can be solved more easily (Gao, Zhang, and Wang 2014). In the case we have a constrained optimization function in the form

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad g_i(x) \leq \mathcal{C}, \quad \forall i \in \mathcal{I}, \tag{2.18}$$

where $\mathcal{C} \in \mathbb{R}$ is some constant. We can rewrite the problem into an unconstrained optimization problem in the form

$$\min_{x \in \mathbb{R}^n} \Phi_\alpha = f(x) + \alpha ||g(x)||_p, \tag{2.19}$$

where $\alpha \geq 0$ is called the *penalty parameter* which should be sufficiently large to make the penalized problem equivalent to the original problem (Dolgopolik 2020). Moreover, in equation (2.19), the p-norm $|| \cdot ||_p$ is used to take into account the constraint $g_i(x) \leq \mathcal{C}$ for all i in the objective function function.

Using this method, we can substitute $g(x)$ for $u(t)$ to remove the set of admissible controls as a hard constraint and transform it to a soft constraint by adding a penalty term to the objective function. Hence, in our design, we will add the penalty term $\alpha ||u(t)||_2^2$ to the optimization function 2.3, where the norm term is squared for computational convenience.

2.3 Forward-backward sweep method

In order to obtain numerical results from the optimal control problem having the dynamical equations (2.16a) and (2.16b), the indirect method is applied which is otherwise known as the *Forward-backward sweep method*(FBSM) (Rodrigues, Monteiro, and Torres 2014; Lenhart and Workman 2007). In this algorithm, first an initial guess is made on the control variable $u(t)$ for all $t \in [0, T]$, for which $u \equiv 0$ is almost always sufficient. Secondly, the state equations (2.16a) are solved forward in time, while the co-state equations (2.16b) are solved backward in time using the initial conditions x_0 and $\lambda(T)$, respectively. In MATLAB®, this will be done by applying the Runge-Kutta 4 (RK4) routine. This method, given a step-size h, solves the initial value problem $\dot{x} = f(t, x)$ by first defining four slopes at a time $t_n \in [0, T]$:

$$\begin{aligned} k_1 &= f\left(t_n, x_{t_n}\right) \\ k_2 &= f\left(t_n + \frac{h}{2}, x_{t_n} + h\frac{k_1}{2}\right) \\ k_3 &= f\left(t_n + \frac{h}{2}, x_{t_n} + h\frac{k_2}{2}\right) \\ k_4 &= f\left(t_n + h, x_{t_n} + hk_3\right), \end{aligned} \tag{2.20}$$

and then approximating the next value with

$$x_{t_{n+1}} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \tag{2.21}$$

Since in calculating k_2 and k_3 the value of $u(t_n + h/2)$ is not yet assigned, this value is approximated with the average of $u(t_n)$ and $u(t_{n+1})$. Furthermore, for the backward calculation of $\dot{\lambda}(t)$, we perform the same routine but here, we start at time T with initial condition $\lambda(T)$ and end at time 0.

In the following step of the FBSM algorithm, the control variable is updated using an update policy where the obtained values of $x(t)$ and $\lambda(t)$ are plugged in the characterization of the optimal control, which was found using PMP and will be denoted by u_{new}. Although a common update policy is to take the average of u_{new} and u_{old}(i.e., the $u(t)$ calculated in the previous iteration), in this paper we will make use of the policy

$$u_{update} = u_{new} \cdot (1 - \omega_i) + u_{old} \cdot \omega_i, \tag{2.22}$$

where i is the current iteration and $0 < \omega_i < 1$ is the weight that is placed on the previous control input.

In the final step of the FBSM algorithm, the convergence of the solution is checked by verifying if the error has become sufficiently small using the equation

$$\frac{||u_{update} - u_{old}||}{||u_{update}||} \leq \delta, \tag{2.23}$$

with δ being the accepted tolerance. As long as the difference between u_{update} and u_{old} is not within the accepted tolerance, the FBSM algorithm will keep iterating and update the control $u(t)$ at each iteration. In this research, the convergence of x and λ will also be set as a requirement and is checked in the same manner as in equation (2.23). To summarize, the steps of the algorithm are also depicted in figure 2.1.

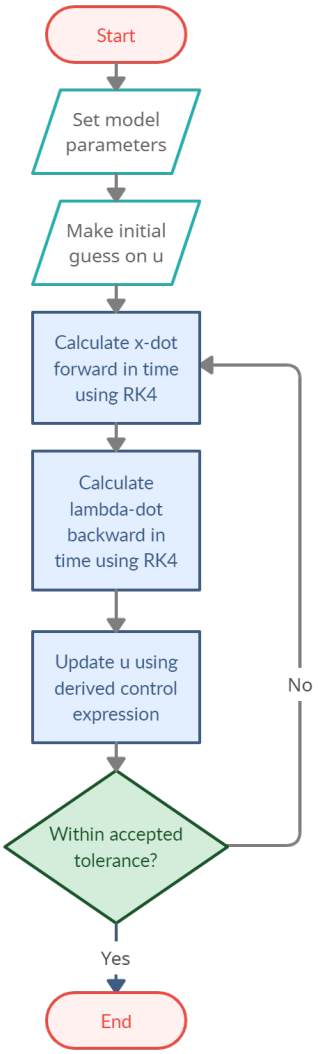


Figure 2.1: Flowchart showing the steps of the FBSM algorithm

Chapter 3

Optimal control design

3.1 Controller design

In section 2.2, we have introduced the calculus of variations and optimal control theory as methods for deriving an intervention policy for the network game model where control constraints are taken into account. In this section, we will apply this theory and propose expressions for the optimal intervention policy for the control problems taking into consideration either admissible control constraints or a penalization on the amount of control effort.

3.1.1 Constrained optimal control

In the optimal control problem, the cost function equation (2.8) is integrated over the time-interval $[0, T]$ since the model deals with continuous strategies. Fort the optimal control problem, this cost criterion has to be minimized for all $t \in [0, T]$. Then, initially, for the boundary conditions we take an arbitrary value for $x(0)$, namely x_0. Furthermore, the problem will also be constrained by the dynamics equation (2.9) and the control bounds $u \in \mathcal{U}$, where $\mathcal{U} = \{u \in \mathbb{R}^n : \ |u_i| \leq u_{max}, \forall i \in \mathcal{I}\}$. With that, we formulate the optimal control problem to be solved:

$$\min_{u} \quad J(x_0, u) = \int_0^T \left(\frac{1}{2} x(t)^T x(t) - x(t)^T (aPx(t) + b + u(t)) \right) dt \tag{3.1a}$$

$$\text{subject to} \quad \dot{x}(t) = (-I + aP)x(t) + b + u(t), \tag{3.1b}$$

$$x(0) = x_0, \tag{3.1c}$$

$$u(t) \in \mathcal{U}, \tag{3.1d}$$

$$t \in [0, T]. \tag{3.1e}$$

What can be noticed from equation (3.1a), is that there is no final cost function $S(x(T))$ included.

Proposition 2.1. *Consider the optimal control problem given in 3.1. Then, by applying Pontryagin's Minimum Principle, the expression for the optimal controller $u^*(t)$ is given by $u_i^*(t) = -u_{max} \operatorname{sign}(\psi_i(t))$.*

Proof. From equation (3.1), we define the associated Hamiltonian to be:

$$H(x, \lambda, u) = \lambda^T \left((-I + aP)x + b + u \right) + \frac{1}{2} x^T x - x^T (aPx + b + u), \tag{3.2}$$

where $\lambda : [0, T] \to \mathbb{R}^n$ plays the role of the Lagrangian multiplier. Subsequently, by applying theorem 2, we can say that if $u^*(t) \in \mathcal{U}$ is an optimal control, and $x^*(t)$ is the resulting optimal

state trajectory, then there exists a function $\lambda^* : [0, T] \to \mathbb{R}^n$ such that:

$$\dot{x}^* = \frac{\partial H}{\partial \lambda}(x^*, \lambda^*, u^*)^T = (-I + aP)x^* + b + u^*, \quad x^*(0) = x_0, \tag{3.3a}$$

$$\dot{\lambda}^* = -\frac{\partial H}{\partial x}(x^*, \lambda^*, u^*)^T = (I - aP^T)\lambda^* - x^* + a(P + P^T)x^* + b + u^*, \tag{3.3b}$$

$$\lambda^*(T) = \frac{\partial S}{\partial x}(x^*(T))^T = 0. \tag{3.3c}$$

Notice that the condition $\frac{\partial H}{\partial u} = 0$ is not included in the necessary conditions above because we are dealing with a restricted set $\mathcal{U}$. Therefore, this condition is replaced by a true minimization condition obtained from Pontryagin's Minimum Principle:

$$H(x^*(t), \lambda^*(t), u^*(t)) = \min_{v \in \mathcal{U}} H(x^*(t), \lambda^*(t), v(t)), \tag{3.4}$$

for which filling in the Hamiltonian leaves us with the equality:

$$(\lambda^{*T}(t) - x^{*T}(t))u^*(t) = \min_{v \in \mathcal{U}}(\lambda^{*T}(t) - x^{*T}(t))v(t), \tag{3.5}$$

for all $t \in [0, T]$ and all $i \in \mathcal{I}$. Then, from equation (3.5), we identify the switching function $\psi_i(t) = \lambda_i^*(t) - x_i^*(t)$, allowing us to rewrite the minimization of the Hamiltonian into:

$$u_i^*(t) = \begin{cases} u_{max} & \text{if } \psi_i(t) < 0, \\ 0 & \text{if } \psi_i(t) = 0, \\ -u_{max} & \text{if } \psi_i(t) > 0, \end{cases} \tag{3.6}$$

for all $i \in \mathcal{I}$. Subsequently, the expression for the optimal control solution can be rewritten as

$$u_i^*(t) = -u_{max}\,\text{sign}(\psi_i(t)), \quad \forall i \in \mathcal{I}. \tag{3.7}$$

□

By numerically solving the first-order necessary conditions equations (3.3a) to (3.3c) and (3.7) with the FBSM algorithm introduced in section 2.3, the optimal state trajectory $x^*(t)$ that minimizes the cost criterion 3.1a is obtained.

3.1.2 Penalized optimal control

From the above result, we notice that the optimal control expression 3.7 generates a discontinuous signal. In other words, the amount of intervention employed by the regulator is either 0% or 100%. Hence, the hard constraint on the admissible control set has made this intervention policy unusable for applications where the amount of intervention that is applied has to be somewhere in between the upper and lower bound of the admissible control set. To remove the strict dependence of the optimal control problem on the set of admissible controls $\mathcal{U}$, a penalty term is added to the optimal control problem shown in equation (3.1), creating the following modified optimal control problem:

$$\min_u \quad \hat{J}(x_0, u) = \int_0^T \left(\frac{1}{2}x(t)^T x(t) - x(t)^T(aPx(t) + b + u(t)) + \underbrace{\alpha||u(t)||_2^2}_{\text{penalty term}} \right) dt \tag{3.8a}$$

$$\text{subject to} \quad \dot{x}(t) = (-I + aP)x(t) + b + u(t), \tag{3.8b}$$

$$x(0) = x_0, \tag{3.8c}$$

$$t \in [0, T]. \tag{3.8d}$$

Proposition 2.2. *Consider the optimal control problem given in 3.1. Then, by applying Pontryagin's Minimum Principle, the expression for the optimal controller $u^*(t)$ is given by $u^*(t) = \frac{x^*(t)-\lambda^*(t)}{2\alpha}$.*

Proof. For the optimal control problem equation (3.8), we write the following Hamiltonian:

$$H(x, \lambda, u) = \lambda^T \left((-I + aP)x + b + u\right) + \frac{1}{2}x^T x - x^T(aPx + b + u) + \alpha||u||_2^2. \tag{3.9}$$

Again, using theorem 2, the first-order necessary conditions for the existence of the optimal controller can be derived:

$$\dot{x}^* = \frac{\partial H}{\partial \lambda}\left(x^*, \lambda^*, u^*\right)^T = (-I + aP)x^* + b + u^*, \quad x^*(0) = x_0, \tag{3.10a}$$

$$\dot{\lambda}^* = -\frac{\partial H}{\partial x}\left(x^*, \lambda^*, u^*\right)^T = (I - aP^T)\lambda^* - x^* + a(P + P^T)x^* + b + u^*, \tag{3.10b}$$

$$\lambda^*(T) = \frac{\partial S}{\partial x}\left(x^*(T)\right)^T = 0, \tag{3.10c}$$

$$\frac{\partial H}{\partial u}\left(x^*, \lambda^*, u^*\right)^T = \lambda^* - x^* + 2\alpha u^* = 0. \tag{3.10d}$$

Now, since the bounds $\mathcal{U}$ on the control are dropped, condition 3.10d becomes relevant. From this equality, we can derive an expression for $u^*(t)$, implying that we do not have to use the equation for the Minimum Principle, as was done in the constrained problem. Hence, the expression for optimal controller becomes:

$$u^*(t) = \frac{x^*(t) - \lambda^*(t)}{2\alpha} \tag{3.11}$$

□

As can be seen, we have now obtained a continuous function for the control input, rather than the discontinuous control function derived in equation (3.7).

3.2 Convergence to the social welfare solution

Although the expressions for the optimal controllers and the necessary conditions that need to be fulfilled have now been defined, this is not yet a guarantee that the state $x^*(t)$ will converge to the social optimal solution formulated in equation (2.12). To do this, we change the optimal control problem defined in the previous sections by adding a fixed endpoint constraint, being:

$$x(T) = x_{opt}, \quad \forall i \in \mathcal{I}. \tag{3.12}$$

Although it seems straightforward that this alteration causes the optimal control expression to solve the social welfare problem, its complexity lies in the numerical implementation of this fixed endpoint constraint. Since x_{opt} is now the specified state that must be reached at the final time, an additional boundary value is introduced to the system and transversality condition ($\lambda^*(T)$) can no longer be zero (Sharp, Burrage, and Simpson 2021). In Lenhart and Workman (2007), a modified version of the FBSM algorithm (see also section 2.3) is introduced that helps to solve the fixed endpoint optimal control problem numerically. For this adapted FBSM algorithm, first an initial guess is made for $\hat{\lambda}^*(T) = \theta$, for some $\theta \in \mathbb{R}$. With this guess, the optimal control problem is then solved using the regular FBSM algorithm. Having obtained the solution of this optimal control problem, the goal of the adapted FBSM becomes finding the value of θ where the function

$$V_i(\theta) = \hat{x}_i(T) - x_{i,opt} \tag{3.13}$$

is equal to zero, for all i. In this way, an outer iterative method is required for finding the roots of $V(\theta)$ and updating the guess θ. The method used for finding these roots is the secant method (Papakonstantinou and Tapia 2013). With this method, the approximating formula

$$x_1 - x_2 = \frac{\theta_2 - \theta_1}{V(\theta_2) - V(\theta_1)} V(\theta_1) \tag{3.14}$$

is used to alter the guess θ by simply subtracting this finite difference approximation $(x_1 - x_2)$. As can be seen, this approximating formula requires two guesses to be made upfront, i.e., θ_1 and θ_2. From these two guesses, the guess θ resulting in the $V(\theta)$ that is the closest to zero is set to θ_1. This guess is then updated through subtracting the finite difference approximation 3.14, for which we then again calculate the value of $V(\theta)$. In this way, the sequence is continued until we find a $V(\theta)$ that is defined to be "close" to zero, i.e., $|V| < \epsilon$, for some small error tolerance ϵ. Hence, by finding the value for θ that is the root for $V(\theta) = 0$, we have found the value for the transversality condition $\lambda^*(T)$ that steers the state trajectory towards the solution of the social welfare problem x_{opt}. Moreover, the roots for $V_i(\theta)$ can be calculated simultaneously for all players. The reason for this is that other than in Lenhart and Workman (2007), each player i considers a dynamic function of the same form, only having different values for the parameters (see also equation (2.16a)). In figure 3.1, a schematic overview is given of the steps involved in the calculation of the roots of $V(\theta)$ using this adapted FBSM algorithm.

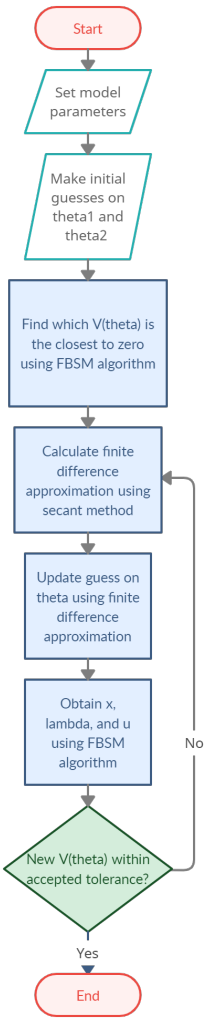


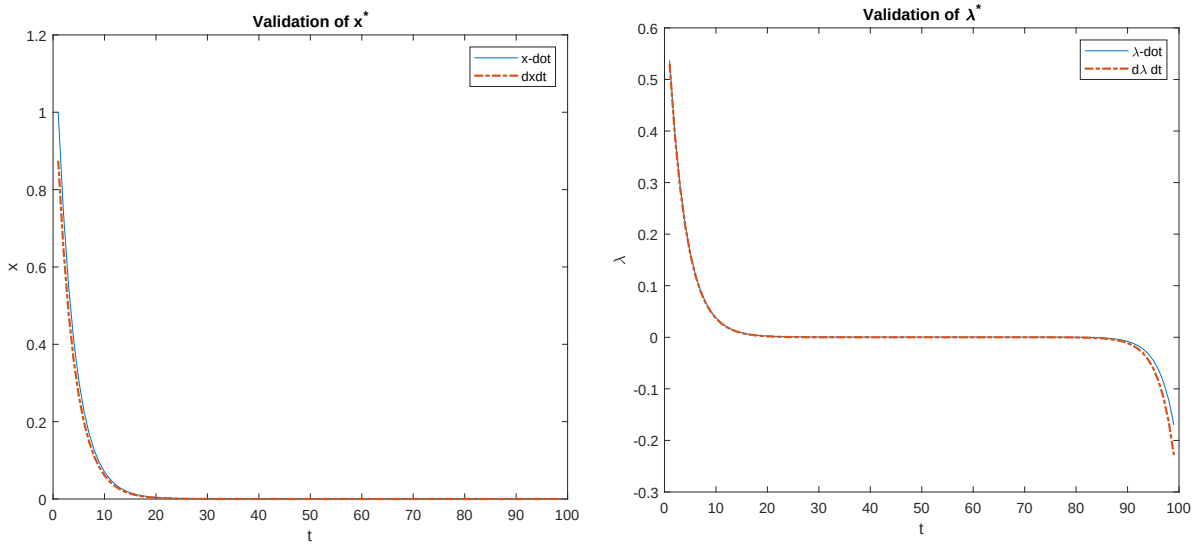
Figure 3.1: Flowchart showing the steps of the adapted FBSM algorithm

3.3 Validation of the controller design in MATLAB®

To validate whether the outcome of the numerical calculations to be performed using the FBSM algorithm are correct, the implementation of the optimal controller in MATLAB® is checked. As previously described in section 2.3, the FBSM algorithm will be used to solve the first-order necessary conditions $\dot{x}(t)$ and $\dot{\lambda}(t)$, and updating the control function according to an updated policy 2.22. For this update policy, we will use the value $\omega = 0.9$, implying that, at each iteration, we will take the average of the control output of the previous iteration and the newly calculated control output to update the controller. Furthermore, an error tolerance of $\delta = 0.001$ is found to be sufficiently small for the convergence of the FBSM algorithm (see equation (2.23)), and will therefore be used throughout this research. To check whether the obtained time series $x^*(t)$ and $\lambda^*(t)$ has been correctly calculated by the numerical algorithm, they are plugged back into their corresponding dynamic functions $\dot{x}^*(t)$ and $\dot{\lambda}^*(t)$, respectively. The resulting time series are then compared to the approximations of the derivatives of $x^*(t)$ and $\lambda^*(t)$, being calculated by:

$$\dot{\hat{x}}^*(t) = \frac{x^*_{t+1} - x^*_t}{dt}, \quad \dot{\hat{\lambda}}^*(t) = \frac{\lambda^*_{t+1} - \lambda^*_t}{dt}, \tag{3.15}$$

respectively, and for all $t \in [0, T]$. In the case that the obtained time series for $\dot{x}^*(t)$ and $\dot{\lambda}^*(t)$ match those of $\dot{\hat{x}}^*(t)$ and $\dot{\hat{\lambda}}^*(t)$, while also taking into account a tolerance δ, it can be said that the numerical results are valid. To demonstrate, we look at the results of the optimal controllers derived in this chapter. In figure 3.2, the comparisons between the dynamics and approximated time derivatives of the state and co-state are given for the controller from equation (3.7). Here, the applied parameters for the network model correspond with those defined in Shakarami, Cherukuri, and Monshizadeh (2021), and will be discussed in more detail in chapter 4. Furthermore, the bound on the controller u_{max} here is set to 1. As can be seen from both figures, the results of the dynamic functions and the approximations of the time derivatives are roughly the same, for which it can be said that the obtained time series of $x^*(t)$ and $\lambda^*(t)$ are both valid.



(a) $\dot{x}^*$ and $\dot{\hat{x}}^*$ **(b)** $\dot{\lambda}^*$ and $\dot{\hat{\lambda}}^*$

Figure 3.2: Action profiles of the dynamics of the state and co-state variables for 100 time-steps under constrained optimal control.

Chapter 4

Numerical simulations

In this chapter, the model and designs described in the previous chapters will be implemented in MATLAB® for numerical simulations to answer the research questions formulated in section 1.3.1.

4.1 Model parameters

As already shortly introduced in the previous chapter, this research will, initially, make use of the same parameters as those used in Shakarami, Cherukuri, and Monshizadeh (2021). Therefore, we will consider a population $\mathcal{I} = \{1, ..., 6\}$ players interacting over the weighted directed graph shown in figure 4.1. Here, the width of each link indicates the weight on each link and thus the value of P_{ij} in the adjacency matrix P, where player i is the one receiving the influence of player j. Then, the number next to each node represents the value of the player's standalone marginal return b_i. Furthermore, this network game will be considered as a game of strategic substitutes, where $a = -0.2$. For the initial condition x_0, we will assign a random value within the interval $[0, 1]$, similarly to what was done in Shakarami, Cherukuri, and Monshizadeh (2021). Additionally, we will also make use of the same initial and final time as in this paper. That is, the initial time will be 0, and the final time T will be 25. For the reader's convenience, we will express this time in seconds.

Given these parameters, we can already conclude that there should exist a unique solution to the social welfare maximization problem, as the condition given in lemma 1 is satisfied. Hence, we also can find the values of the social optimal states x_{opt}, which are found by filling in the given model parameters into equation (2.12).

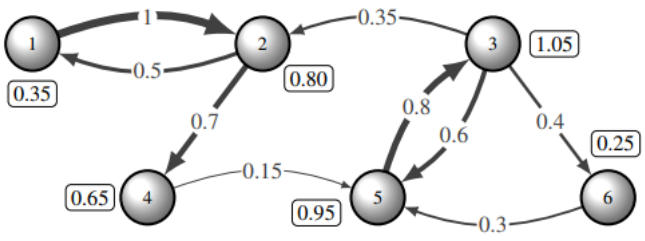


Figure 4.1: Directed network which was also used in Shakarami, Cherukuri, and Monshizadeh (2021).

4.2 Numerical experiments

In order to test the performance of the obtained necessary conditions for the optimal controllers derived in chapter 3, four experiments have been set up. The first experiment here is used to obtain a baseline result which is then used to analyse the behaviour of the different controllers in the case of a free endpoint $x(T)$. Then, in the second experiment the final time condition $x(T) = x_{opt}$ is added to ensure convergence to the social optimal solution. Simultaneously, this experiment is also used to investigate the behaviour of the penalized optimal control for increasing values of the penalty parameter α. Subsequently, in the third experiment, the numerical results for the optimal controllers will be compared to those obtained in Shakarami, Cherukuri, and Monshizadeh (2021) to evaluate whether the policy obtained our research can indeed be referred to as *optimal* compared to other existing policies. Finally, a fourth experiment is conducted to examine the effect of changing the network on the behaviour of the controllers and therefore analyse the robustness of the optimal intervention policies. That said, the remainder of this section will be dedicated to elaborating on each experiment to be performed in more detail and also providing their numerical results.

4.2.1 Experiment 1: Free endpoint optimal control

Experimental setup

In the first experiment, all controllers derived in chapter 3 will be tested without having the constraint on the final time that enforces the social welfare solution. Hence, the regular FBSM algorithm which was described in section 2.3 will be applied, where we set the transversality condition equal to zero, i.e., $\lambda_i(T) = 0$ for all $i \in \mathcal{I}$. In this way, we can analyse and compare the behaviour of the constrained and the penalized controllers and the solutions to which they converge. Furthermore, for the constrained optimal controller (see equation (3.7)), the boundary of the admissible control set $\mathcal{U}$ will be set to $u_{max} = 1$. For the penalized controller, we will choose the penalty parameters α having the values 1, 3, and 5.

Next to the solutions of the optimal control problems, also the results of implementing a static *open-loop* intervention will be shown for reference. As also described in Shakarami, Cherukuri, and Monshizadeh (2021), this intervention is a constant signal where the regulator possesses full knowledge of the game, namely the pair (aP, b), and therefore also the value of x_{opt}. With that, the control signal for this open-loop intervention becomes $u(t) \equiv (I - aP)x_{opt} - b$.

Simulation results

In figure 4.2, the results of solving the free endpoint constrained optimal control (see section 3.1.1) using regular FBSM are given for different sizes of the controller bounds u_{max}. The first thing to be noticed from these plots is that, for every player, the constrained optimal controller produces a constant control input signal that is equal to the value of u_{max} and thus does not change to 0 or $-u_{max}$. When comparing input signals to the trajectories shown in the upper plots of the states and co-states, we see that this behaviour of the control signal can be explained by the values of the states being higher than those of the co-states at all times. Next to that, we see that after approximately three time units, the input signal causes the state variables of the individual players to converge to constant values. Furthermore, figures 4.2a to 4.2c also show that the value to which the state converges also increases when allowing for higher values of the controller in the set $\mathcal{U}$, which is logical given the linear relationship between the dynamics $\dot{x}(t)$ and the control input $u(t)$.

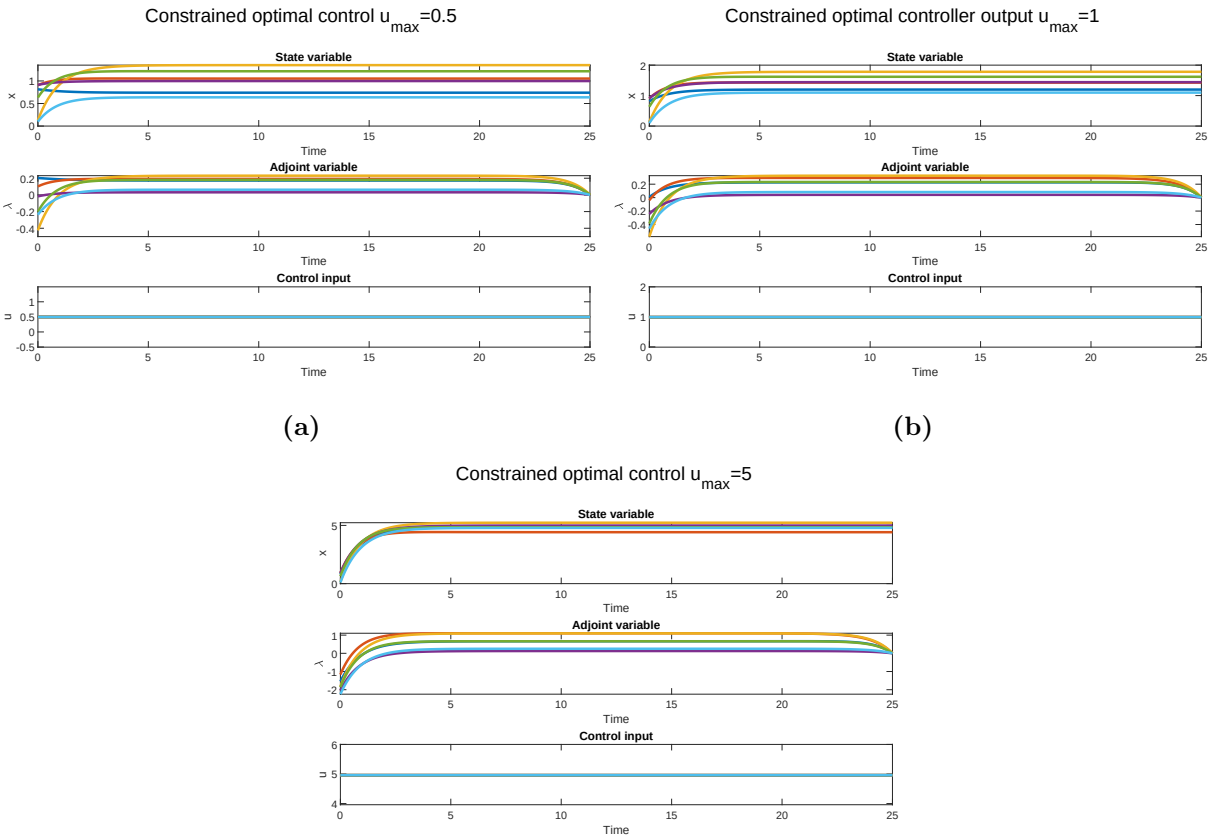
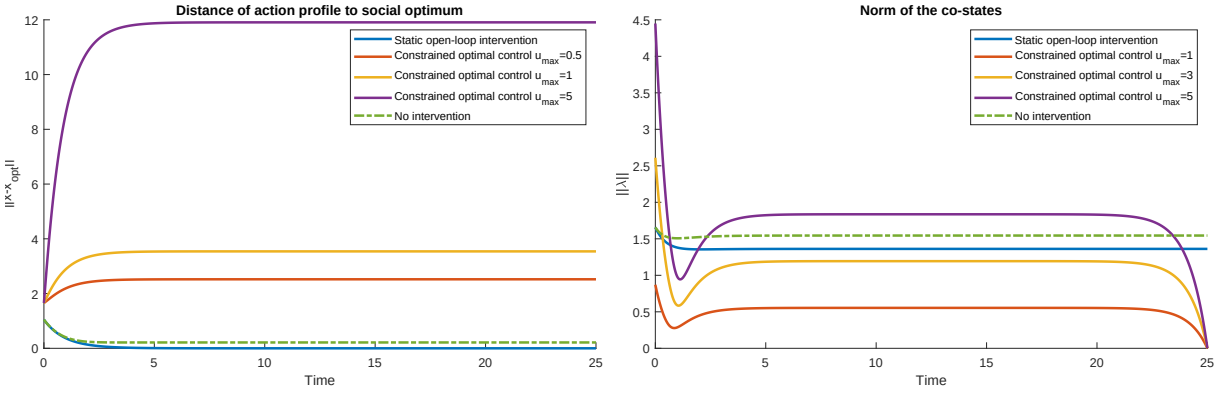


Figure 4.2: Solutions of the constrained optimal control problem with controller bounds 0.5, 1, and 5.

If we then plot the distance of the state trajectories to the social optimal state, as is done in figure 4.3a, we see that the states of the constrained optimal controllers steer away from the desired state. Furthermore, we observe that for larger values of u_{max}, the deviation from the social optimal state becomes larger. For the trajectory of the co-state variable λ, we observe in figure 4.3b that similar behaviour is obtained when comparing the different values of u_{max}.



(a) **(b)**

Figure 4.3: Results of the numerical simulations of the free endpoint constrained optimal control problem for three different values of u_{max}.

Next, in figures 4.4a to 4.4c, the outputs of the simulations using the penalized controllers are given, obtained by solving the optimal control problem formulated in section 3.1.2. Compared to the constrained controllers, we see that the values of the control inputs u now vary over time and with respect to each other, rather than all taking the same constant value. Furthermore, for the chosen values of α, the control inputs all converge to a constant value after 3 seconds, after which they deviate from this value at around 24 seconds. This behaviour is also similar to what can be seen in the plots of the adjoint variables, where the backward calculation starts from $\lambda = 0$ and then moves towards a steady-state value.

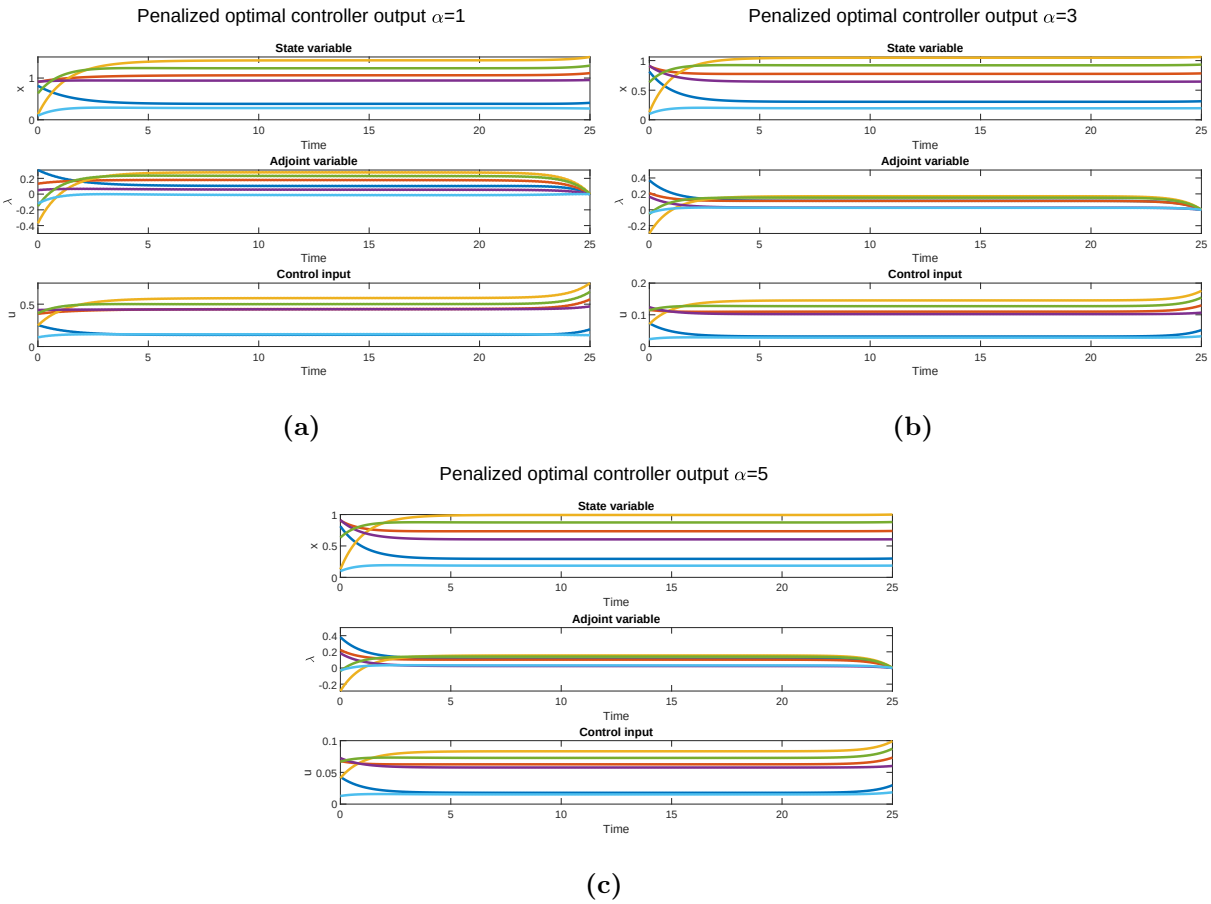
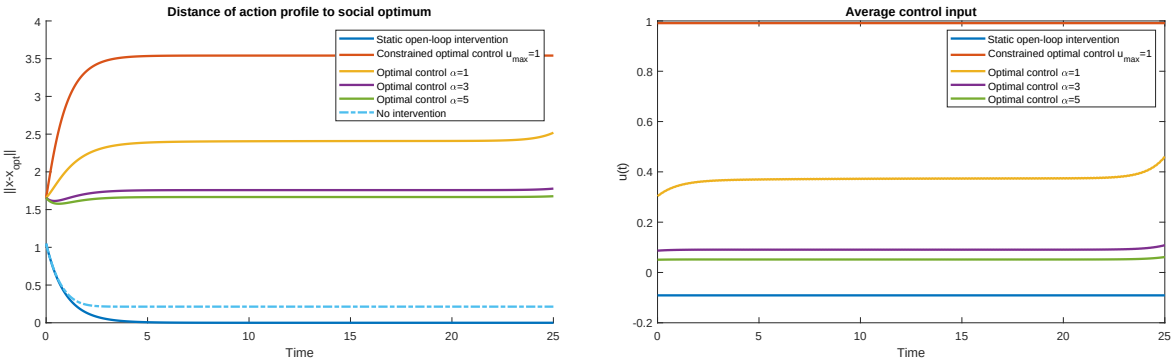


Figure 4.4: Solutions of the penalized optimal control problem with penalty parameters 1, 3, and 5.

In figure 4.5a, the solutions of the penalized optimal control problems are shown together with those of the constrained optimal controller having $u_{max} = 1$. Similar to figure 4.3a, the norm of the difference between each player's trajectory and the social optimal solution is taken. When comparing the two different types of controllers, we see that the solution of the penalized optimal controllers also converge to a constant steady-state value after around 3 seconds but changes course in the last seconds of the time series. What becomes clear from this plot, however, is that it seems that for increasing values of α, the amount of deviation from the steady-state value decreases. Next to that, The trajectory also becomes more similar to that of the solution where the intervention u is equal to zero. Inherently, this makes sense since, for higher values of α, the punishment on the intervention becomes more and more fierce, thus further demotivating the regulator of performing any interventions in the network.



(a) **(b)**

Figure 4.5: Results of the numerical simulations of the free endpoint optimal control problems, including the solutions of both the constrained and the penalized controllers.

4.2.2 Experiment 2: Reaching the social welfare solution

Experimental setup

In the second experiment, the final time constraint $x(T) = x_{opt}$ will be added to the optimal control problem such that the solution is guaranteed to converge to the social welfare solution. Moreover, in this experiment only the penalized optimal controller is taken into consideration for including the final time constraint $x(T) = x_{opt}$ since the discontinuous nature of the constrained optimal control signal does not allow the implementation of the adapted FBSM algorithm. Therefore, the optimal control problem formulated in section 3.1.2 is slightly rewritten:

$$\min_{u} \quad \hat{J}(x_0, u) = \int_0^T \left(\frac{1}{2} x(t)^T x(t) - x(t)^T (aPx(t) + b + u(t)) + \alpha ||u(t)||_2^2 \right) dt \tag{4.1a}$$

$$\text{subject to} \quad \dot{x}(t) = (-I + aP)x(t) + b + u(t), \tag{4.1b}$$

$$x(0) = x_0, \tag{4.1c}$$

$$x(T) = x_{opt}, \tag{4.1d}$$

$$t \in [0, T]. \tag{4.1e}$$

Since the addition of the final time constraint cannot be handled by the regular FBSM algorithm, the adapted FBSM algorithm will be used here to calculate the corresponding values of $\lambda_i(T)$ at each $x_i(T)$, for all i (see also section 3.2). For the initial guesses on the transversality condition $\lambda(T) = \theta$, we select $\theta_1 = 10$ and $\theta_2 = -10$, for all $i \in \mathcal{I}$. These values are found to be sufficiently large to find achieve convergence to the social welfare solution with an error tolerance $\epsilon = 0.0015$. With that, this experiment will be used to check the convergence of the penalized optimal controller for different values of α, running from $\alpha = 1$ until $\alpha = 20$. In this way, we seek to find the effect of increasing the penalization on the intervention u.

Simulation results

After solving the fixed endpoint optimal control problem with the adapted FBSM, we see in figure 4.6a that the social welfare state has indeed been reached for all $\alpha \in [1, 20]$. On a side note, however, during the testing phase, also penalty parameter values lower than 1 have been tested but for these, the algorithm was not able to converge to the social optimum. That said, the second thing to be noticed in the figure is the shape of the trajectories, which all converge to a constant value up until the last seconds, where all trajectories change directions and steer towards the social welfare state. Taking then into account the norm of the trajectories of the

controller input depicted in figure 4.6d, we see that at the same time, the norms of the inputs all increase. In other words, to reach the social welfare solution, a higher amount of intervention is required compared to the steady-state situation. Furthermore, from figure 4.6c we see that the change from a decreasing value of $||u(t)||$ to an increasing value in figure 4.6d is a result of the control effort changing signs from positive to negative in the last second.

When comparing the solutions for penalty parameters varying from 1 to 20, we can see from figure 4.6a that increasing the penalty for intervention does not imply better performance in terms of convergence speed to the social optimum due to the late change of trajectory discussed in the previous paragraph. Additionally, the graph also shows that for low values of α, the distance to the social optimal state is larger than for the higher values. Furthermore, figure 4.6d shows us that for increasing values of α, the amount of control effort performed by the regulator becomes lower, which is logical since more intervention is then punished by increased costs. Accordingly, figures 4.6e and 4.6f show decreasing costs for fiercer punishments on the intervention. An important side-note to these figures is that these plots consider the costs that are incurred by the regulator, i.e., the cost function equation (2.2). Therefore, these costs do not take into account the added costs of the performed intervention but only consider the calculated action profile x^* and the other constants of the model. Moving on, another relation that is shown by the latter graph is that from a penalty parameter value of $\alpha = 10$, an even further increase of α will have little effect on the total costs incurred during the time running from 0 to 25 seconds.

This can also be seen in table 4.1, which shows the convergence results of the parameter variation on the penalty parameter α. Furthermore, this table also shows that as α increases, fewer iterations are required for both the FBSM algorithm, and the outer algorithm searching for the roots of $V(\theta)$ using the secant method.

Table 4.1: Convergence results of the numerical simulation of the penalized optimal control problem with the parameter variation on $\alpha \in [1, 20]$.

α	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Total costs	-562.47	-1038.41	-1108.93	-1134.53	-1147.35	-1154.96	-1159.96	-1163.48	-1166.10	-1168.10
Secant iterations	11	5	5	5	5	6	5	5	5	5
FBSM iterations	814	290	265	255	250	294	240	240	236	235
α	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Total costs	-1169.69	-1170.98	-1172.05	-1172.95	-1173.71	-1174.37	-1174.95	-1175.45	-1175.91	-1176.30
Secant iterations	5	5	7	3	3	3	4	4	2	2
FBSM iterations	235	235	329	141	139	138	184	184	92	92

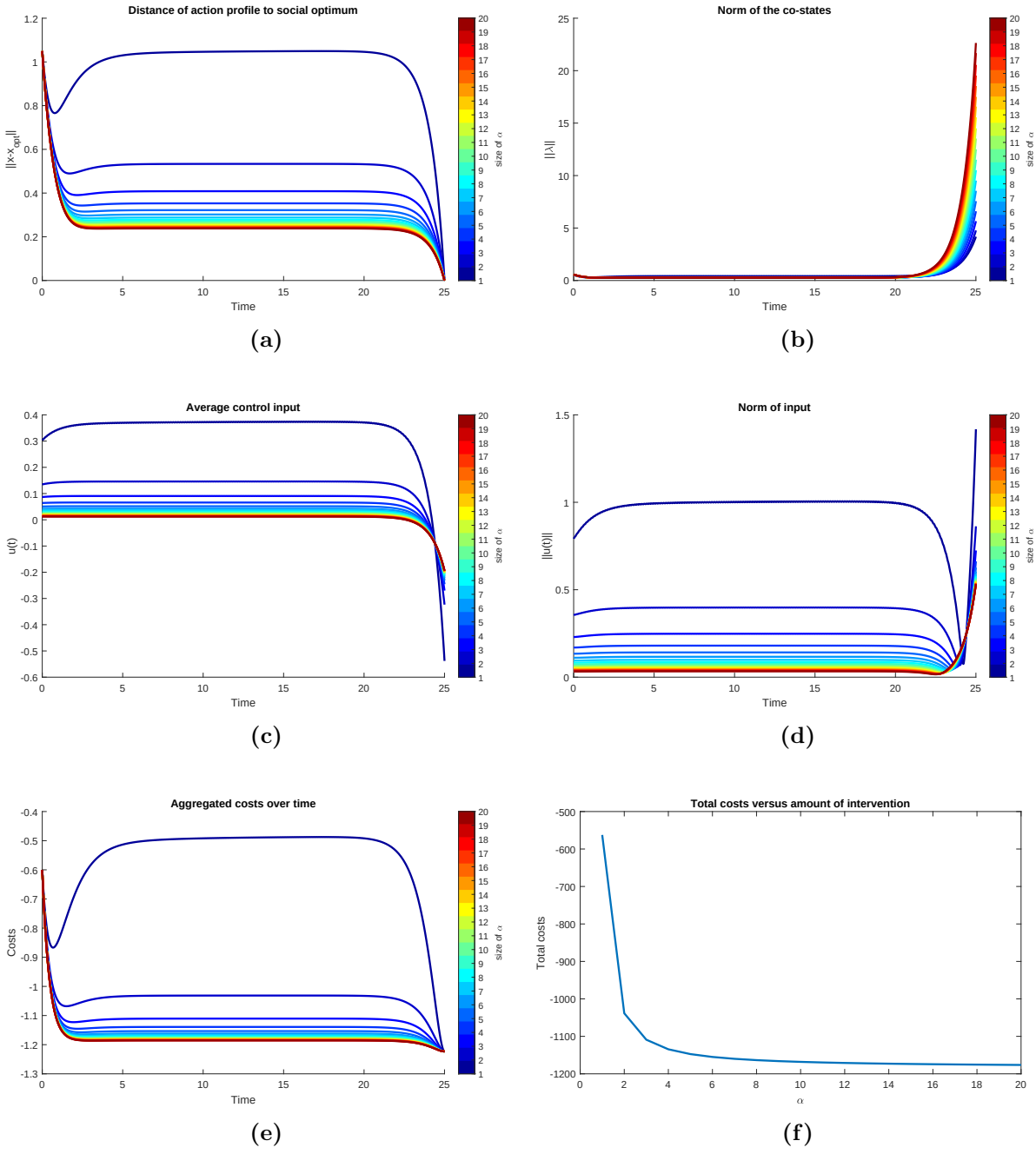


Figure 4.6: Plotted results of the parameter variation experiment on the fixed endpoint optimal control problem for $\alpha \in [1, 20]$.

4.2.3 Experiment 3: Comparison with static and dynamic interventions

Experimental setup

After testing the optimal control solutions for different intervention sizes, these solutions will be compared to two of the intervention policies described in Shakarami, Cherukuri, and Monshizadeh (2021). The policies to be considered in this paper will be the static feedback policy and the dynamic feedback policy. In the first intervention policy, the regulator has knowledge of the network structure and the impact of the players on each other, i.e., aP. With that, a feedback loop is included in the control, making up the static feedback intervention $u(t) = aP^T x(t)$.

Next to this static intervention policy, Shakarami, Cherukuri, and Monshizadeh (2021) also proposes a dynamic intervention policy that does not require the knowledge of aP but uses an estimate of the social optimum $x_s \in \mathbb{R}^n$. With that, an integral control-based intervention is proposed taking the form $\dot{u}(t) = -(x(t) - x_s)$. Similar to the paper, we assume that the regulator knows the social optimum and we set $x_s = x_{opt}$.

Then to recall, figure 4.7 depicts the results of the paper, which were obtained by using the `ode45` function of MATLAB®. Although the aim of the paper is to maximize the payoff function that is equal to $-W_i(x_i, z_i(x))$ from equation (2.2), the comparison to the optimal control to be performed in this experiment will still be valid since the numerical calculations take into account the same dynamics $\dot{x}(t)$.

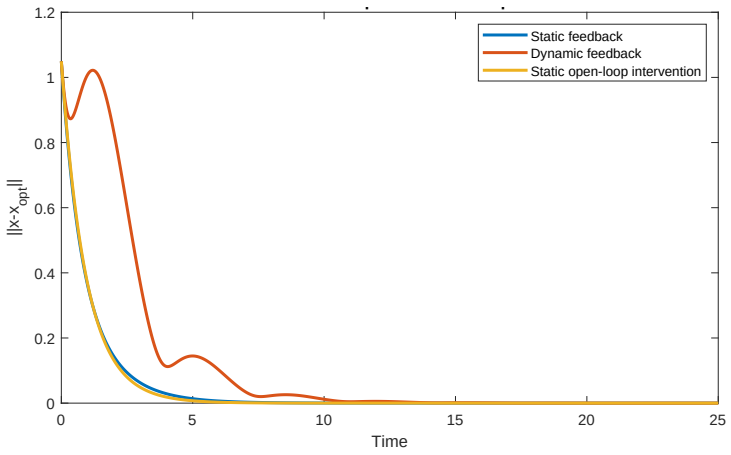


Figure 4.7: Numerical results of the intervention policies discussed in Shakarami, Cherukuri, and Monshizadeh (2021), showing the distance of the action profile to the social welfare state.

Simulation results

Now that the social welfare solution has been reached and the performance of different sizes of penalization on the intervention has been analysed, the third experiment will be dedicated to further investigating the controllers' performance in comparison to the results of Shakarami, Cherukuri, and Monshizadeh (2021). Figure 4.8 shows the numerical results of the penalized optimal intervention policies, compared with those of the static and dynamic policies. From figure 4.8a, the first thing that strikes is that both the static and dynamic intervention policies converge faster to the social welfare state x_{opt}. As a result, the costs incurred by the regulator are lower when applying the static and dynamic intervention policies, as is also shown in figures 4.8c and 4.8d. Moreover, figure 4.8d also makes it clear that even for the higher values of α, the costs are not as low as for the static intervention policy, which has a total regulator cost of -1210.86 (equal to a payoff of +1210.86) over the entire time period and is marked in the plot with a red line.

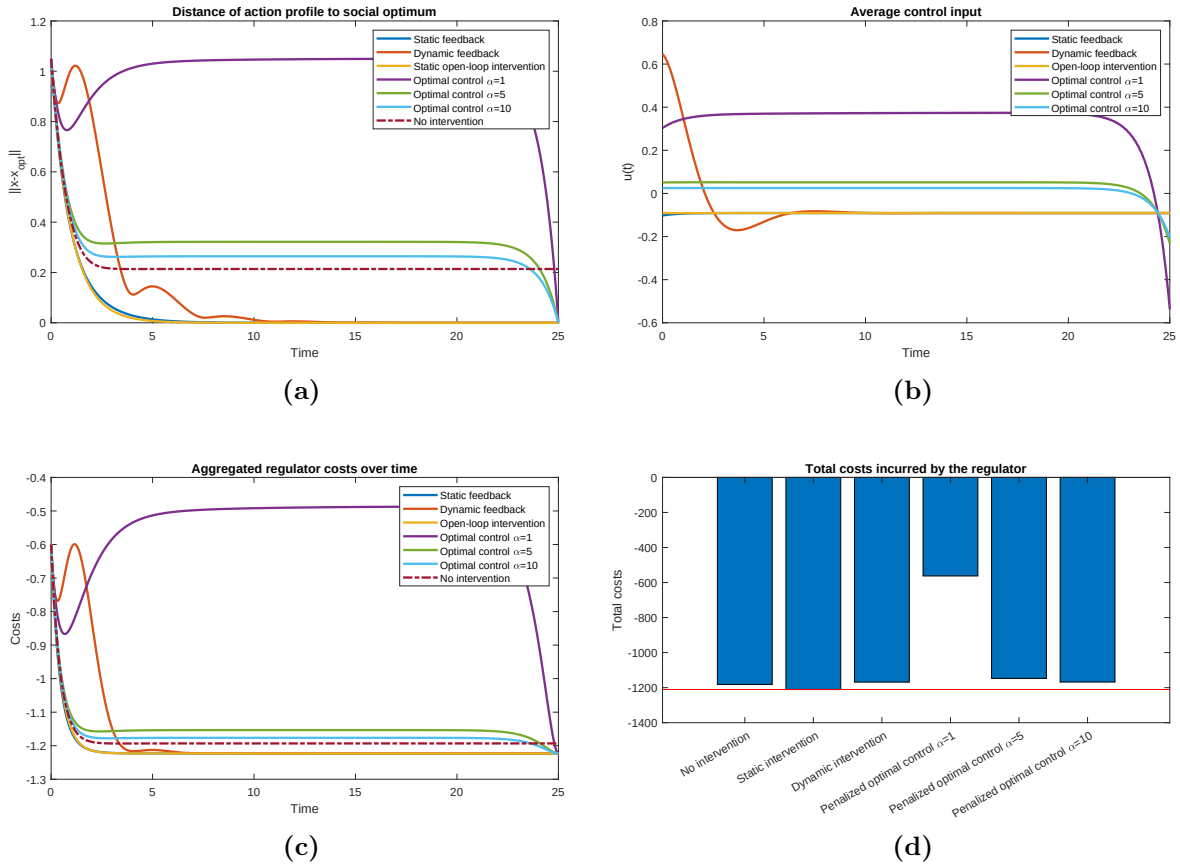


Figure 4.8: Results of the numerical simulations of the fixed endpoint optimal control problems compared with the static and dynamic intervention policies.

4.2.4 Experiment 4: Optimal control with altered cost criterion

Experimental setup

Seeing the results of experiments 2 and 3, we notice that although the proposed intervention policies achieve in steering the network towards social welfare, they cannot yet be called truly optimal. This is for the reason that compared to existing intervention policies, more time is needed for achieving the social welfare state, implying also higher costs. Therefore, we reevaluate the optimal control problem shown in equation (4.1). What stands out now after looking at the results, is that the cost functions used for deriving the optimal controller (see equations (3.1a) and (3.8a)) take into consideration the modification term $-x^T u$ that is applied by the regulator. What is then important to remark is that the addition of this modification term is also the difference between the cost function that is to be optimized by the regulator(see equation (2.2)), and the cost function as it is observed by the individual agents (see equation (2.3)). In that way, the use of the latter cost function in the formulation of the optimal control problem, therefore, implies that there already is a penalization on the control from the regulator's point of view. Moreover, in the penalized optimal control problem, adding the penalty term $\alpha||u(t)||$ also causes a double penalization on the amount of intervention performed by the regulator. As the term $u(t)$ now also appears in the dynamics of the co-state $\dot{\lambda}$, the transversality condition $\lambda(T)$ that is essential for reaching the social welfare solution, is now also dependent on the amount of intervention that is performed. With that, we can see why the trajectories of the solutions to the optimal control problems desolate their steady-state value in order to satisfy the transversality condition. Therefore, for this last experiment we propose the following modified optimal control

problem, where we remove the double dependency on the control effort $u(t)$:

$$\min_{u} \quad \hat{J}(x_0, u) = \int_0^T \left(\frac{1}{2} x(t)^T x(t) - x(t)^T (aPx(t) + b) + \alpha ||u(t)||_2^2 \right) dt \tag{4.2a}$$

$$\text{subject to} \quad \dot{x}(t) = (-I + aP)x(t) + b + u(t), \tag{4.2b}$$

$$x(0) = x_0, \tag{4.2c}$$

$$x(T) = x_{opt}, \tag{4.2d}$$

$$t \in [0, T]. \tag{4.2e}$$

As can be seen, the minimization objective has now become equal to $W(x(t), z(x(t))) + \alpha ||u(t)||_2^2$, and thus includes a single penalization on the control effort $u(t)$. Then, by applying PMP as was also done in chapter 3, we derive the following first order necessary conditions for the modified optimal control problem:

$$\dot{x}^*(t) = (-I + aP)x^*(t) + b + u^*(t), \quad x^*(0) = x_0, \tag{4.3a}$$

$$\dot{\lambda}^*(t) = (I - aP^T)\lambda^*(t) - x^*(t) + a(P + P^T)x^*(t) + b, \tag{4.3b}$$

$$\lambda^*(T) = \theta, \tag{4.3c}$$

$$u^* = -\frac{\lambda^*}{2\alpha}. \tag{4.3d}$$

The next section will provide the simulation results for the numerical calculations on the above equations using the adapted FBSM algorithm. Here, we will again use $\theta_1 = 10$ and $\theta_2 = -10$ for all $i \in \mathcal{I}$ as the first guesses on the transversality condition equation (4.3c).

Simulation results

In table 4.2 and figure 4.9, we show the results of the parameter variation on α for the modified optimal control problem 4.2. With this modification of the optimal control problem, the adapted FBSM algorithm is now able to obtain numerical results converging to the social welfare solution for smaller values of α. Therefore, the interval used for the parameter variation on the penalty parameter is now $[0.03, 20]$, with 0.03 being the smallest value for which results could be obtained. When comparing figure 4.9a to figure 4.6a, we see that the solutions of the modified optimal control problem show similar behaviour in terms of converging to a steady-state value before steering to the social optimal state in the last seconds of the time window. However, the steady-state values obtained by the solutions of the modified optimal control problem are now closer to the social optimum than in figure 4.6a, especially for the lower values of α. Furthermore, figure 4.9c also shows that the control effort is now negative during the entire time window of the simulation, other than figure 4.6c where it changes from positive to negative in the last seconds. Then, looking at the costs in figures 4.9e and 4.9f, we now see that for lower values of the penalty parameter α, fewer costs are incurred by the regulator. This is again contrary to the solutions shown in figure 4.6, where costs increased for lower values of α.

Table 4.2: Convergence results of the numerical simulation of the adjusted penalized optimal control problem with the parameter variation on $\alpha \in [0.03, 20]$.

α	**0.03**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Total costs	-1218.99	-1199.80	-1193.70	-1190.86	-1189.23	-1188.18	-1187.43	-1186.89	-1186.46	-1186.13	-1185.85
Secant iterations	2	3	3	4	4	4	4	4	4	4	4
FBSM iterations	68	105	117	164	168	172	172	176	176	176	176
α	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
Total costs	-1185.63	-1185.44	-1185.28	-1185.14	-1185.02	-1184.91	-1184.81	-1184.73	-1184.65	-1184.59	
Secant iterations	4	5	5	6	5	4	3	6	4	2	
FBSM iterations	176	220	220	270	225	180	135	270	180	90	

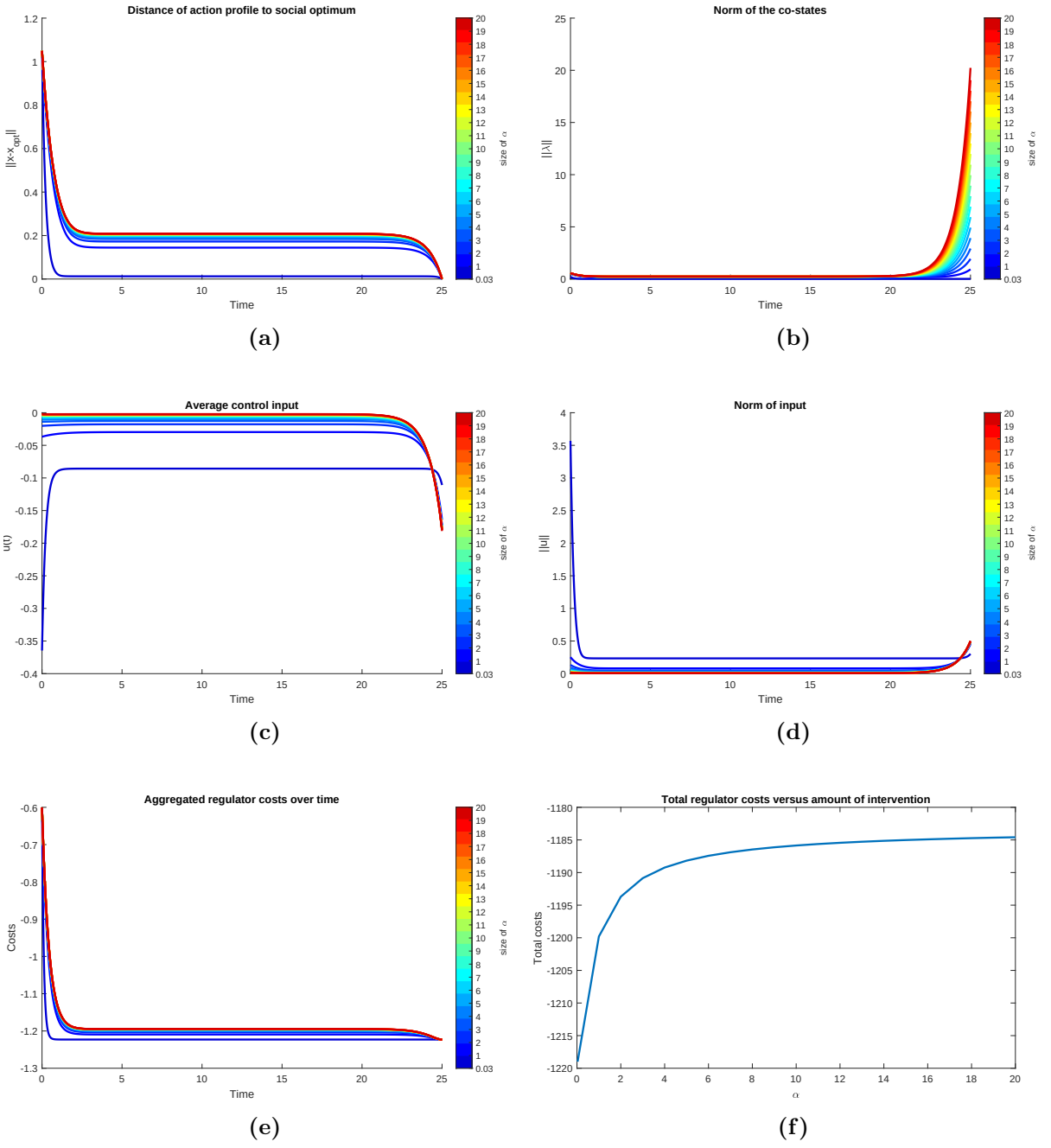


Figure 4.9: Plotted results of the numerical simulations of the modified fixed endpoint optimal control problem for $\alpha \in [0.03, 20]$.

Then, with figure 4.10 we compare the results of the modified controller to the solutions of the static and dynamic interventions. Considering figure 4.10a, we notice that for the optimal control solution with penalty parameter $\alpha = 0.03$, the action profile rapidly converges to a steady-state with a distance of approximately 0.01 from the social optimal state, after which it steers to x_{opt}. Comparing this to the action profiles of the static and dynamic intervention policies, we see that a steady state is reached quicker with the penalized optimal control with $\alpha = 0.03$. Even though the steady-state of the penalized optimal controller is not yet the social optimal state, as is the case for the static and dynamic controllers, figures 4.10c and 4.10d show that the behaviour of the optimal controller does cause slightly lower costs for the regulator.

Explicitly, the costs incurred when using the penalized optimal control with $\alpha = 0.03$ are now -1218.99, whereas the costs with using static feedback intervention are -1210.86. Hence, we can conclude from this that the penalized optimal controller with a penalty parameter value has slightly better performance than the existing intervention policies considered in this section.

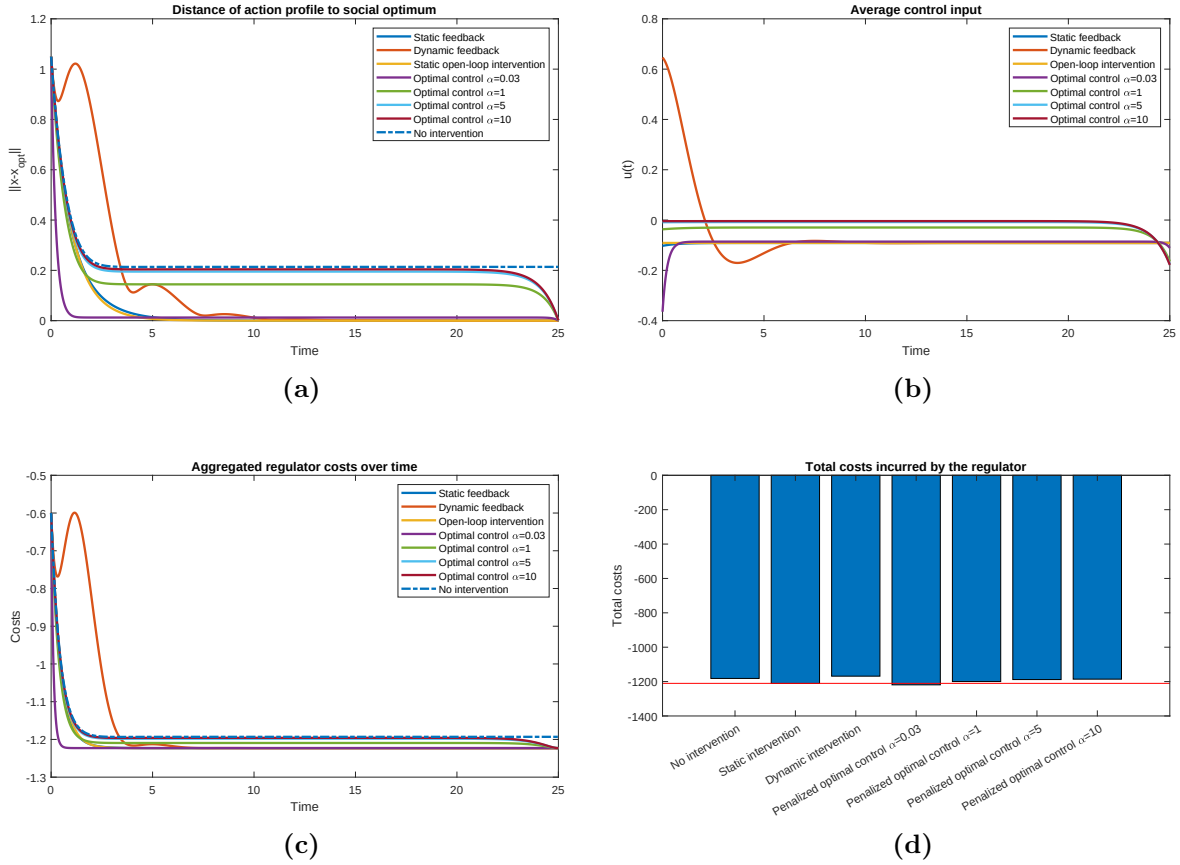


Figure 4.10: Results of the numerical simulations of the fixed endpoint optimal control problems compared with the static and dynamic intervention policies.

Chapter 5

Discussion

5.1 Reflection on the results

Now that we have shown the results of the numerical simulations, we will use this chapter to reflect on the results by taking into account the research questions formulated in section 1.3.1. After analysing the linear-quadratic network game model and defining the requirement for a social optimal solution with theorem 1, chapter 3 showed the derivation of two different types of intervention policies: one that includes a constraint on the maximum amount of control effort to be applied by the regulator, and one that lifts this constraint by including a penalty term in the cost function. Although the first-order necessary conditions remain the same in both optimal control problems, a crucial difference between the two obtained optimal intervention policies is that the constrained optimal controller has a discontinuous nature, whereas the penalized optimal controller is a continuous function. This difference was further illustrated in the first experiment, where it was seen that the discontinuous control function caused the control effort to always take the (positive) value of the maximum allowed intervention u_{max}, causing the action profiles of x^* to converge to a constant value. Moreover, the penalized controller showed that for different sizes of α the action profiles of x^* also reached a steady state at a particular distance from the social optimal state but moved away from that to fulfil the final time condition of the co-state variable $\lambda(T)$. This behaviour became more evident in the second experiment where convergence to the social welfare state was enforced by adding it as a final time constraint in the optimal control problem. By adding the final time constraint, the regular FBSM algorithm needed to be adjusted to find the corresponding value for the transversality condition $\lambda(T)$ that represents the constraint $x(T) = x_{opt}$. In the adjusted algorithm, the guesses $\theta_1 = 10$ and $\theta_2 = -10$ proved to be sufficient for estimating $\lambda(T)$ but convergence could not be achieved for lower values of α in solving the penalized optimal control problem. For penalty parameter values running from 1 to 20, however, we noticed that the dynamics considered in solving the optimal control problem ($\dot{x}(t)$ and $\dot{\lambda}(t)$) tend to steer the solution of the optimal control problem to a certain equilibrium value, only for it to be desolated to reach a final solution that is dependent on the estimated value of the transversality condition $\lambda(T)$.

When comparing the solution of the penalized optimal control to three existing intervention policies found in Shakarami, Cherukuri, and Monshizadeh (2021), we found that the costs incurred by the regulator were higher when using the penalized intervention policy for different α's. Therefore, in the final experiment, the optimal control problem was revised to exclude the double penalization on the control effort $u(t)$ and thus minimize the function $W(x, z(x)$. By solving the modified optimal control problem, we have seen that increasing the penalty parameter α for this optimal control problem entailed an increase in the regulator's costs as well. However, as the value of α approached 20, the increase in costs ebbed. By interpolation, we can

then state that further increases of the penalty parameter α will have less and less effect on the costs. Furthermore, from solving the modified problem we saw that it is now possible to obtain slightly lower costs with the use of a penalized optimal intervention policy with a low amount of penalization on the amount of control effort applied by the regulator.

5.2 Limitations and future research

Through the modification of the optimal control problem in the last experiment, we have shown that in some situations the intervention policy derived in this research can be called slightly more optimal compared to existing policies. However, since this research focused mainly on the derivation of the first-order necessary conditions for this controller and their implementation for numerical calculations, true optimality could not yet be proven. Therefore, to strengthen the argument for the controller's optimality, more significant proof needs to be obtained from experiments, also taking into account comparison with other existing intervention policies, and simulations for different network topologies to test robustness.

The second limitation of this research is that we have not been able to fully compare the working of the optimal controller with admissible control constraints with that of the penalized controller in terms of convergence to the social optimum. The reason for this is that for the constrained optimal control, convergence to the social optimal solution could not be guaranteed using the adapted FBSM algorithm due to the discontinuous nature of the controller expression $u^*(t)$. Therefore, we suggest approximating the expression for the optimal controller using the so-called *control barrier function* technique, a method where the control bounds are included in the objective function to alleviate the admissible control $\mathcal{U}$ set as a constraint (Xiao, Cassandras, and Belta 2019). By doing so, the expression for the controller will take a more continuous form that will have more capability of causing the solution of the optimal control problem to converge to social welfare using the adapted FBSM algorithm.

Lastly, the accuracy of the numerical methods used for obtaining the solution of the optimal control problem could be improved by applying techniques to improve convergence such as those applied in Sharp, Burrage, and Simpson (2021), or by using a direct numerical method rather than the indirect method used in this research. With that, the increase in the accuracy of the results will contribute to the argument for using optimal control as a method to derive policies for social welfare maximization in network games.

Chapter 6

Conclusion

In this research, we have shown to be able to derive and solve the optimal control problems for a network game where players have linear quadratic utility functions. Additionally, we have defined the condition for finding a solution in which social welfare is obtained and for which the regulator has to design an intervention policy. Here the distinction was made between two types of optimal intervention policies: one where the amount of intervention to be performed by the regulator was bounded, and one where intervention by the regulator was punished by adding a penalty term to the cost function. To ensure that social welfare could be reached using the optimal control technique, the social welfare state x_{opt} was added to the formulation of the optimal control problem as a final time constraint. To solve this fixed endpoint optimal control problem, an adaptation of the forward-backward sweep method was required. In the numerical experiments, we observed that due to the constrained controller's discontinuous nature, social welfare could only be achieved for the penalized controller. With that, the second experiment also showed that for increasing values of the penalty parameter α, the amount of intervention, and with that also the costs incurred by the regulator, decreased. However, the third experiment showed that compared to static and dynamic feedback intervention policies derived in preceding literature, the optimal controller derived in this research showed worse performance in terms of total costs incurred by the regulator. Therefore, another experiment was performed with a modified cost function where the double penalization of the control effort $u(t)$ was removed. The numerical simulation for this experiment showed that the penalized controller could offer slightly lower regulator costs compared to the static and dynamic intervention policies. For future work, we suggest seeking to strengthen the argument for calling this intervention policy truly optimal by performing more experiments for obtaining more significant and accurate proof.

Bibliography

Başar, Tamer and Georges Zaccour (2018). *Handbook of dynamic game theory*. Springer.

Başar, Tamer and Quanyan Zhu (2011). "Prices of anarchy, information, and cooperation in differential games". In: *Dynamic Games and Applications* 1.1, pp. 50–73.

Bui, Khac-Hoai Nam and Jason J Jung (2018). "Cooperative game-theoretic approach to traffic flow optimization for multiple intersections". In: *Computers & Electrical Engineering* 71, pp. 1012–1024.

Corbo, Jacomo, Antoni Calvó-Armengol, and David C Parkes (2007). "The importance of network topology in local contribution games". In: *International workshop on web and internet economics*. Springer, pp. 388–395.

Currarini, Sergio and Francesco Feri (2015). "Information sharing networks in linear quadratic games". In: *International Journal of Game Theory* 44.3, pp. 701–732.

De Persis, Claudio and Sergio Grammatico (2019). "Distributed averaging integral Nash equilibrium seeking on networks". In: *Automatica* 110, p. 108548.

Deori, Luca, Kostas Margellos, and Maria Prandini (2018). "Price of anarchy in electric vehicle charging control games: When Nash equilibria achieve social welfare". In: *Automatica* 96, pp. 150–158.

Dolgopolik, MV (2020). "Exact penalty functions for optimal control problems II: Exact penalization of terminal and pointwise state constraints". In: *Optimal Control Applications and Methods* 41.3, pp. 898–947.

Dolgopolik, MV and AV Fominyh (2019). "Exact penalty functions for optimal control problems I: Main theorem and free-endpoint problems". In: *Optimal Control Applications and Methods* 40.6, pp. 1018–1044.

Galeotti, Andrea, Benjamin Golub, and Sanjeev Goyal (2020). "Targeting interventions in networks". In: *Econometrica* 88.6, pp. 2445–2471.

Gao, Xiangyu, Xian Zhang, and Yantao Wang (2014). "A simple exact penalty function method for optimal control problem with continuous inequality constraints". In: *Abstract and Applied Analysis*. Vol. 2014. Hindawi.

Geering, Hans P (2007). *Optimal control with engineering applications*. Springer.

Khan, M Ali and Yeneng Sun (2002). "Non-cooperative games with many players". In: *Handbook of game theory with economic applications* 3, pp. 1761–1808.

Lenhart, Suzanne and John T Workman (2007). *Optimal control applied to biological models*. Chapman and Hall/CRC.

Ma, Zhongjing, Duncan S Callaway, and Ian A Hiskens (2011). "Decentralized charging control of large populations of plug-in electric vehicles". In: *IEEE Transactions on control systems technology* 21.1, pp. 67–78.

Papakonstantinou, Joanna M and Richard A Tapia (2013). "Origin and evolution of the secant method in one dimension". In: *The American Mathematical Monthly* 120.6, pp. 500–517.

Parise, Francesca and Asuman Ozdaglar (2021). "Analysis and interventions in large network games". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4, pp. 455–486.

Rodrigues, Helena Sofia, M Teresa T Monteiro, and Delfim FM Torres (2014). "Optimal control and numerical software: an overview". In: *arXiv preprint arXiv:1401.7279*.

Shakarami, Mehran, Ashish Cherukuri, and Nima Monshizadeh (2021). "Adaptive interventions for social welfare maximization in network games". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 942–947.

Sharp, Jesse A, Kevin Burrage, and Matthew J Simpson (2021). "Implementation and acceleration of optimal control for systems biology". In: *bioRxiv*.

Vasnani, Neelesh N et al. (2019). "Game theory in supply chain management: Current trends and applications". In: *International Journal of Applied Decision Sciences* 12.1, pp. 56–97.

Xiao, Wei, Christos G Cassandras, and Calin Belta (2019). "Decentralized merging control in traffic networks with noisy vehicle dynamics: A joint optimal control and barrier function approach". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, pp. 3162–3167.

Zhu, Quanyan and Tamer Başar (2010). "Price of anarchy and price of information in N-person linear-quadratic differential games". In: *Proceedings of the 2010 American Control Conference*. IEEE, pp. 762–767.

Zwart, Hans et al. (Nov. 2012). *Lecture notes in Calculus of Variations and Optimal Control*.

Appendix A

Derivation of first-order necessary conditions for the optimal control problem 4.2

Consider the following optimal control problem:

$$\min_{u} \quad \hat{J}(x_0, u) = \int_0^T \underbrace{\left(\frac{1}{2}x(t)^T x(t) - x(t)^T(aPx(t)+b) + \alpha||u(t)||_2^2\right)}_{L(x,u)} dt \tag{A.1a}$$

$$\text{subject to} \quad \dot{x}(t) = (-I + aP)x(t) + b + u(t), \tag{A.1b}$$

$$x(0) = x_0, \tag{A.1c}$$

$$x(T) = x_{opt}, \tag{A.1d}$$

$$t \in [0, T]. \tag{A.1e}$$

With this, we derive the following Hamiltonian function to be minimized:

$$H(x(t), \lambda(t), u(t)) = \lambda^T(t) f\left(x(t), u(t)\right) + L\left(x(t), u(t)\right), \tag{A.2}$$

$$= \lambda^T(t)(-I + aP)x(t) + b + u(t) + \frac{1}{2}x(t)^T x(t) - x(t)^T(aPx(t)+b) + \alpha||u(t)||_2^2, \tag{A.3}$$

where $\lambda : [0, T] \to \mathbb{R}^n$ plays the role of the Lagrangian multiplier. Then, using theorem 2, we can derive the first order necessary conditions for optimal control:

$$\dot{x}^*(t) = \frac{\partial H}{\partial \lambda}\left(x^*(t), \lambda^*(t), u^*(t)\right)^T = (-I + aP)x^*(t) + b + u^*(t), \quad x^*(0) = x_0, \tag{A.4a}$$

$$\dot{\lambda}^*(t) = -\frac{\partial H}{\partial x}\left(x^*(t), \lambda^*(t), u^*(t)\right)^T = (I - aP^T)\lambda^*(t) - x^*(t) + a(P + P^T)x^*(t) + b, \tag{A.4b}$$

$$\lambda^*(T) = \theta, \tag{A.4c}$$

$$\frac{\partial H}{\partial u}\left(x^*(t), \lambda^*(t), u^*(t)\right)^T = -\frac{\lambda^*}{2\alpha}. \tag{A.4d}$$

Notice that for equation (A.4c), we already made a guess on the value of λ^* at time T (namely θ) and therefore we do not consider a final cost function $S(x(T))$ to be added to the cost criterion A.1a. Furthermore, if we rewrite the optimality condition A.4d, we obtain the expression for the optimal controller $u^*(t)$:

$$u^*(t) = -\frac{\lambda}{2\alpha}. \tag{A.5}$$

Appendix B

Matlab code

B.1 Main program

```
1
2  %% Set-up
3  clear; clc; close all;
4  set(0,'DefaultFigureWindowStyle','docked')
5  %set(0,'DefaultFigureWindowStyle','normal')
6  rng('default');
7  t0 = 0;                                                    % initial time
8  parameters.tf = 25;                                        % final time
9  parameters.dt = 0.025;                                     % size of time-step
10 parameters.N = floor(parameters.tf/parameters.dt-1);        % number of ...
       nodes in time discretization
11 N = parameters.N;
12 parameters.tspan = linspace(t0,parameters.tf,parameters.N+1);% time ...
       discretization
13 tspan = parameters.tspan;
14 parameters.omega = 0.9;                                     % weighting ...
       factor of previous iteration's control maintained
15 parameters.delta = 1e-3;                                   % tolerance of ...
       the FBSM
16 parameters.epsilon = 1.5e-3;                               % tolerance of ...
       the secant method
17 experiment = 2;                                            % experiment ...
       indicator
18
19 % Model parameters
20 parameters.dim = 6;                                        % dimension of ...
       state variable
21 dim = parameters.dim;
22 P_dir =     [0 0.5 0 0 0 0;
23              1 0 0.35 0 0 0;
24              0 0 0 0 0.8 0;
25              0 0.7 0 0 0 0;
26              0 0 0.6 0.15 0 0.3;
27              0 0 0.4 0 0 0];                               % adjacency ...
                    matrix directed network
28 parameters.P = P_dir(1:dim,1:dim);
29 P = parameters.P;
30 parameters.a = -0.2;                                      % effect of ...
       neighbours in the network
31 a = parameters.a;
32 b = [0.35 ; 0.80 ; 1.05 ; 0.65 ; 0.95 ; 0.25];            % standalone ...
       marginal returns
```

```
parameters.b = b(1:dim);
b = parameters.b;
parameters.I = eye(dim);                                    % create ...
    variable for identity matrix
xopt = (parameters.I-a*(P+P'))^-1*b;                        % calculate ...
    value of social optimal state
umax = 1;                                                   % bounds on the ...
    control
alphas = 1:20;                                              % vector of ...
    different penalty parameters


% Initial conditions
x = cell(length(alphas),1);                                 % create cell to ...
    store the time series x for each iteration
lambda = cell(length(alphas),1);
U = cell(length(alphas),1);
J = zeros(3+length(alphas),1);
Jt = zeros(3+length(alphas),N+1);
numberIterationsFBSM = zeros(length(alphas),1);
numberIterationsSecant = zeros(length(alphas),1);
y0 = [rand(dim,1); rand(dim,1)];                            % initial state
x0 = y0(1:dim);
parameters.theta1 = 10*ones(dim,1);                         % first guess on ...
    lambda(T) used for adapted FBSM
parameters.theta2 = -10*ones(dim,1);                        % second guess ...
    on lambda(T) used for adapted FBSM
parameters.lambdaT = 0*ones(dim,1);                         % lambda(T) used ...
    for regular FBSM


%% Numerical calculations
% Results of Shakarami, et al. (2021)
% static open-loop
[~, ySOL] = ode45(@(t,y) dynFun(y,"uSOL",parameters,xopt),tspan,y0);
uSOL = (eye(dim)-a*P)*xopt-b;
JtOL = sum((1/2)*ySOL(:,1:dim)'.^2 - ySOL(:,1:dim)'.*(a*P*ySOL(:,1:dim)' + b));
JOL =trapz(JtOL,2);

% no intervention
[~, yNo] = ode45(@(t,y) dynFun(y,"u0",parameters,xopt),tspan,y0);
Jt(1,:) = sum((1/2)*yNo(:,1:dim)'.^2-yNo(:,1:dim)'.*(a*P*yNo(:,1:dim)'+b));
J(1) =trapz(Jt(1,:),2);

% static feedback
[~, yS] = ode45(@(t,y) dynFun(y,"uS",parameters,xopt),tspan,y0);
uS = (a*P'*yS(:,1:dim)');
Jt(2,:) = sum((1/2)*yS(:,1:dim)'.^2-yS(:,1:dim)'.*(a*P*yS(:,1:dim)'+b));
J(2) =trapz(Jt(2,:),2);

% dynamic feedback
[~, yD] = ode45(@(t,y) dynFun(y,"uD",parameters,xopt),tspan,y0);
Jt(3,:) = sum((1/2)*yD(:,1:dim)'.^2-yD(:,1:dim)'.*(a*P*yD(:,1:dim)'+b));
J(3) =trapz(Jt(3,:),2);


% Results of optimal controllers using regular Forward-Backward Sweep
% (experiment 1)
% for i = 1:length(alphas)
%     parameters.alpha = alphas(i);
%     parameters.V1 = zeros(dim,1);
%     parameters.uBound = umax*ones(dim,1);
```

```
%     [x{i},lambda{i},U{i},Jt(i+3,:),numberIterationsFBSM(i)] = ...
    FBSM(x0,parameters);
%     J(i+3) = trapz(Jt(i+3,:),2);
% end

% Results of optimal controllers using Adapted Forward-Backward Sweep
for i = 1:length(alphas)
    parameters.alpha = alphas(i);
    [x{i},lambda{i},U{i},Jt(i+3,:),numberIterationsFBSM(i), ...
        numberIterationsSecant(i)] = AFBSM(x0,parameters,xopt);
    J(i+3) = trapz(Jt(i+3,:),2);
end


% Calculate norms and averages
p_norm = 2;
normx = zeros(length(alphas),N+1);
normlambda = zeros(length(alphas),N+1);
normu = zeros(length(alphas),N+1);
normxdist = zeros(length(alphas),N+1);
normxNo = zeros(parameters.N+1,1);
normxdistNo = zeros(parameters.N+1,1);
normxS = zeros(parameters.N+1,1);
normxSOL = zeros(parameters.N+1,1);
normxdistS = zeros(parameters.N+1,1);
normxdistSOL = zeros(parameters.N+1,1);
normxD = zeros(N+1,1);
normxdistD = zeros(N+1,1);
avgu = zeros(length(alphas),N+1);
avguS = zeros(N+1,1);
avguSOL = zeros(N+1,1);
avguD = zeros(N+1,1);

for i = 1:N+1
    for k = 1:length(alphas)
        normx(k,i) = norm(x{k}(:,i));
        normlambda(k,i) = norm(lambda{k}(:,i));
        normu(k,i) = norm(U{k}(:,i));
        normxdist(k,i) = norm(x{k}(:,i) - xopt,p_norm);
        avgu(k,i) = mean(U{k}(:,i));
    end
    normxNo(i) = norm((yNo(i,1:dim)),p_norm);
    normxdistNo(i) = norm((yNo(i,1:dim) - xopt'),p_norm);
    normxS(i) = norm((yS(i,1:dim)),p_norm);
    normxSOL(i) = norm((ySOL(i,1:dim)),p_norm);
    normxdistS(i) = norm((yS(i,1:dim) - xopt'),p_norm);
    normxdistSOL(i) = norm((ySOL(i,1:dim) - xopt'),p_norm);
    normxD(i) = norm((yD(i,1:dim)),p_norm);
    normxdistD(i) = norm((yD(i,1:dim) - xopt'),p_norm);
    avguSOL(i) = mean(uSOL);
    avguS(i) = mean(uS(:,i));
    avguD(i) =  mean(yD(i,1+dim:2*dim));
end



% Display simulation results in command window
disp('------------------------------------------------------------------------');
disp('Numerical results:')
disp([' ' 'total costs without intervention: ' num2str(sum(J(1)))]);
disp([' ' 'total costs using dynamic intervention: ' num2str(sum(J(2)))]);
disp([' ' 'total costs using dynamic intervention: ' num2str(sum(J(3)))]);
disp('------------------------------------------------------------------------');
```

```
148 for i = 1:length(alphas)
149 disp([' ' 'total number of FBSM iterations optimal control with penalty ...
        alpha=' num2str(alphas(i)) ': ' num2str(numberIterationsFBSM(i))]);
150 disp([' ' 'number of secant iterations optimal control with penalty alpha=' ...
        num2str(alphas(i)) ': ' num2str(numberIterationsSecant(i))]);
151 disp([' ' 'total costs optimal control: ' num2str(sum(J(i+3)))]);
152 disp('------------------------------------------------------------------------');
153 end
154
155
156
157 %% Experiment 1
158 if experiment == 1
159     close all
160     % Norm of the states
161     figure
162     hold on
163     plot(tspan,normxSOL,'LineWidth',2,'DisplayName',"Static open-loop ...
            intervention")
164     xlabel('Time')
165     ylabel('||x||')
166     for i = 1:length(alphas)
167         if alphas(i)==0
168             lgdtxt = ['Constrained optimal control u_{max}=' num2str(umax)];
169         else
170             lgdtxt = ['Optimal control \alpha=' num2str(alphas(i))];
171         end
172         plot(tspan,normx(i,:),'LineWidth',2,'DisplayName',lgdtxt)
173     end
174     plot(tspan,normxNo,'-.','LineWidth',2,'DisplayName',"No intervention")
175     hold off
176     legend show
177     title("Norm of the players' action profiles")
178
179     % Norm of the co-states
180     figure
181     hold on
182     plot(tspan,normxSOL,'LineWidth',2,'DisplayName',"Static open-loop ...
            intervention")
183     xlabel('Time')
184     ylabel('||\lambda||')
185     for i = 1:length(alphas)
186         if alphas(i)==0
187             lgdtxt = ['Constrained optimal control u_{max}=' num2str(umax)];
188         else
189             lgdtxt = ['Optimal control \alpha=' num2str(alphas(i))];
190         end
191         plot(tspan,normlambda(i,:),'LineWidth',2,'DisplayName',lgdtxt)
192     end
193     plot(tspan,normxNo,'-.','LineWidth',2,'DisplayName',"No intervention")
194     hold off
195     legend show
196     title("Norm of the co-states")
197
198     % Distance of action profiles to social optimum
199     figure
200     hold on
201     plot(tspan,normxdistSOL,'LineWidth',2,'DisplayName',"Static open-loop ...
            intervention")
202     xlabel('Time')
203     ylabel('||x-x_{opt}||')
204     for i = 1:length(alphas)
205         if alphas(i)==0
```

```
206             lgdtxt = ['Constrained optimal control u_{max}=' num2str(umax)];
207         else
208             lgdtxt = ['Optimal control \alpha=' num2str(alphas(i))];
209         end
210         plot(tspan,normx(i,:),'LineWidth',2,'DisplayName',lgdtxt)
211     end
212     plot(tspan,normxdistNo,'-.','LineWidth',2,'DisplayName',"No intervention")
213     hold off
214     legend show
215     title('Distance of action profile to social optimum')
216
217     % Plots of the outputs of the optimal controllers
218     for k = 1:length(alphas)
219         figure
220         subplot(3,1,1)
221         plot(tspan,x{k}(1:dim,:),'LineWidth',2)
222         xlabel('Time')
223         ylabel('x')
224         title('State variable')
225
226         subplot(3,1,2)
227         plot(tspan,lambda{k}(1:dim,:),'LineWidth',2)
228         xlabel('Time')
229         ylabel('\lambda')
230         title('Adjoint variable')
231
232         subplot(3,1,3)
233         plot(tspan,U{k}(1:dim,:),'LineWidth',2)
234         xlabel('Time')
235         ylabel('u')
236         title('Control input')
237
238         if alphas(k) ==0
239             txt = ['Constrained optimal control u_{max}=' num2str(umax)];
240         else
241             txt = ['Penalized optimal controller output \alpha=' ...
                    num2str(alphas(k))];
242         end
243         sgtitle(txt)
244     end
245
246     % Plot of the average control input
247     figure
248     plot(tspan,avguSOL,'LineWidth',2,'DisplayName',"Static open-loop ...
            intervention")
249     hold on
250     xlabel('Time')
251     ylabel('u(t)')
252     for i = 1:length(alphas)
253         if alphas(i)==0
254             lgdtxt = ['Constrained optimal control u_{max}=' num2str(umax)];
255         else
256             lgdtxt = ['Optimal control \alpha=' num2str(alphas(i))];
257         end
258         plot(tspan,avgu(i,:),'LineWidth',2,'DisplayName',lgdtxt)
259     end
260     hold off
261     title('Average control input')
262     legend show
263
264
265 end
266
```

```
%% Experiment 2: Reaching the social welfare solution
if experiment == 2
    close all
    % Distance of action profiles to social optimum
    figure
    xlabel('Time')
    ylabel('||x-x_{opt}||')
    hold on
    cc = colormap(jet(length(alphas)));
    for i = 1:length(alphas)
        plot(tspan,normxdist(i,:),'color',cc(i,:),'LineWidth',2)
    end
    hold off
    caxis([min(alphas) max(alphas)])
    c = colorbar('Ticks',alphas);
    c.Label.String = 'size of \alpha';
    title('Distance of action profile to social optimum')

    % Norm of the co-states
    figure
    hold on
    cc = colormap(jet(length(alphas)));
    caxis([min(alphas) max(alphas)])
    c = colorbar('Ticks',alphas);
    c.Label.String = 'size of \alpha';
    xlabel('Time')
    ylabel('||\lambda||')
    for i = 1:length(alphas)
        plot(tspan,normlambda(i,:),'color',cc(i,:),'LineWidth',2)
    end
    hold off
    title("Norm of the co-states")

    % Plot of average input
    figure
    xlabel('Time')
    ylabel('u(t)')
    cc = colormap(jet(length(alphas)));
    caxis([min(alphas) max(alphas)])
    c = colorbar('Ticks',alphas);
    c.Label.String = 'size of \alpha';
    hold on
    for i = 1:length(alphas)
        plot(tspan,avgu(i,:),'Color',cc(i,:),'LineWidth',2)
    end
    hold off
    title('Average control input')

    % Plot of norm of the input
    figure
    xlabel('Time')
    ylabel('||u||')
    cc = colormap(jet(length(alphas)));
    caxis([min(alphas) max(alphas)])
    c = colorbar('Ticks',alphas);
    c.Label.String = 'size of \alpha';
    hold on
    for i = 1:length(alphas)
        plot(tspan,normu(i,:),'Color',cc(i,:),'LineWidth',2)
    end
    hold off
    title('Norm of input')

```

```
330
331     % Plots of state variables of optimal controllers
332     for k = 1:length(alphas)
333         figure
334         subplot(3,1,1)
335         plot(tspan,x{k}(1:dim,:),'LineWidth',2)
336         xlabel('Time')
337         ylabel('x')
338         title('State variable')
339
340         subplot(3,1,2)
341         plot(tspan,lambda{k}(1:dim,:),'LineWidth',2)
342         xlabel('Time')
343         ylabel('\lambda')
344         title('Adjoint variable')
345
346         subplot(3,1,3)
347         plot(tspan,U{k}(1:dim,:),'LineWidth',2)
348         xlabel('Time')
349         ylabel('u')
350         title('Control input')
351         if alphas(k) ==0
352             txt = 'Constrained optimal controller output';
353         else
354             txt = ['Penalized optimal controller output \alpha=' ...
                    num2str(alphas(k))];
355         end
356         sgtitle(txt)
357     end
358
359
360     % Plot of the costs over time
361     figure
362     hold on
363     cc= colormap(jet(length(alphas)));
364     caxis([min(alphas) max(alphas)])
365     c = colorbar('Ticks',alphas);
366     c.Label.String = 'size of \alpha';
367     for i = 1:length(alphas)
368         plot(tspan,Jt(i+3,:),'Color',cc(i,:),'LineWidth',2,'DisplayName')
369     end
370     hold off
371     title('Aggregated regulator costs over time')
372     xlabel('Time')
373     ylabel('Costs')
374
375
376     % Plot of costs versus increasing values of alpha
377     figure
378     plot(alphas,J(4:end),'LineWidth',2)
379     title('Total regulator costs versus amount of intervention')
380     xlabel('\alpha')
381     ylabel('Total costs')
382
383 end
384 %% Experiment 3: Comparison with static and dynamic intervention policies
385 if experiment == 3
386     close all
387
388     % Norm of the states
389     figure
390     plot(tspan,normxdistS,'LineWidth',2,'DisplayName',"Static feedback")
391     xlabel('Time')
```

```
392     ylabel('||x-x_{opt}||')
393     hold on
394     plot(tspan,normxdistD,'LineWidth',2,'DisplayName',"Dynamic feedback")
395     plot(tspan,normxdistSOL,'LineWidth',2,'DisplayName',"Static open-loop ...
            intervention")
396     for i = 1:length(alphas)
397         if alphas(i)==0
398             lgdtxt = 'Constrained optimal control';
399         else
400             lgdtxt = ['Optimal control \alpha=' num2str(alphas(i))];
401         end
402         plot(tspan,normxdist(i,:),'LineWidth',2,'DisplayName',lgdtxt)
403     end
404     plot(tspan,normxdistNo,'-.','LineWidth',2,'DisplayName',"No intervention")
405     hold off
406     legend show
407     title("Distance of action profile to social optimum")
408
409     % Plot of input norms
410     figure
411     plot(tspan,avguS,'LineWidth',2,'DisplayName',"Static feedback")
412     xlabel('Time')
413     ylabel('u(t)')
414     hold on
415     plot(tspan,avguD,'LineWidth',2,'DisplayName',"Dynamic feedback")
416     plot(tspan,avguSOL,'LineWidth',2,'DisplayName',"Open-loop intervention")
417     for i = 1:length(alphas)
418         if alphas(i)==0
419             lgdtxt = 'Constrained optimal control';
420         else
421             lgdtxt = ['Optimal control \alpha=' num2str(alphas(i))];
422         end
423         plot(tspan,avgu(i,:),'LineWidth',2,'DisplayName',lgdtxt)
424     end
425     hold off
426     title('Average control input')
427     legend show
428
429
430     % Plot of the costs over time
431     figure
432     plot(tspan,Jt(2,:),'LineWidth',2,'DisplayName',"Static feedback")
433     hold on
434     plot(tspan,Jt(3,:),'LineWidth',2,'DisplayName',"Dynamic feedback")
435     plot(tspan,JtOL,'LineWidth',2,'DisplayName',"Open-loop intervention")
436     for i = 1:length(alphas)
437         if alphas(i)==0
438             lgdtxt = 'Constrained optimal control';
439         else
440             lgdtxt = ['Optimal control \alpha=' num2str(alphas(i))];
441         end
442         plot(tspan,Jt(i+3,:),'LineWidth',2,'DisplayName',lgdtxt)
443     end
444     plot(tspan,Jt(1,:),'-.','LineWidth',2,'DisplayName',"No intervention")
445     hold off
446     legend show
447     title('Aggregated regulator costs over time')
448     xlabel('Time')
449     ylabel('Costs')
450
451     % Bar plot of total costs
452     figure
453     xlabels = {'No intervention','Static intervention','Dynamic intervention'};
```

```
454     for i = 1:length(alphas)
455         if alphas(i) ==0
456             xlabels{i+3} = 'Constrained optimal control';
457         else
458             xlabels{i+3} = ['Penalized optimal control \alpha=' ...
                    num2str(alphas(i))];
459         end
460     end
461     barx = 1:length(alphas)+3;
462     bar(barx,J)
463     line(xlim,[J(2) J(2)],'Color','r');
464     xticks(1:length(alphas)+3);
465     xticklabels(xlabels)
466     ylabel('Total costs')
467     title('Total costs incurred by the regulator')
468 end
```

B.2 Forward-backward sweep algorithm

```
1  function [x,lambda,u,Jt,numberIterations] = FBSM(x0,parameters)
2
3  % Initialize parameters
4  dim = parameters.dim;
5  a = parameters.a;
6  b = parameters.b;
7  P = parameters.P;
8  N = parameters.N;
9  h = parameters.tf/N;
10 tspan = parameters.tspan;
11 omega = parameters.omega;
12 delta = parameters.delta;
13 test = -1;
14 numberIterations = 0;
15 maxIterations = 300;
16
17 % Initialize state, co-state and control
18 x = zeros(dim,N+1);
19 x(:,1) = x0;
20 lambda = zeros(dim,N+1);
21 lambda(:,end) = parameters.lambdaT;
22 u = zeros(dim,N+1);
23
24 while(test<0)
25
26     % Save variables of previous iteration
27     uOld = u;
28     xOld = x;
29     lambdaOld = lambda;
30
31     % Plot intermediate results
32 %     if parameters.V1 < parameters.epsilon
33 %         figure(30)
34 %         subplot(3,1,1)
35 %         plot(tspan,x(1:dim,:))
36 %         xlabel('Time')
37 %         ylabel('x')
38 %         title('State variable')
39 %
40 %         subplot(3,1,2)
41 %         plot(tspan,lambda(1:dim,:))
```

```
42 %         xlabel('Time')
43 %         ylabel('\lambda')
44 %         title('Adjoint variable')
45 %
46 %         subplot(3,1,3)
47 %         plot(tspan,u(1:dim,:))
48 %         xlabel('Time')
49 %         ylabel('u')
50 %         title('Control input')
51 %         if parameters.alpha ==0
52 %             txt = 'Constrained optimal controller feedback';
53 %         else
54 %             txt = ['Penalized optimal controller feedback \alpha=' ...
       num2str(parameters.alpha)];
55 %         end
56 %         sgtitle(txt)
57 %     end
58
59     % Forward sweep using Runge-Kutta 4
60     for i = 1:N
61         k1 = stateEq(x(:,i), u(:,i), parameters);
62         k2 = stateEq(x(:,i)+h*k1/2, 0.5*(u(:,i) + u(:,i+1)), parameters);
63         k3 = stateEq(x(:,i)+h*k2/2, 0.5*(u(:,i) + u(:,i+1)), parameters);
64         k4 = stateEq(x(:,i)+h*k3, u(:,i+1), parameters);
65         x(:,i+1) = x(:,i) + (h/6)*(k1 + 2*k2 + 2*k3 + k4);
66     end
67
68     % Backward sweep using Runge-Kutta 4
69     for i = 1:N
70         j = N+2-i;
71         k1 = costateEq(x(:,j), lambda(:,j), u(:,j),parameters);
72         k2 = costateEq(0.5*(x(:,j)+x(:,j-1)), lambda(:,j)-h*k1/2, ...
               0.5*(u(:,j)+u(j-1)), parameters);
73         k3 = costateEq(0.5*(x(:,j)+x(:,j-1)), lambda(:,j)-h*k2/2, ...
               0.5*(u(:,j)+u(j-1)), parameters);
74         k4 = costateEq(x(:,j-1), lambda(:,j)+h*k3, u(:,j-1),parameters);
75         lambda(:,j-1) = lambda(:,j) - (h/6)*(k1 + 2*k2 + 2*k3 + k4);
76     end
77
78
79     % Update the control
80     uNew = controlEq(x,lambda,parameters);
81     u = omega*uOld + (1-omega)*uNew;
82
83     % Check convergence
84     temp1 = delta*sum(abs(u),2)-sum(abs(uOld-u),2);
85     temp2 = delta*sum(abs(x),2)-sum(abs(xOld-x),2);
86     temp3 = delta*sum(abs(lambda),2)-sum(abs(lambdaOld-lambda),2);
87     test = min(min(temp1, min(temp2,temp3)));
88
89     % Give error message if number of iterations becomes too high
90     if numberIterations > maxIterations && test<0
91         error('Maximum number of % i FBSM iterations reached, current test ...
               value is %4.3f. alpha=%4.2f \n',maxIterations,test,parameters.alpha)
92     end
93     numberIterations = numberIterations + 1;
94 end
95
96 % Calculate costs over time
97 Jt = sum((1/2)*x.^2-x.*(a*P*x + b));
98
99
100
```

```
% Validate the obtained results
% controlValidation(x,lambda,u,parameters);


end
```

B.3 Program used for root finding in the adapted FBSM algorithm

```
function [x,lambda,u,Jt,numberIterationsFBSM, numberIterationsSecant] = ...
    AFBSM(x0,parameters,xopt)

% Initialize parameters
test = -1;
epsilon = parameters.epsilon;
tspan = parameters.tspan;
dim = parameters.dim;
numberIterationsFBSM = 0;
numberIterationsSecant = 0;
maxIterations = 200;
parameters.V1 = ones(dim,1);

% Calculate V(theta1)
theta1 = parameters.theta1;
parameters.lambdaT = theta1;
[x,~,~,~,~] = FBSM(x0,parameters);
V1 = x(:,end) - xopt;

% Calculate V(theta2)
theta2 = parameters.theta2;
parameters.lambdaT = theta2;
[x,~,~,~,~] = FBSM(x0,parameters);
V2 = x(:,end) - xopt;

while(test<0)
    % If the guess for theta2 is better, switch the variables for theta to
    % use the better one
    for i = 1:dim
        if abs(V1(i)) > abs(V2(i))
            temp = theta1;
            theta1 = theta2;
            theta2 = temp;
            temp = V1;
            V1 = V2;
            V2 = temp;
        end
    end
    % Calculate finite difference approximation times the function value,
    % and use that to make better estimate of theta
    fdapprox = V1.*((theta2-theta1)./(V2-V1));
    theta2 = theta1;
    V2 = V1;
    theta1 = theta1 - fdapprox;
    parameters.lambdaT = theta1;
    [x,lambda,u,Jt,numberIterations] = FBSM(x0,parameters);
    V1 = x(:,end) - xopt;
    parameters.V1 = V1;
    numberIterationsFBSM = numberIterationsFBSM + numberIterations;


```

```
51      % If solution converges to social optimal state, stop iterating
52      if max(abs(V1)) < epsilon
53          test = 0;
54      elseif isnan(parameters.lambdaT(1))
55          error('lambdaT has obtained an intractable value. Highest value of V1 ...
                is %4.3d.',max(abs(V1)))
56      elseif numberIterationsSecant > maxIterations
57          error('Maximum number of % i secant iterations reached. Highest value ...
                of V1 is %4.3d, alpha=%4.3f. ...
                \n',maxIterations,max(abs(V1)),parameters.alpha)
58      end
59      numberIterationsSecant = numberIterationsSecant+1;
60      parameters.currentSecantIt = numberIterationsSecant;
61  end
62
63  % Validate the obtained results
64  % controlValidation(x,lambda,u,parameters);
65
66  end
```

B.4 Function used for validating the calculations

```
1   function controlValidation(x,lambda,u,parameters)
2
3   % Initialize
4   alpha = parameters.alpha;
5   dim = parameters.dim;
6   a = parameters.a;
7   b = parameters.b;
8   P = parameters.P;
9   I = parameters.I;
10  dt = parameters.dt;
11  N = parameters.N;
12  delta = 0.01;
13
14  % Calculate derivatives using their prescribed functions and the actual ...
        derivatives
15  testxdot = (-I+a*P)*x + b + u;
16  testlambdadot = (I-a*P')*lambda - x + a*(P+P')*x + b + u;
17  dxdt = diff(x,1,2)/dt;
18  dlambdadt = diff(lambda,1,2)/dt;
19
20  testx = abs(testxdot(:,1:end-1) - dxdt);
21  testlambda = abs(testlambdadot(:,1:end-1) - dlambdadt);
22
23  % Test whether the derivatives are the same and give an error if results do
24  % not match within the given tolerance
25  if testx > delta*ones(dim,length(dxdt))
26      largesterr = max(max(testx - delta*ones(dim,length(dxdt))));
27      error('Obtained results cannot be validated, x is not valid. Norm of test ...
            x1: %5.2f, largest error is %4.3f',norm(testx),largesterr)
28  elseif testlambda > delta*ones(dim,length(dxdt))
29      largesterr = max(max(testlambda - delta*ones(dim,length(dlambdadt))));
30      error('Obtained results cannot be validated, lambda is not valid. Norm of ...
            test lambda1: %5.2f, largest error is %4.3f',norm(testlambda),largesterr)
31  else
32      txt = [' ' 'calculated vectors of x and lambda have been validated'];
33  end
34
35  % Display results of validation and show plots
```

```
disp(['Validation result for alpha = ' num2str(alpha) ':']);
disp(txt);
disp('------------------------------------------------------------------------');

figure
plot(1:N,sum(testxdot(:,1:end-1)/dim),'Displayname','x-dot')
hold on
plot(1:length(dxdt),sum(dxdt)/dim,'-.','LineWidth',1.5,'DisplayName','dxdt')
legend show
ylabel('$\dot{x}$', 'Interpreter','latex')
xlabel('t')
title(['Validation of x^*, \alpha=' num2str(alpha)])

figure
plot(1:N,sum(testlambdadot(:,1:end-1)/dim),'Displayname','\lambda-dot')
hold on
plot(1:length(dlambdadt),sum(dlambdadt)/dim,'-.','LineWidth',1.5,'DisplayName','d\lambda ...
    dt')
legend show
ylabel('$\dot{\lambda}$', 'Interpreter','latex')
xlabel('t')
title(['Validation of \lambda^*, \alpha=' num2str(alpha)])

end
```

B.5 Dynamic functions and controller expression

B.5.1 State dynamics

```
function xdot = stateEq(x,u,parameters)

% initialize parameters
a = parameters.a;
b = parameters.b;
P = parameters.P;
I = parameters.I;

% define state equation
xdot = (-I + a*P)*x + b + u;

end
```

B.5.2 Co-state dynamics

```
function lambdadot = costateEq(x,lambda,u,parameters)

% initialize parameters
a = parameters.a;
b = parameters.b;
P = parameters.P;
I = parameters.I;

% define costate equation
lambdadot = (I-a*P')*lambda - x + a*(P+P')*x + b + u;

end
```

B.5.3 Optimal controllers

```
function u = controlEq(x,lambda,parameters)

% initialize parameters
alpha = parameters.alpha;

% define optimal controller equations for constrained and penalized
% intervention
if alpha == 0
    u = -parameters.uBound.*sign(lambda-x);
else
    u = (x-lambda)/(2*alpha);
end

end
```

B.5.4 Function used for simulating the static and dynamic intervention policies

```
function [sys,u] = dynFun(y,uType,parameters,xopt)

% initialize model parameters
dim = parameters.dim;
a = parameters.a;
b = parameters.b;
P = parameters.P;
I = parameters.I;
x = y(1:dim);
u = y(1+dim:2*dim);

if uType == "uSOL"
    u = (I-a*P)*xopt -b;
    udot = zeros(dim,1);
elseif uType == "uS"
    u = a*P'*x;
    udot = zeros(dim,1);
elseif uType == "uD"
    udot = -(x-xopt);
elseif uType == "u0"
    u = zeros(dim,1);
    udot = zeros(dim,1);
end

%formulate diferential equations
xdot = (-I + a*P)*x + b + u;
sys = [xdot ; udot];
end
```