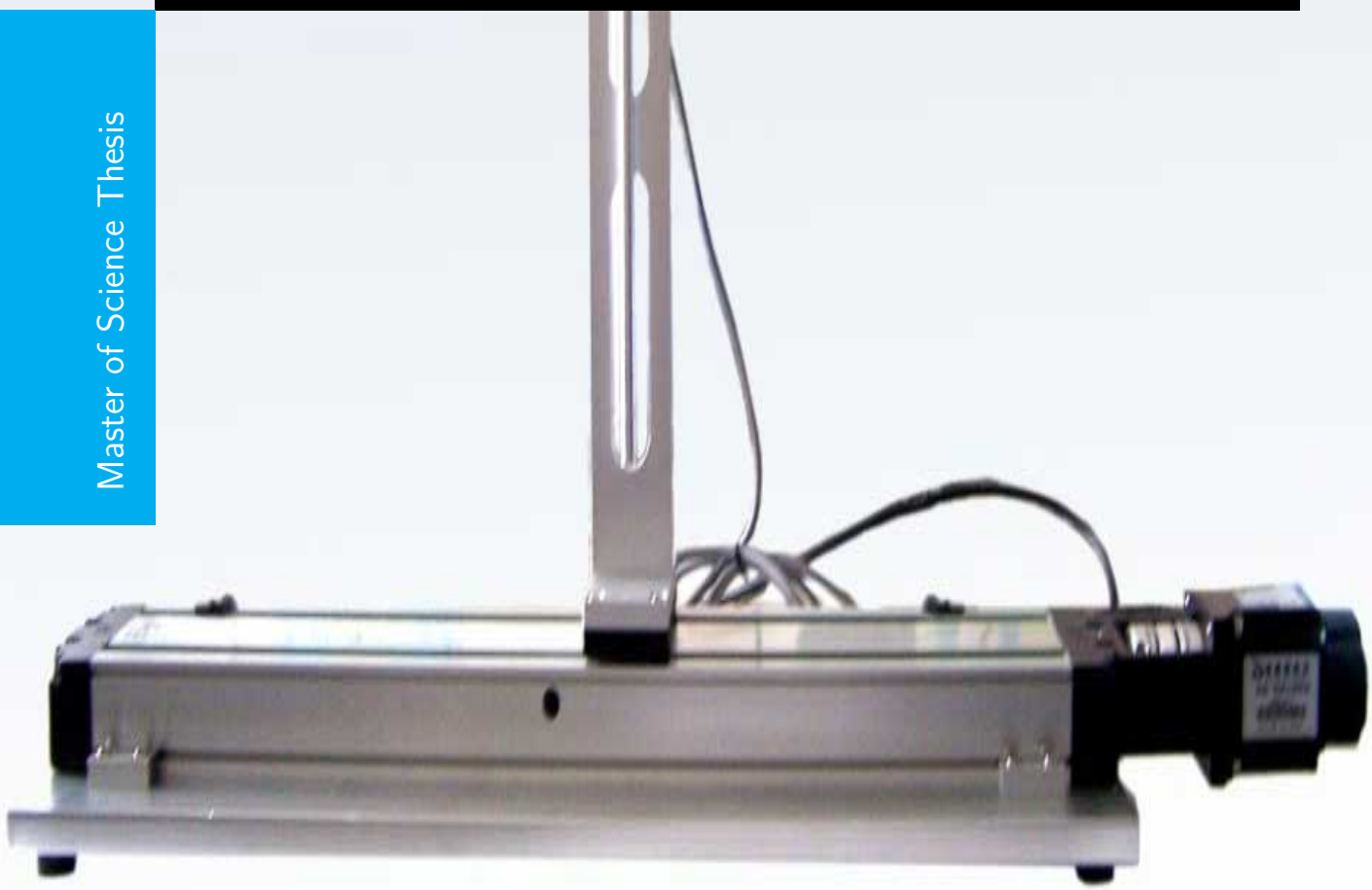
Optimal Swing-Up Control of an Inverted Pendulum

Yuzhang Wang

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Optimal Swing-Up Control of an Inverted Pendulum

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Yuzhang Wang

June 10, 2016

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Inverted pendulums have been classic setups in the control laboratories since the 1950s. They were originally used to illustrate ideas in linear control such as stabilization of unstable systems. Because of their nonlinear nature, pendulums have maintained their usefulness and they are now used to illustrate many of the ideas emerging in the field of nonlinear control. Typical examples are feedback stabilization, variable structure control, passivity based control, back-stepping and forwarding, nonlinear observers, friction compensation, and nonlinear model reduction. Pendulums have also been used to illustrate task oriented control such as swinging up and excellently suited to illustrate hybrid systems and control of chaotic systems.

The purpose of this project is to apply, compare and extend some recently developed methodologies for optimal control with input saturation constraints to the problem of control of an inverted pendulum. In particular we will focus on two strategies: a model based actor-critic strategy, and a sum-of-squares based control strategy. All these strategies aim, from different perspectives, at optimal control of nonlinear systems. For this reason, to evaluate and compare the performance of the algorithms, I will apply them to the task of swinging up an inverted pendulum. The swing-up task is chosen because of its low-dimensional, but challenging, highly nonlinear nature. As the process has two states and one action, it allows for easy visualization of the functions of interest (value function, control policy, phase plane).

The numerical simulations show that in model learning methods for actor-critic control, although ideally the neural network (NN) approximation can approximate any smooth function to arbitrary precision, it looks like model based actor-critic (MBAC) algorithm is not always able to reach the optimum. In contrast to the MBAC approach, the nonlinear policy iteration approach guarantees that every new control policy will be stabilizing and generally lead to a monotonically decreasing cost, whereas in the MBAC approach neither stabilization nor monotonic convergence can be guaranteed. In particular, in the MBAC approach, it is observed that the best value function is not always corresponding to the last one.

Table of Contents

List of Figures

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Dr. ir. S. Baldi for the continuous support of my graduation project, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing this thesis. I could not have imagined having a better supervisor and mentor for my final graduation research study and it was my honor to learn from him directly.

Besides my supervisor, I would like to thank the rest of my thesis committee members: Prof. dr. ir. J. Hellendoorn, Dr. ir. W. Mugge and MSc. S. Yuan for their encouragement, insightful comments, hard questions and positive feedback.

I need thank my friends at Delft University of Technology for their support during my master study. In particular, I am grateful to my girlfriend for her patience and love.

Last but not the least, I would like to thank my family: my parents, for giving birth to me at the first place and supporting me spiritually throughout my life.

Delft, University of Technology Yuzhang Wang
June 10, 2016

Chapter 1

Introduction

For several decades, inverted pendulum systems have served as excellent test beds for control theory. Because they exhibit nonlinear, unstable, non-minimum phase dynamics, and control objectives are always challenging [4]. They were originally used to illustrate ideas in linear control such as stabilization of unstable systems [5] [6]. Because of their nonlinear nature pendulums have maintained their usefulness and they are now used to illustrate many of the ideas emerging in the field of nonlinear control. Typical examples are feedback stabilization, variable structure control [7], passivity based control [8], back-stepping and forwarding [9], nonlinear observers [10], friction compensation [11], and nonlinear model reduction. Pendulums have also been used to illustrate task oriented control such as swinging up and catching the pendulum [12] [13] [14] and excellently suited to illustrate hybrid systems [15] and control of chaotic systems [16].

In this thesis, the swing-up controller, which swings the pendulum from the pointing-down initial position to the unstable upright position "as quickly as possible" and stabilize it in this position is studied. The term "as quickly as possible" is quantified in term of a cost function to be minimized. The (fully measurable) state x consists of the angle ϕ of the pendulum and the angular velocity $\dot{\phi}$ of the pendulum.

In this thesis, two strategies are studied: a Neural Network based strategy and a Sum-of-Squares based strategy. The first strategy is based on Model Based Actor-Critic (MBAC) method, which is such a learning method that the user sets a certain goal by specifying a suitable reward function for the controller, and the controller then learns to maximize the cumulative reward received over time (the value function) in order to reach that goal. The optimal value function and the optimal control policy are approximated via two Neural Networks (NNs), a critic NNs and an actor NNs respectively.

For the second strategy, this thesis revises a policy iteration procedure for the synthesis of optimal and global stabilizing control policies for Linear Time Invariant (LTI) Asymptotically Nullcontrollable with Bounded Inputs (ANCBI) systems. This class includes systems with eigenvalues on the imaginary axis (possibly repeated) but no pole with positive real part. An important aspect of the applied piecewise policy is that at each step of the iteration,

the computed policy is globally stabilizing and the existence of an improving value function should be guaranteed as well. The solution to Lyapunov function which is required to hold at each step of the policy iteration, is obtained by solving Sum-of-Squares Programs (SOSP) that can be efficiently implemented with semidefinite programming (SDP) solvers.

1-1 Research Motivation

Many processes in industry can potentially benefit from control algorithms that learn to optimize a certain cost function. However, due to limited research, the following problems remain open.

A. Nonlinearity of the inverted pendulum system
The piecewise policy iteration of optimal control laws are defined for linear systems, but the motion equation of the inverted pendulum system contains the term $sin(\phi)$, where ϕ is the angle of the pendulum. This makes the system nonlinear.

B. Convergence of the value function
It is not clear whether a value function coming from a nonlinear policy iteration method will converge (and how fast converge) to the optimal value function.

1-2 Goals of Master Thesis

This master thesis aims to apply and compare two methods of optimal control of nonlinear systems, by using an inverted pendulum as an example. Furthermore, the two main contributions of this work are the following.

A. Extend piecewise policy iteration from linear systems to nonlinear systems
The piecewise policy iteration for input-saturated systems have been defined for linear systems. However, the inverted pendulum system is nonlinear. Therefore, in this work we need extend piecewise policy iteration from linear systems with input-saturation to nonlinear systems with input-saturation. Algorithmic solutions based on polynomial tools have been recently proposed [17].

B. Evaluate nonlinear polynomial approximation of value function
In order to assess convergence to the optimal value function, in this work we will compare neural network value function approximation with nonlinear polynomial approximation. In addition, the stability and convergence results, which have been proposed in the linear unsaturated case, will be checked in the nonlinear saturated case.

1-3 Research Approach and Process

The thesis is organized as follows,

- Chapter 2 presents the method of model based actor-critic control (MBAC). This method is briefly explained for comparison purposes. Neural networks are used to approximate the actor and critic.
- Chapter 3 presents the method of policy iterations for synthesis of optimal control laws in linear systems with input-saturation.
- Chapter 4 exploits the results from analysis of non-polynomial systems using the sum of squares decomposition. This results are used to construct polynomial Lyapunov functions and extend policy iteration from linear systems to nonlinear systems with input-saturation.
- Chapter 5 applies these two optimal control methods to the case study of swing-up control of inverted pendulum. The performance of these two methods are evaluated and compared in terms of convergence and the minimization of cost.
- Chapter 6 gives final conclusions and suggestions for future work.

Chapter 2

Model Based Method for Actor-Critic Control

In this chapter, the reinforcement learning approach to optimal control is briefly explained for comparison purposes. The presentation of the model learning for actor-critic control method follows the paper [1]. This chapter is organized as follows, Section 2.1 presents the Reinforcement Learning (RL) principles. Section 2.2 presents "actor-critic" techniques, a class of reinforcement learning methods which learn a critic function (value function) and a separate actor function (policy function). The neural networks used to approximate the value function and the policy are discussed in Section 2.3. In Section 2.4 we present the model based actor-critic (MBAC) method, a modified version of model learning actor-critic (MLAC) algorithm by assuming that the process model is known.

2-1 Reinforcement Learning

Reinforcement learning is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. The problem, due to its generality, is studied in many other disciplines, such as control theory. In the operations research and control literature, the field where reinforcement learning methods are studied is called approximate dynamic programming. The problem has been studied in the theory of optimal control, though most studies are concerned with the existence of optimal solutions and their characterization, and not with the learning or approximation aspects.

In machine learning, the environment is typically formulated as a Markov decision process (MDP) as many reinforcement learning algorithms for this context utilize dynamic programming techniques. The main difference between the classical techniques and reinforcement learning algorithms is that the latter do not need knowledge about the MDP and they target large MDPs where exact methods become infeasible. The basic reinforcement learning model consists of:

- a set of environment states S
- a set of actions A
- rules of transitioning between states
- rules that determine the scalar immediate reward of a transition
- rules that describe what the agent observes

A reinforcement learning agent interacts with its environment in discrete time steps. At each time t, the agent receives an observation o_t, which typically includes the reward r_t. Then it chooses an action a_t from the set of actions available, which is subsequently sent to the environment. The environment moves to a new state s_{t+1} and the reward r_{t+1} associated with the transition (s_t, a_t, s_{t+1}) is determined. The goal of a reinforcement learning agent is to collect as much reward as possible. The agent can choose any action as a function of the history and it can even randomize its action selection.

In this chapter, the RL problem can be described as a Markov decision process (MDP). We use RL in a deterministic setting, and hence, we will present the deterministic description. The MDP is defined by the tuple $M\,(X, U, f, \rho)$, where X is the state space, U is the action space, $f : X \times U \mapsto X$ is the state transition function, and $\rho : X \times U \mapsto \mathbb{R}$ is the reward function.

The process to be controlled can be described by the state transition function $f : X \times U \mapsto X$, which returns the state x_k where reaches from state x_{k-1} after applying action u_{k-1}. Then after each transition, the controller receives a scalar reward $r_k \in \mathbb{R}$,given by the reward function $r_k = \rho(x_{k-1}, u_{k-1})$. The actions are chosen according to the policy $\pi : X \mapsto U$. So the goal in RL is to find a policy, such that make the sum of future rewards is maximized. This sum is stored in a value function $V^\pi : X \mapsto \mathbb{R}$, which is defined as

$$V^\pi(x) = \sum_{j=0}^{\infty} \gamma^j r_{k+j+1} \qquad \text{with } x_k = x \tag{2-1}$$

where $\gamma \in [0, 1)$ is the so-called discount-factor. Since the undiscounted return is a special case of the discounted return, from now on we will assume discounting. Although this looks innocent enough, discounting is in fact problematic if one cares about online performance. This is because discounting makes the initial time steps more important. Since a learning agent is likely to make mistakes during the first few steps after its "life" starts, no uninformed learning algorithm can achieve near-optimal performance under discounting even if the class of environments is restricted to that of finite MDPs.

This function satisfies the Bellman equation [18]

$$V^\pi(x) = \rho(x, \pi(x)) + \gamma V^\pi(x') \tag{2-2}$$

where x' is given by the state transition function while using the policy π, i.e., $x' = f(x, \pi(x))$. The Bellman equation is the basis upon which RL can improve the policy π. In continuous (or infinite discrete) state and action spaces, it is necessary to approximate the exact value function V^π and the exact policy π with function approximators.

The problem then is to specify an algorithm that can be used to find a policy with maximum expected return. From the theory of MDPs it is known that, without loss of generality, the search can be restricted to the set of the so-called stationary policies. A policy is called stationary if the action-distribution returned by it depends only on the last state visited. In fact, the search can be further restricted to deterministic stationary policies. A deterministic stationary policy is one which deterministically selects actions based on the current state. Since any such policy can be identified with a mapping from the set of states to the set of actions, these policies can be identified with such mappings with no loss of generality. Therefore, in continuous (or infinite discrete) state and action spaces, it is necessary to approximate the exact value function V^π and the exact policy π with function approximators.

2-2 Actor-Critic RL

Actor-critic techniques which were introduced in [19] are characterized by learning separate functions for the actor (the policy π) and the critic (the value function V^π). Actor-critic methods belong to the class of policy gradient methods. In these methods, the policy is represented by a differentiable parameterization, and gradient updates are performed to find the parameters that lead to (locally) maximal returns [20]. The critic takes the role of the value function and evaluates the performance of the actor, hereby helping with the estimation of the gradient to use for the actor's updates. The use of gradient-based policy updates makes actor-critic techniques suitable for continuous action spaces [21]. However, a few problems with this procedure are as follows:

- The procedure may waste too much time on evaluating a suboptimal policy.
- It uses samples inefficiently in that a long trajectory is used to improve the estimate only of the single state-action pair that started the trajectory.
- When the returns along the trajectories have high variance, convergence will be slow.
- It works in small, finite MDPs only.

The first issue is easily corrected by allowing the procedure to change the policy (at all, or at some states) before the values settle. However good this sounds, this may be problematic as this might prevent convergence. Still, most current algorithms implement this idea, giving rise to the class of generalized policy iteration algorithm. We note in passing that actor-critic methods belong to this category. The second issue can be corrected within the algorithm by allowing trajectories to contribute to any state-action pair in them. This may also help to some extent with the third problem, although a better solution when returns have high variance is to use temporal difference (TD) methods which are based on the recursive Bellman equation.

Thus, in this thesis, a temporal-difference-based actor-critic method serves as a baseline to compare our new method to. We will refer to this baseline as the standard actor-critic (S-AC) algorithm. As shown in Figure 2-1, it describes how different entities interact within the S-AC algorithm.

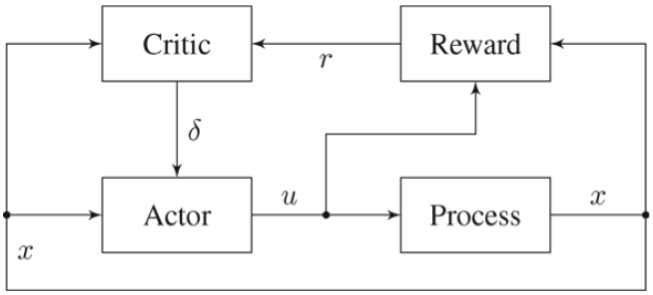


Figure 2-1: Block diagram of the S-AC algorithm

Methods based on temporal differences also overcome the second but last issue. In order to address the last issue mentioned before, function approximation methods are used. Denoting the approximate value function parameterized by the vector θ with $V(x, \theta)$, the temporal difference is defined as

$$\delta_k = r_k + \gamma V(x_k, \theta_{k-1}) - V(x_{k-1}, \theta_{k-1}) \tag{2-3}$$

This is the difference between the right-hand and left-hand sides of the Bellman equation (2-2). The goal is to make the approximation of the critic satisfy the Bellman equation. By using the temporal difference, the gradient-descent update rule for the critic parameter vector θ is

$$\theta_k = \theta_{k-1} + \alpha_c \delta_k \left. \frac{\partial V(x, \theta)}{\partial \theta} \right|_{\substack{x=x_{k-1} \\ \theta=\theta_{k-1}}} \tag{2-4}$$

where $\alpha_c > 0$ is the learning rate of the critic. This parameter update adapts the approximate value function such that the error between the approximated and real values at the state considered is minimized.

Using (2-4) to update the critic results in a one-step backup, whereas the reward received is often the result of a series of steps. Eligibility traces offer a better way of assigning credit to states visited several steps earlier. The eligibility trace for a certain state x at time instant k is denoted with $e_k(x)$

$$e_k(x) = \begin{cases} 1 & \text{if } x = x_k \\ \lambda\gamma e_{k-1}(x) & \text{otherwise.} \end{cases}$$

It decays with time by a factor $\lambda\gamma$, with $\lambda \in [0, 1)$ being the trace decay rate. This makes more recently visited states more eligible for receiving credit. All states along the trajectory now influence the update of θ with the following equation:

$$\theta_k = \theta_{k-1} + \alpha_c \delta_k \sum_{x_v \in \mathcal{X}_v} \left. \frac{\partial V(x, \theta)}{\partial \theta} \right|_{\substack{x=x_v \\ \theta=\theta_{k-1}}} e_k(x_v) \tag{2-5}$$

where $\mathcal{X}_v$ denotes the set of states visited during the current trial. The use of eligibility traces speeds up the learning considerably.

The approximate policy is parameterized by ϑ with $\pi(x, \vartheta)$ and indicates the action to take in a state x. However RL requires the use of exploration to keep trying new, possibly better,

actions in the states encountered. With exploration, the control action u_k is different from the action indicated by the policy. This can be achieved by perturbing the action with a zero-mean random exploration term Δu_k,

$$u_k = \pi(x_k, \vartheta_{k-1}) + \Delta u_k \tag{2-6}$$

When the exploration Δu_k leads to a positive temporal difference ($\delta_k > 0$), the policy is adjusted toward this perturbed action. Conversely, when $\delta_k < 0$, the policy is adjusted away from this perturbation. So this will lead to the following update rule for the actor:

$$\vartheta_k = \vartheta_{k-1} + \alpha_a \delta_k \Delta u_{k-1} \left. \frac{\partial \pi(x, \vartheta)}{\partial \vartheta} \right|_{\substack{x=x_{k-1} \\ \upsilon=\vartheta_{k-1}}} \tag{2-7}$$

where $\alpha_a > 0$ is the learning rate of the actor. Then the temporal difference is interpreted as a correction of the predicted performance; if $\delta_k > 0$, the obtained performance is considered better than the predicted one. Finally, the full implementation of the S-AC algorithm is shown in the following Figure 2-2.

Algorithm 1 S-AC

Input: γ, λ, α_c, α_a
$e_0(x) = 0 \quad \forall x$
Initialize x_0, θ_0 and ϑ_0
Apply random input u_0
$k \leftarrow 1$
loop
    Choose Δu_k at random
    Measure x_k, r_k
    $u_k \leftarrow \pi(x_k, \vartheta_{k-1}) + \Delta u_k$
    Apply u_k
    $\delta_k \leftarrow r_k + \gamma V(x_k, \theta_{k-1}) - V(x_{k-1}, \theta_{k-1})$
    $e_k(x) = \begin{cases} 1, & \text{if } x = x_k \\ \lambda \gamma e_{k-1}(x), & \text{otherwise} \end{cases}$
    $\theta_k \leftarrow \theta_{k-1} + \alpha_c \delta_k \sum_{x \in \mathcal{X}_v} (\partial V(x, \theta)/\partial \theta) e_k(x)$
    $\vartheta_k \leftarrow \vartheta_{k-1} + \alpha_a \delta_k \Delta u_{k-1} (\partial \pi(x, \vartheta)/\partial \vartheta)$
    $k \leftarrow k + 1$
end loop

Figure 2-2: S-AC algorithm as present in [1]

2-3 Function Approximation

In actor-critic methods, both the policy and the value function are represented using function approximation techniques. The algorithms use neural networks (NNs) as a function approximator. NNs is a parametric memory-based method for approximating nonlinear functions. Memory-based methods are also called case based, exemple based, lazy and instance based, or experience based [22]. It has been shown that memory-based learning can work in RL and

can quickly approximate a function with only a few observations [23]. This is particularly useful at the start of learning.

The main advantage of memory-based methods is that the user does not need to specify a global structure or predefine features for the (approximate) model. Instead of trying to fit a global structure to observations of the unknown function, NNs simply stores the observations in a memory. A stored observation is called a sample $s_i = [x_i^T | y_i^T]^T$ with $i = 1, ..., N$. One sample s_i is a column vector containing the input data $x_i \in \mathbb{R}^n$ and output data $y_i \in \mathbb{R}^m$. The samples are stored in a matrix called the memory M with size $(n + m) \times N$ whose columns each represent one sample.

When a query x_q is made, NNs uses the stored samples to give a prediction $\hat{y}_q$ of the true output y_q. The prediction is computed by finding a local neighborhood of x_q in the samples stored in memory. This neighborhood is found by applying a weighted distance metric d_i (e.g., the 1-norm or 2-norm) to the query point x_q and the input data x_i of all samples in M. The weighting W is used to scale the inputs x and has a large influence on the resulting neighborhood and thus on the accuracy of the prediction. Note that, in this thesis, the input samples x_i are actually stored in the NNs memories as weighted samples Wx_i. Searching through the memory for nearest neighbor samples is computationally expensive. Here, a simple sorting algorithm was used, but one can reduce the computational burden by using, for instance, $k - d$ trees [24].

Thus, there are three steps to get the prediction $\hat{y}_q$. In the first step, by selecting a limited number of K samples with the smallest distance d, we create a subset $\mathcal{K}(x_q)$ with the indices of nearest neighbor samples. Only these K nearest neighbors are then used to make a prediction of $\hat{y}_q$. The prediction is computed by fitting a linear model to these nearest neighbors. Applying the resulting linear model to the query point x_q yields the predicted value $\hat{y}_q$. In addition, the matrices X and Y need to be constructed using the K nearest neighbor samples

$$Y = \begin{bmatrix} y_1 & y_2 & ... & y_K \end{bmatrix}$$
$$X = \begin{bmatrix} x_1 & x_2 & ... & x_K \\ 1 & 1 & ... & 1 \end{bmatrix}$$

The X and Y matrices form an overdetermined set of equations for the model parameter matrix $\beta \in \mathbb{R}^{m \times (n+1)}$

$$Y = \beta X$$

then in the second step, it can be solved by the least squares method using the right pseudoinverse of X,

$$\beta = YX^T(XX^T)^{-1}$$

Finally, the model parameter matrix β is used to compute the prediction for the query x_q

$$\hat{y}_q = \beta x_q$$

As a result, the globally nonlinear function is approximated locally by a linear function. At the start of a trial, the matrices X and Y will not yet form a fully determined set of equations. In this case, there are infinitely many solutions, and β will be chosen as the solution with the smallest norm.

In addition, the use of NNs comes with a few assumptions. The first and foremost one is that the approximated function should be smooth enough so that it can be captured by locally linear models. Any function with discontinuities or other nonsmooth behavior will be tough to approximate. This also depends on the maximum possible number of samples in the NNs memory. This number should be large enough so that the neighborhood in which a locally linear model is calculated is small enough, i.e., the linear model is indeed local enough. More specifically, when applying NNs in RL algorithms, the sampling time used should be small enough so that a locally linear model calculated at one time step is still good enough at the next time step. This is because we also use the model for predictions at the next time step.

2-4 Model Based Actor-Critic Method

In this section, a modified version of model learning actor-critic (MLAC) algorithm is introduced. The MLAC algorithm was proposed in paper [1] to learn a process model in addition to the actor and critic. The actor update is done using a policy gradient calculated from a local gradient of the critic and a local gradient of the learned process model. In this thesis, we slightly modify the algorithm by assuming that the process model is known. We refer to this algorithm as model based actor-critic (MBAC). In the implementation of the algorithms, we always use NNs to learn and approximate the functions involved.

The actor and critic are updated by inserting the last observed sample into the memory, as the most up-to-date knowledge should be incorporated in any approximation calculated from the memory as following,

- The critic memory M^C holds samples of the form $s_i = [x_i^T | V_i]^T$ with $i = 1, ..., N^C$;
- The actor memory M^A has samples $s_i = [x_i^T | u_i^T]^T$ with $i = 1, ..., N^A$.

During the learning process, the actor, critic are updated by adjusting the output parts of the nearest neighbor samples s_i that relate to the query point x_q.

In addition to learning the actor and critic functions, the MBAC method uses the process model $x' = \hat{f}(x, u)$. The available process model simplifies the update of the actor, as it allows us to predict what the next state x' will be, given some input u. Together with the approximate value function, this allows us to obtain information on the value $V(x')$ of the next state x'. This means that we can choose the input u such that $V(x')$ is optimal.

In Figure 2-3, it shows the scheme of MBAC. The solid lines indicate actual signals. The dashed lines indicate the use of a local linear model or gradient from a particular entity.

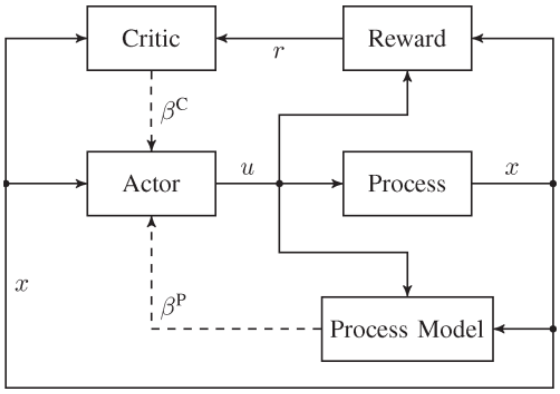


Figure 2-3: Block diagram of the MBAC algorithm

However, since we assume that our action space is continuous, we cannot enumerate over all possible inputs u and therefore choose to put a policy gradient in place. Hence, by virtue of neural networks (NNs), we can easily estimate the gradient of the value function with respect to the state x and the gradient of the process model with respect to the input u. Then after applying the chain rule, we have a gradient of the value function with respect to the input u available and use this to update the actor.

As a result, the actor is updated by multiplying the local gradients of the value function and of the process model to obtain a gradient of the value function with respect to chosen input u. By adjusting the input u in the direction given by this gradient and saturating the result element-wise such that the new input lies in the allowed input range $[u_{min},\ u_{max}]$, the actor is trying to maximize $V(x')$

$$u_i \leftarrow \text{sat}\left\{u_i + \alpha_a \frac{\partial V}{\partial x}\bigg|_{x=x'} \frac{\partial x'}{\partial u}\right\} \qquad \forall i \in \mathcal{K}(x) \tag{2-8}$$

Recall that x' is given by the state transition function $x' = f(x, u)$.

The value function is approximated by NNs which estimates a local linear model on the basis of previous observations of $V(x)$. The local linear model is of the form

$$V(x) = \beta^C \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} = \begin{bmatrix} \beta_x^C & \beta_b^C \end{bmatrix} \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} \tag{2-9}$$

where β_x^C which is the part of β^C, that is the gradient $\frac{\partial V}{\partial x}$ relates the input x to the output V. The gradient $\frac{\partial x'}{\partial u}$ can be found by NNs on previous observations of the process dynamics.

The process model is linearized in the form

$$x' = \hat{f}(x, u) = \beta^P \cdot \begin{bmatrix} x \\ u \\ 1 \end{bmatrix} = \begin{bmatrix} \beta_x^P & \beta_u^P & \beta_b^P \end{bmatrix} \cdot \begin{bmatrix} x \\ u \\ 1 \end{bmatrix} \tag{2-10}$$

where β_u^P which is the part of β^P, that is the gradient $\frac{\partial x'}{\partial u}$ relates u to x'. So we can now use β_x^C, β_x^P and (2-8) to improve the actor by adapting the nearest neighbor samples with

$$u_i \leftarrow \text{sat}\{u_i + \alpha_a \beta_x^C \beta_u^P\} \qquad \forall i \in \mathcal{K}(x) \tag{2-11}$$

The pseudocode is found in Algorithm 2. In the pseudocode, the set $\mathcal{K}_+(x_q)$ is $\mathcal{K}(x_q)$, extended with the index where the sample representing x_q was inserted.

Algorithm 2 MBAC

Input: γ, λ, α_c, α_a
Initialize x_0, M^C, M^A and M^P
$V_0 = 0$, $\beta^C = 0$
$e_0(s_i) = 0 \quad \forall s_i \in M^C$
Apply random input u_0
$k \leftarrow 1$
loop
  Choose Δu_k at random
  Measure x_k, r_k
  Obtain β^A from M^A for x_k
  $u_k \leftarrow \beta^A \cdot [x_k^T \; 1]^T + \Delta u_k$
  Apply u_k
  % Linearlize process model
  x' $= \beta^P \cdot$ [x u 1]^T
  % Update actor
  Insert $[x_{k-1}^T | (u_{k-1} + \alpha_a \beta_x^C \beta_u^P)^T]^T$ in M^A
  for $\forall i \in \mathcal{K}(x_{k-1})$ of M^A **do**
    $u_i \leftarrow \text{sat}\left\{u_i + \alpha_a \beta_x^C \beta_u^P\right\}$
  end for
  % Update critic
  Obtain β^C from M^C for x_k
  $V_k \leftarrow \beta^C \cdot [x_k^T \; 1]^T$
  Insert $[x_{k-1}^T | V_{k-1}]^T$ in M^C
  $\delta_k \leftarrow r_k + \gamma V_k - V_{k-1}$
  for $\forall s_i \in M^C$ **do**
    $e_k(s_i) = \begin{cases} 1, & \text{if } i \in \mathcal{K}_+(x_{k-1}) \\ \lambda\gamma e_{k-1}(s_i), & \text{otherwise} \end{cases}$
    $V_i \leftarrow V_i + \alpha_c \delta_k e_k(s_i)$
  end for
  $k \leftarrow k + 1$
end loop

Figure 2-4: MBAC algorithm

2-5 Partial Conclusions

This chapter has explained the model based actor-critic method (MBAC), which uses NNs as a parametric memory-based function approximator. The MBAC uses a process model and employs it to update the actor. However, instead of using the process model to generate simulated experiences as most model learning RL algorithms do [25] [26] [27], it uses the model to directly calculate an accurate policy gradient, which accelerates learning compared to other policy gradient methods. This novel algorithm uses neural networks (NNs) to approximate the actor and critic. And memory-based learning methods have successfully been applied to RL before, mostly as an approximator for the value function [23] [28] and in some cases, also for the process model [29] [30]. Although it is not exploited in this thesis, one benefit of the memory-based function approximators is that they can easily be initialized with samples of prior knowledge.

Since the NN approximation allows to approximate any smooth function, the MBAC method can be applied to any nonlinear process, eventually in the presence of input-saturation. For this reason, the MBAC algorithm is a good candidate for the optimal swing up control of a pendulum.

Chapter 3

Piecewise Policy Iterations in Linear Systems with Input-Saturation

As we know, the Hamilton-Jacobi-Bellman (HJB) equation is in general hard to solve and the most celebrated method for solving it is Dynamic Programming (DP)[31]. However, the computational burden associated to DP is prohibitive as the dimension of the problem increases, which identified by Bellman himself with the term 'curse of dimensionality'. Hence, adaptive dynamic programming (ADP) recalls different methods to solve the problem of 'curse of dimensionality' by approximating the solution of the HJB equation [32] [33]. Among the several ADP techniques, this chapter recalls a policy iteration procedure for the synthesis of optimal and globally stabilizing control policies for Linear Time Invariant (LTI) Asymptotically Null-controllable with Bounded Inputs (ANCBI) systems.

This chapter is organized as follows. Section 3.1 presents the control problem formulation for the class of LTI-ANCBI systems in the presence of input saturation and some preliminary results which are useful to the synthesis of optimal control laws. Section 3.2 presents the policy iteration method under saturation constraints, which is composed respectively of policy evaluation with piecewise value functions and of piecewise policy improvement. In Section 3.3, numerical formulation of the proposed conditions are relaxed to Sum-of-Squares conditions that can be implemented via Semi-Definite Programming.

3-1 Problem Formulation

We study the class of LTI Asymptotically Null-controllable with Bounded Inputs (ANCBI) systems in the presence of input saturation, consisting of dynamic linear systems without exponentially unstable modes. It is worth mentioning that for linear systems with exponentially unstable modes it is not possible to achieve global stabilization with bounded inputs [34].

Consider the input-saturated system

$$\dot{x} = Ax + Bsat\left(u(x)\right), \quad x(0) = x_0, \tag{3-1}$$

with $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$, $\max(\lambda(A)) \leq 0$.

The function $sat : \mathbb{R}^m \to \mathcal{U}$ is a vector saturation function, with entries satisfying

$$\left(sat\left(u(x)\right)\right)_j = \begin{cases} \overline{u}_j, & \text{if} \quad u_j \succ \overline{u}_j \\ u_j, & \text{if} \quad \underline{u}_j \preceq u_j \preceq \overline{u}_j \\ \underline{u}_j, & \text{if} \quad \underline{u} \succ u_j \end{cases} \tag{3-2}$$

and the set of inputs is defined as

$$\mathcal{U} := \left\{ u \in \mathbb{R}^m | \, \underline{u}_j \preceq u \preceq \overline{u}_j, j = 1, \ldots, m \right\}.$$

then we introduce the dead-zone function

$$dz\left(u(x)\right) := u(x) - sat\left(u(x)\right)$$

and rewrite the system (3-1) as

$$\dot{x} = Ax + Bu(x) - Bdz\left(u(x)\right), \quad x(0) = x_0. \tag{3-3}$$

In order to have a well-posed problem we make the assumption: there exists a globally stabilizing control policy $\bar{u}$.

We also introduce a discounted cost function for the system (3-3) of the form

$$J = \int_0^\infty e^{-\lambda t} L(x,u)dt = \int_0^\infty e^{-\lambda t} \left[x'Qx + sat'\left(u(x)\right) Rsat\left(u(x)\right)\right] dt. \tag{3-4}$$

So the objective of the control problem can be stated as:

Problem 1: given the input-saturated system (3-3), the goal is to design the optimal control policy $u^o(x)$, such that the origin of the system (3-3) is globally asymptotically stable, and the cost function (3-4) with $u(x) = u^o(x)$ is minimized.

After defining the control problem, some results which are preliminary to the development of a policy iteration mechanism are presented in the following.

Firstly, we introduce sector condition relate to the deadzone function $q(x) = dz\left(u(x)\right)$,

$$q'(x)\left(u(x) - q(x)\right) \geq 0, \quad \forall\, x \in \mathbb{R}^n. \tag{3-5}$$

Furthermore, define $\phi(x) := \frac{d\, dz(u(x))}{dt}$ satisfying

$$\phi(x) = \begin{cases} 0 & \text{if } q(x) = 0 \\ \dot{u}(x) & \text{if } q(x) \neq 0, \end{cases} \tag{3-6}$$

which can be expressed by the two equalities

$$\phi'(x)\left(\dot{u}(x) - \phi(x)\right) = 0 \tag{3-7}$$

$$q'(x)\left(\dot{u}(x) - \phi(x)\right) = 0 \tag{3-8}$$

In order to find optimal control policies with respect to the cost (3-4), we adopt the well-known result from optimal control theory [35], that states that the optimal control policy $u^o(x)$ that minimizes (3-4) satisfies

$$u^o = \arg\min_{u(\cdot)\in\mathcal{U}} \left\{ -\lambda V^o + \frac{d\,V^o}{dx}\left(Ax + Bu\right) + L(x,u) \right\}, \tag{3-9}$$

where $V^o(x)$ is the value function that solves the Hamilton-Jacobi-Bellman equation

$$\min_{u(\cdot)\in\mathcal{U}} \left\{ -\lambda V^o + \frac{d\,V^o}{dx}\left(Ax + Bu\right) + L(x,u) \right\} = 0. \tag{3-10}$$

As mentioned before, the Hamilton-Jacobi-Bellman (HJB) equation is in general hard to solve and the most celebrated method for solving it is Dynamic Programming (DP). However it also has some constraints, such as

- Analytical solutions only in special cases (e.g. linear systems).
- Numerical solutions require prohibitive numerical burden.

So we adopt Adaptive Dynamic Programming (ADP) in the next section. where we apply an iterative strategy to solve Problem 1 based on approximate solutions to (3-10).

3-2 Policy Iterations under Saturation Constraints

Apart from some special cases, the HJB equation (3-10) is hard to solve, and even numerical solutions obtained with DP may require prohibitive computational burden. The policy iteration method is typically used as an iterative approach to compute approximate solutions to the HJB equation [36] [37]. Here we first applies this method of the unsaturated condition and then the main difficulties related to the extension to input-saturated plants will be discussed.

Given an LTI plant

$$\dot{x} = Ax + Bu, \quad x(0) = x_0, \tag{3-11}$$

and a discounted cost function in the form

$$J_{un} = \int_0^\infty e^{-\lambda t}\left[x'Qx + u'Ru\right] dt \tag{3-12}$$

An approximate solution to (3-10) can be obtained with the iterative method sketched in the following Algorithm 1.

Algorithm 1 Policy iteration

1: *Initialize*:
2: $c \leftarrow 0$.
3: $u^c \leftarrow u^0$.
4: *Policy evaluation*:
5: Given $u^c(x)$ **solve** for $V^c(x)$
6:
$$-\lambda V^c(x) + \frac{d\,V^c(x)}{dx}(Ax + Bu^c) + x'Qx + u^{c'}Ru^c = 0.$$
7: *Policy improvement*:
8: **Update** the control policy
9:
$$u^{c+1} = -\frac{1}{2}R^{-1}B'\frac{d\,V^{c\,\prime}}{dx}.$$
10:
11: **if** $\Delta V^c(x(0)) := V^c(x(0)) - V^{(c-1)}(x(0)) < \delta$, **STOP**
12: **else** $c \leftarrow c+1$, **goto** *Policy improvement*.

Figure 3-1: Algorithm of policy iterations under unsaturation constraints

We can see that the policy iteration technique consists of a 2-stage process: in the first step, given a control policy u^c, a value function V^c is computed. In the second step, given the value function V^c, the updated policy u^{c+1} is computed. Note that the initial policy u^0 must be admissible, which requires it to be stabilizing. In the final step, the iterative process stops upon convergence of the value function.

In the unsaturated case, it is possible to show that the presented policy iterations converge to the optimal value function V^o and the optimal globally stabilizing control policy u^o. In principle, although the fact that policy iteration techniques and Sum-of-Squares relaxations can be extended to the saturated case, it is unclear how to address the non-differentiability nature of the saturation function with polynomial value functions and whether we can guarantee similar convergence properties as the unsaturated case. In the following, we present a piecewise value function that explicitly accounts for the non-differentiability of the vector field introduced by the saturating inputs.

3-2-1 Piecewise policy evaluation

Consider a piecewise value function of the form $W^c(x, q^c(x))$ instead of $V^c(x)$

$$V^c(x) = W^c(x, dz(u^c(x))) = W^c(x, q^c(x)) \tag{3-13}$$

Given the input-saturated system (3-1), a cost function (3-4) and the above structures for the value function, the equation to be solved in policy evaluation is given by,

$$-\lambda W^c(x(t), q^c(x(t))) + \frac{d\,W^c(x(t), q^c(x(t)))}{dt} + x'Qx + (u^c - q^c)'R(u^c - q^c) = 0. \tag{3-14}$$

Since q is non-differentiable, we consider it as an independent variable, which allow us to describe its time-derivative in terms of a variable $\phi(x)$ as in (3-6), and (3-14) is modified according to

$$-\lambda W^c + \frac{\partial W^c}{\partial x}(Ax + B(u^c - q^c)) + \frac{\partial W^c}{\partial q^c}\phi^{c/c} + x'Qx + (u^c - q^c)'R(u^c - q^c) = 0 \quad (3\text{-}15)$$

3-2-2 Piecewise policy improvement

The policy improvement step consists of solving the following problem

$$\min_{u\in\mathcal{U}} \left[\frac{d\,V^c}{dx}(Ax + B(u^c - q^c)) + x'Q(x)x + (u^c - q^c)'R(u^c - q^c)\right] \quad (3\text{-}16)$$

Using the Karush-Kuhn-Tucker conditions [38], the necessary conditions for the solution of (3-16) are

$$u = -\frac{1}{2}R^{-1}\left[B'\left[\frac{\partial W^c}{\partial x} + \frac{\partial W^c}{\partial q^c}\frac{\partial dz(u^c)}{\partial x}\right]' + \mu_1(x) - \mu_2(x)\right] \quad (3\text{-}17)$$

where μ_1, μ_2 keep the input inside the saturation bounds.

$$\mu_1 = \max\left(0 \quad 2R\left(-\overline{\mu} - \frac{1}{2}R^{-1}B'\frac{d\,V'}{dx}\right)\right)$$
$$\mu_2 = \max\left(0 \quad 2R\left(\underline{\mu} + \frac{1}{2}R^{-1}B'\frac{d\,V'}{dx}\right)\right)$$

The unsaturated control policy u^{c+1} at the next policy evaluation then becomes

$$u^{c+1}(x) = -\frac{1}{2}R^{-1}B'\left[\frac{\partial W^c}{\partial x} + \frac{\partial W^c}{\partial q^c}\frac{\partial dz(u^c)}{\partial x}\right]' \quad (3\text{-}18)$$

However, the policy (3-18) can not be implemented without introducing conservativeness since the value function W^c is defined in terms of the previous policy u^c. This makes the policy improvement in (3-18) to be defined also in terms of the previous policy u^{c-1}. Therefore, the implementation of (3-18) requires control policies from previous iterations.

As a result, in order to avoid storing the iteration history required to implement (3-18), we consider the following approximate policy improvement,

$$u_{ap}^{c+1}(x) = -\frac{1}{2}R^{-1}B'\left.\frac{\partial W^{c\,\prime}}{\partial x}\right|_{q^c=0} \quad (3\text{-}19)$$

Note that (3-19) represents the policy that results from ignoring the contribution of term $\frac{\partial W^c}{\partial q^c}\frac{\partial dz(u^c)}{\partial x}$ in (3-18). As a consequence, $u_{ap}^{c+1}(x) = u^{c+1}(x)$ in the set

$$\Omega^c(x) = \{x : dz(u^c(x)) = 0\} \quad (3\text{-}20)$$

The approximated policy improvement (3-19) is thus the optimal solution to (3-16) for the unsaturated region of the previous policy $u^c(x)$. Furthermore, the policy $u_{ap}^{c+1}(x)$ in (3-19) defines a different unsaturated region as

$$\Omega^{c+1}(x) = \left\{x : dz(u_{ap}^{c+1}(x)) = 0\right\} \quad (3\text{-}21)$$

3-2-3 Modified policy iteration

We now discuss properties of the policy (3-19). Let us first define the following state-space partition defined by sets Ω^c and Ω^{c+1} in (3-20), (3-21)

$$\begin{array}{rcll} \Xi_1^c & := & \Omega^c \cap \Omega^{c+1} & \text{(Region 1)} \\ \Xi_2^c & := & \Omega^c \backslash \Omega^{c+1} & \text{(Region 2)} \\ \Xi_3^c & := & \Omega^{c+1} \backslash \Omega^c & \text{(Region 3)} \\ \Xi_4^c & := & \mathbb{R}^n \backslash (\Omega^c \cup \Omega^{c+1}) & \text{(Region 4)} \end{array} \tag{3-22}$$

satisfying $\cup_i \Xi_i^c = \mathbb{R}^n$ and $\Xi_i^c \cap \Xi_j^c = \emptyset$, $i \neq j$. Note that Ξ_1^c corresponds to the set in which both control policies respect the saturation bounds; $\Xi_2^c = \emptyset$, whenever $\Omega^c \subset \Omega^{c+1}$; $\Xi_3^c = \emptyset$, whenever $\Omega^{c+1} \subset \Omega^c$; and Ξ_4^c is the set in which both control policies exceed the saturation bounds.

In the following, we study the stability properties of the policy u_{ap}^{c+1}, given a globally stabilizing policy u^c and a value function W^c that certify global stability. To this purpose, define the piecewise policy

$$sat\left(u_{pw}^{c+1}\right) = \begin{cases} sat\left(u_{ap}^{c+1}\right) & \text{in } \Xi_1^c \cup \Xi_2^c \cup \Xi_3^c \\ sat\left(u^c\right) & \text{in } \Xi_4^c \end{cases} \tag{3-23}$$

and the value function

$$W_{pw}^c := \begin{cases} W_{un}^c & \text{in } \Xi_1^c \cup \Xi_2^c \cup \Xi_3^c \\ W^c & \text{in } \Xi_4^c \end{cases} \tag{3-24}$$

where W_{un}^c is the unsaturated value function defined as

$$W_{un}^c(x) := W^c(x, 0) \tag{3-25}$$

Then we obtain the following result,

Proposition 1: The piecewise value function (3-24) certifies the global stability of the piecewise control policy (3-23).

Notice that in the process of proof, we define

$$\mathcal{L}(W^c, u) := \frac{d}{dx}\left(W^c(x, dz(u^c(x)))\right)(Ax + Bu) \tag{3-26}$$

which represents the time-derivative of W^c along the trajectories of the system under a control policy u. Next, under assumption $\mathcal{L}(W_{un}^c, sat\left(u^c\right)) < 0$, we have

for $x \in \Xi_1^c$

$$\mathcal{L}(W^c, u_{ap}^{c+1}) = \mathcal{L}(W^c, u^c) - (u_{ap}^{c+1} - u^c)' R (u_{ap}^{c+1} - u^c) < 0 \tag{3-27}$$

for $x \in \Xi_2^c$

$$\mathcal{L}(W^c, sat\left(u_{ap}^{c+1}\right)) = \mathcal{L}(W^c, u^c) - (sat\left(u_{ap}^{c+1}\right) - u^c)' R (sat\left(u_{ap}^{c+1}\right) - u^c) < 0 \tag{3-28}$$

for $x \in \Xi_3^c$

$$\mathcal{L}(W_{un}^c, u_{ap}^{c+1}) = \mathcal{L}(W_{un}^c, sat\,(u^c)) - (u_{ap}^{c+1} - sat\,(u^c))' R (u_{ap}^{c+1} - sat\,(u^c)) < 0 \tag{3-29}$$

for $x \in \Xi_4^c$

$$\mathcal{L}(W_{un}^c, sat\left(u_{ap}^{c+1}\right)) = \mathcal{L}(W_{un}^c, u^c) - (u_{ap}^{c+1} - sat\,(u^c))' R (u_{ap}^{c+1} - sat\,(u^c)) + q^{c+1'} R q^{c+1} \tag{3-30}$$

As shown in (3-27),(3-28),(3-29), we obtain in Ξ_1^c, Ξ_2^c and Ξ_3^c,

$$\mathcal{L}(W_{un}^c, sat\left(u_{ap}^{c+1}\right)) < 0 \tag{3-31}$$

which means that, in these three regions, W_{un}^c is a feasible local Lyapunov function for the approximate policy iteration u_{ap}^{c+1}.

Figure 3-2 shows the algorithm of modified policy iteration with piecewise value function.

Algorithm 2 Modified policy iteration

Initialize:
  $c \leftarrow 0$.
  $\bar{u}_{pw}^c \leftarrow u^0$.
  $u_{pw}^c \leftarrow u^0$.
Policy evaluation:
  Given u_{pw}^c, **solve** for $V^c(x) = W^c(x, dz\,(u^c(x)))$

$$-\lambda V^c(x) + \frac{d\,V^c(x)}{dx}\left(Ax + Bsat\left(u_{pw}^c\right)\right) + L(x, u_{pw}^c) = 0$$

Feasibility:
  With $W^c(x, dz\,(u^c(x)))$ of *Policy evaluation*, **check**

$$-\lambda V^c(x) + \frac{d\,V^c(x)}{dx}\left(Ax + Bsat\,(u^c)\right) + L(x, u_{pw}^c) < 0$$

  if (34) is *feasible*, $\bar{u}^c(x) \leftarrow u^c(x)$
  else $\bar{u}^c(x) \leftarrow u^{(c-1)}(x)$
Policy improvement:
  Update the piecewise control policy

$$u_{pw}^{c+1} = \begin{cases} -\frac{1}{2} R^{-1} B' \left.\frac{\partial W^{c\,\prime}}{\partial x}\right|_{q^c=0} & \text{in } \Xi_1^c \cup \Xi_2^c \cup \Xi_3^c \\ \bar{u}^c & \text{in } \Xi_4^c \end{cases}$$

if $\Delta W^c(x(0)) := W^c(x(0)) - W^{(c-1)}(x(0)) < \delta$, **STOP**
else $c \leftarrow c+1$, **goto** *Policy improvement*.

Figure 3-2: Algorithm of modified policy iterations under saturation constraints

As a result, we got the conclusion from the consequence of (3-27),(3-28),(3-29): in Ξ_1^c, Ξ_2^c and Ξ_3^c, it is possible to find a new value function W^{c+1} that improves with respect to the unsaturated value function W_{un}^c found at the previous policy evaluation. Furthermore, in Ξ_4^c,

no stability guarantees can be given for the policy $sat\left(u_{ap}^{c+1}\right)$, the piecewise policy (3-23) and piecewise value function (3-24) are defined.

The feasibility step is done in order to check if the control policy in Ξ_4^c can be updated. If such step is feasible, the most recent policy $u^c(x)$ will be adopted in that region. Stopping criteria is defined in terms of the improvement on the value function at each iteration, that is, in terms of $\Delta W^c(x(0)) := W^c(x(0)) - W^{(c-1)}(x(0))$.

3-3 Numerical Formulation

The inequalities in the policy evaluation steps are, in general, difficult to solve. However, for the case of L and W^c being polynomial functions, the resulting polynomial inequalities are numerically tractable by restricting the search to the set of Sum-of-Squares (SOS) polynomials. The cone of SOS polynomials is a subset of the set of positive polynomials and checking if a polynomial is a SOS can be performed by solving a semi-definite program (SDP), thus efficiently solved with software packages implementing interior point methods. The details will be presented in next chapter.

The steps of the modified iteration procedure of Section (3-2) are then formulated as follows,

Step 1: SOS policy evaluation

We provide a sufficient condition for calculating a (piecewise) certificate $W^c(x, q^c)$ for global stability of the system (3-1) with a piecewise policy u_{pw}^c defined in a partition Ξ

$$u_{pw}^c = \begin{cases} h^c(x) & x \in \cup_{i\in\{1,2,3\}} \Xi_i^{(c-1)} \\ h^{(c-1)}(x) & x \in \Xi_4^{(c-1)}, \end{cases} \tag{3-32}$$

then, find $W^c(x, q^c)$, $\Pi_i^{pw}(\xi)$ satisfying

$$\begin{aligned} p_1(\xi) = -\lambda W^c - \frac{\partial W^c}{\partial x}\left(Ax + B\left(u^{pw} - q^{pw}\right)\right) - \frac{\partial W^c}{\partial q^c}\phi^{c/sw} \\ - x'Q(x)x - (u^{pw} - q^{pw})'R(x)(u^{pw} - q^{pw}) \\ + q^{sw'}\Pi_1^{pw}(\xi)(u^{pw} - q^{pw}) + \phi^{c/sw'}\Pi_2^{pw}(\xi)(\dot{u}^{c/sw} - \phi^{c/sw}) \\ + q^{c'}\Pi_3^{pw}(\xi)(\dot{u}^{c/sw} - \phi^{c/sw}) > 0 \quad \forall x \in \mathbb{R}^n \end{aligned} \tag{3-33}$$

$$\Pi_1^{pw}(\xi) \geq 0 \tag{3-34}$$

with

$$\xi = \begin{bmatrix} x' & u^{pw\prime} & q^{pw\prime} & \phi^{c/sw\prime} \end{bmatrix}' \tag{3-35}$$

where $\phi^{c/sw}$ denotes the time-derivative of q^c along the trajectories corresponding to policy u_{pw}^c and $\Pi_1^{pw}(\xi)$. The expression for $\dot{u}^{c/sw}$

$$\dot{u}^{c/sw} = \frac{d\, h^c(x)}{dx}\left(Ax + B\left(u^{pw} - q^{pw}\right)\right). \tag{3-36}$$

Notice that, since (3-32) is defined according to the partition, the variable u^{pw} in (3-33) satisfies

$$(u^{pw} - h^c) = 0 \quad x \in \cup_{i\in\{1,2,3\}}\Xi_i^{(c-1)} \tag{3-37}$$

$$(u^{pw} - h^{(c-1)}) = 0 \quad x \in \Xi_4^{(c-1)}. \tag{3-38}$$

Therefore, considering the representation of sets

$$\begin{array}{l} \cup_{i\in\{1,2,3\}}\Xi_i^{(c-1)} = \{x \in \mathbb{R}^n | p_{0j}(x, \underline{u}, \overline{u}) \geq 0, j = 1, \ldots, n_0\} \\ \Xi_4^{(c-1)} = \{x \in \mathbb{R}^n | p_{4j}(x, \underline{u}, \overline{u}) \geq 0, j = 1, \ldots, n_4\} \end{array} \tag{3-39}$$

and assuming polynomial dependence of multipliers Π_i on variable x, we obtain the following sufficient sum-of-squares condition for (3-33) to hold with control policy (3-32). According to S-Procedure,

$$p_1(\xi) + \Pi_{41}(xi)(u^{pw} - h^c) + \sum_{j=1}^{n_0} m_0(\xi)p_{0j}(x, \underline{u}, \overline{u}) > 0 \tag{3-40}$$

$$p_1(\xi) + \Pi_{42}(xi)(u^{pw} - h^{(c-1)}) + \sum_{j=1}^{n_4} m_4(\xi)p_{4j}(x, \underline{u}, \overline{u}) > 0 \tag{3-41}$$

$$\Pi_1 > 0 \tag{3-42}$$

with $m_0(\xi) > 0, m_4(\xi) > 0$. Since $W^c(x, q^c)$ is an upper bound of the value function (given that the current policy may not be the optimal one), we solve

$$\begin{array}{ll} \text{minimize} & W^c(x(0), dz(u^c(x(0)))) \quad (3-43) \\ \text{subject to} & (3-40) - (3-42) \quad (3-44) \end{array}$$

that minimizes the cost function with initial state $x(0)$.

Step 2: Feasibility of W^c

In this step, we check for the feasibility of W^c as a certificate of global stability for u^c to provide a stable control policy in the entire state space by solving

$$p_1(\xi) > 0, \tag{3-45}$$

$$\Pi_1 > 0. \tag{3-46}$$

in which we substitute u^c by $h^c(x)$.

Provided the above SOS constraints (3-45) (3-46) are satisfied, the control policy is updated as $\overline{u}^c(x) = h^c(x)$; otherwise $\overline{u}^c(x) = h^{(c-1)}(x)$. The policy improvement step consists on defining the new policy completing the policy update step.

$$u_{pw}^{c+1} = \begin{cases} h^{c+1} \to -\frac{1}{2}R^{-1}B' \left.\frac{\partial W^c}{\partial x}'\right|_{q^c=0} & x \in \cup_{i\in\{1,2,3\}}\Xi_i^{(c-1)} \\ h^c \to \overline{u}^c(x) & x \in \Xi_4^c \end{cases} \tag{3-47}$$

3-4 Partial Conclusions

This chapter applies a policy iteration procedure for the synthesis of optimal and globally stabilizing control policies for Linear Time Invariant (LTI) Asymptotically Null-controllable with Bounded Inputs (ANCBI) systems. The first step of the policy iteration relies on finding a class of piecewise quadratic value function (Lyapunov functions) which is non-differentiable, but continuous, and polynomial in both the state and the deadzone functions of the input signals associated to a control policy. The second step of the policy iteration is based on a piecewise control policy improvement. An important aspect of the applied piecewise policy is that at each step of the iteration, the computed policy is globally stabilizing and the existence of an improving value function is guaranteed as well. The solution to the inequalities which is required to hold at each step of the policy iteration, is obtained by solving Sum-of-Squares Programs (SOSP) that can be efficiently implemented with semidefinite programming (SDP) solvers.

This methodology has to be extended to nonlinear systems in order to the applicable to the pendulum swing-up problem. And in the process of piecewise polynomial policy iterations for optimal control laws in input-saturated systems, the extension of the applied methodology to nonlinear system of swing-up inverted pendulum is achievable. Such an extension will account for generalized sector condition which is instrumental to compute region of attraction estimates. We will also generalize the obtained conditions to systems defined by polynomial vector fields and polynomial input matrices.

Chapter 4

Handling Nonlinear Systems with Sum of Squares Decomposition

The analysis of nonlinear systems has always been a difficult task as the only direct, efficient methodology requires the construction of what is called a Lyapunov function. The difficulty lies not only in the "manual" construction of Lyapunov functions but also in the complexity of testing the non-negativity of the two Lyapunov conditions. Indeed, even if someone was to propose a high order Lyapunov function, it might not at all be possible to verify the two conditions that it needs to satisfy: that it is positive definite in some region around the zero equilibrium and that its derivative along the system's trajectories is non-positive.

Recent advances in the areas of semidefinite programming along with use of the sum of squares decomposition to efficiently check nonnegative have allowed an algorithmic procedure for systems analysis. But this methodology is restricted to systems described by polynomial vector fields whereas physical systems, the functionality of which is in the focus of many research areas, seldom have polynomial vector fields. In this case, the stability analysis of the closed loop system using the above methodology becomes difficult, as the same variable appears both in polynomial and non-polynomial terms. However, it has been shown in [39] that any system with non-polynomial nonlinearities can be transformed through a simple series of steps to equivalent polynomial systems under equality and inequality constraints on the state variables.

The aim of this chapter is to extend Lyapunov's stability theorem to handle recasted systems and then use the sum of squares decomposition to construct Lyapunov functions in the new coordinates. When mapped back to the original variables, these Lyapunov functions will contain the original non-polynomial terms. And this chapter is organized as follows, in Section 4.1, we review briefly the Lyapunov stability theory for nonlinear systems and how the sum of squares (SOS) decomposition can be used to construct Lyapunov functions. In Section 4.2 we present the recasting algorithm and extend the standard Lyapunov theorem to handle the recasted systems. In Section 4.3, we present the analysis of system which have non-polynomial vector fields.

4-1 Lyapunov Stability for Nonlinear Systems

Here we concentrate on the nonlinear systems of the form

$$\dot{z} = f(z) \tag{4-1}$$

where $z \in \mathbb{R}^n$ and for which we assume without loss of generality that $f(0) = 0$, i.e. the origin is an equilibrium of the system. One of the most important properties related to this equilibrium is its stability, and assessing whether stability of the equilibrium holds has been in the center of systems and control research for more than a century. It was not until just before the turn of the 19th century that A. M. Lyapunov formulated sufficient conditions for stability [40] that do not require knowledge of the solution, but are based on the construction of an "energy-like" function, well known nowadays as a "Lyapunov function".

More precisely, the conditions are stated in the following theorem.

Theorem 1 (Lyapunov). For an open set $\mathcal{D} \subset \mathbb{R}^n$ with $0 \in \mathcal{D}$, suppose there exists a continuously differentiable function $V : \mathcal{D} \to \mathbb{R}$ such that

$$V(0) = 0, \tag{4-2}$$
$$V(z) > 0 \qquad \forall z \in \mathcal{D} \setminus \{0\}, \tag{4-3}$$
$$\frac{\partial V(z)}{\partial z} f(z) \leq 0 \quad \forall z \in \mathcal{D}. \tag{4-4}$$

Then $z = 0$ is a stable equilibrium of (4-1).

It is unfortunate that even with such a powerful theorem, the problem of proving stability of equilibrium of nonlinear systems is still difficult; the reason is that there has been no coherent methodology for constructing the Lyapunov function $V(z)$.

In order to simplify the problem at hand, let us assume that $f(z)$ is a polynomial vector field, and that we will be searching for $V(z)$ that is also a polynomial in z. Then the two conditions in Theorem 1 become polynomial nonnegative conditions. To get rid of the difficult task of testing them, we can restrict our attention to cases in which the two conditions admit SOS decompositions. For $\mathcal{D} = \mathbb{R}^n$, the conditions in Theorem 1 can then be formulated as SOS program stated in the following proposition, and a Lyapunov function that satisfies these conditions can be constructed using semidefinite programming.

Proposition 2. Suppose that for the system (4-1) there exists a polynomial function $V(z)$ such that

$$V(0) = 0, \tag{4-5}$$
$$V(z) - \phi(z) \quad \text{is SOS}, \tag{4-6}$$
$$-\frac{\partial V(z)}{\partial z} f(z) \quad \text{is SOS}. \tag{4-7}$$

where $\phi(z) > 0$ for $z \neq 0$. Then the zero equilibrium of (4-1) is stable.

Proof. Condition (4-6) enforces $V(z)$ to be positive definite. Since condition (4-7) implies that $\dot{V}(z)$ is negative semidefinite, it follows that $V(z)$ is a Lyapunov function that proves stability of the origin.

In the above proposition, the function $\phi(z)$ is used to enforce positive definiteness of $V(z)$. If $V(z)$ is a polynomial of degree $2d$, then $\phi(z)$ may be chosen as follows:

$$\phi(z) = \sum_{i=1}^{n} \sum_{j=1}^{d} \epsilon_{ij} z_i^{2j}$$

where ϵ satisfy

$$\sum_{j=1}^{m} \epsilon_{ij} > \gamma \qquad \forall i = 1, ..., n$$

with γ a positive number, and $\epsilon_{ij} \geq 0$ for all i and j. In fact, this choice of $\phi(z)$ will force $V(z)$ to be radially unbounded, and hence the stability property holds globally if the conditions in Proposition 2 are met.

4-2 Recasting and Analysis of Recasted Systems

In this section we present an algorithm that can be used to convert a non-polynomial system into a polynomial system. The algorithm is adapted from [39], and it is applicable to a very large class of non-polynomial systems, namely those whose vector field is composed of sums and products of elementary functions, or nested elementary functions of elementary functions. What are meant by elementary functions here are functions with explicit symbolic derivatives such as exponential (e^x), logarithm ($\ln x$), power (x^a), trigonometric ($sinx, cosx, etc.$), and hyperbolic functions ($sinhx, coshx, etc.$).

Suppose that the original system is given in the form

$$\dot{z}_i = \sum_j a_j \prod_k F_{ijk}(z) \tag{4-8}$$

where $i = 1, ..., n$; a_j's are real numbers; and $z = (z_1, ..., z_n)$. In the above equation, $F_{ijk}(z)$ are assumed to be elementary functions, or nested elementary functions of elementary functions. For the above system, the recasting algorithm is stated below.

Algorithm (adapted from [39], with some modifications)

1. Let $x_i = z_i$, for $i = 1, ..., n$.
2. For each $F_{ijk}(z)$ that is not of the form z_l^a, where a is some integer and $1 \leq l \leq n$, introduce a new variable x_m. Define $x_m = F_{ijk}(z)$.
3. Compute the differential equation describing the time evolution of x_m using the chain rule of differentiation.
4. Replace all appearances of such $F_{ijk}(z)$ in the system equations by x_m.
5. Repeat steps 2-4, until we obtain system equations with polynomial forms.

It is best to illustrate the application of the above algorithm by an example.

Example. Consider the differential equation

$$\dot{z} = sin(e^z - 1) + 4ln(z^2 + 1)$$

which we want to transform to a system with polynomial vector field. We start by defining $x_1 = z, x_2 = sin(e^z - 1)$, and $x_3 = ln(z^2 + 1)$. By the chain rule of differentiation and replacing the appearances of $z, sin(e^z - 1)$, and $ln(z^2 + 1)$ in the resulting equations by x_1, x_2, and x_3, we obtain

$$\begin{aligned}
\dot{x_1} &= x_2 + 4x_3, \\
\dot{x_2} &= cos(e^z - 1)\, e^z\, \dot{z} \\
&= cos(e^{x_1} - 1)\, e^{x_1}(x_2 + 4x_3), \\
\dot{x_3} &= \frac{2}{z^2 + 1} z\dot{z} \\
&= \frac{x_1(x_2 + 4x_3)}{x_1^2 + 1}
\end{aligned}$$

Notice that the equations for $\dot{x_1}$ and $\dot{x_3}$ are in polynomial forms. However, the equation for $\dot{x_2}$ is not in a polynomial form and thus we continue by defining $x_4 = cos(e^{x_1} - 1)$ and $x_5 = e^{x_1}$. Using the chain rule of differentiation again, we obtain

$$\begin{aligned}
\dot{x_2} &= x_4 x_5 (x_2 + 4x_3), \\
\dot{x_4} &= -sin(e^{x_1} - 1)e^{x_1}(x_2 + 4x_3) \\
&= -x_2 x_5 (x_2 + 4x_3), \\
\dot{x_5} &= e^{x_1}(x_2 + 4x_3) \\
&= x_5(x_2 + 4x_3)
\end{aligned}$$

At this point, we terminate the recasting process, since the differential equations describing the evolutions of $x_1, ..., x_5$ are already in rational forms. More examples can be found in the Appendix.

The recasting process described in the previous subsection generally produces a recasted system whose dimension is higher than the dimension of the original system. To describe the original system faithfully, constraints of the form $x_{n+1} = F(x_1, ..., x_n)$ that are created when new variables are introduced should be taken into account. These constraints define an n-dimensional manifold on which the solutions to the original differential equations lie. In general such constraints cannot be converted into polynomial forms, even though sometimes there exist polynomial constraints that are induced by the recasting process. For example:

- Two variables introduced for trigonometric functions such as $x_2 = sinx_1, x_3 = cosx_1$ are constrained via $x_2^2 + x_3^2 = 1$.
- Introducing a variable to replace a power function such as $x_2 = \sqrt{x_1}$ introduces the constraints $x_2^2 - x_1 = 0, x_2 \geq 0$.

- Introducing a variable to replace an exponential function such as $x_2 = e^{x_1}$ induces the constraint $x_2 \geq 0$.

We will shortly discuss how both types of constraints described above can be taken into account in the stability analysis using the sum of squares decomposition technique. For our purpose, suppose that for a non-polynomial system

$$\dot{z} = f(z) \tag{4-9}$$

which has an equilibrium at the origin, the recasted system obtained using the procedure of the previous subsection is written as

$$\dot{\tilde{x}}_1 = f_1(\tilde{x}_1, \tilde{x}_2), \tag{4-10}$$
$$\dot{\tilde{x}}_2 = f_2(\tilde{x}_1, \tilde{x}_2), \tag{4-11}$$

where $\tilde{x}_1 = (x_1, ..., x_n = z)$ are the state variables of the original system, $x_2 = (x_{n+1}, ..., x_{n+m})$ are the new variables introduced in the recasting process, and $f_1(\tilde{x}_1, \tilde{x}_2), f_2(\tilde{x}_1, \tilde{x}_2)$ have polynomial forms.

We denote the constraints that arise directly from the recasting process by

$$\tilde{x}_2 = F(\tilde{x}_1) \tag{4-12}$$

and those that arise indirectly by

$$G_1(\tilde{x}_1, \tilde{x}_2) = 0, \tag{4-13}$$
$$G_2(\tilde{x}_1, \tilde{x}_2) \geq 0 \tag{4-14}$$

where F, G_1 and G_2 are column vectors of functions with appropriate dimensions, and the equalities or inequalities hold entry-wise. Finally, denote the collective denominator of $f_1(\tilde{x}_1, \tilde{x}_2)$, and $f_2(\tilde{x}_1, \tilde{x}_2)$ by $g(\tilde{x}_1, \tilde{x}_2)$. That is, $g(\tilde{x}_1, \tilde{x}_2)$ should be a polynomial function such that $g(\tilde{x}_1, \tilde{x}_2)f_1(\tilde{x}_1, \tilde{x}_2)$ and $g(\tilde{x}_1, \tilde{x}_2)f_2(\tilde{x}_1, \tilde{x}_2)$ are polynomials. We also assume that $g(\tilde{x}_1, \tilde{x}_2) > 0 \quad \forall (\tilde{x}_1, \tilde{x}_2) \in \mathcal{D}_1 \times \mathcal{D}_2$, since otherwise the system is not well-posed.

Proving stability of the zero equilibrium of the original system (4-9) amounts to proving that all trajectories starting close enough to $z = 0$ will remain close to this equilibrium point. This can be accomplished by finding a Lyapunov function $V(z)$ that satisfies the following conditions of Lyapunov's stability theorem, Theorem 1. In terms of the new variables $\tilde{x}_1$ and $\tilde{x}_2$, sufficient conditions that guarantee the existence of a Lyapunov function for the original system are stated in the following proposition.

Proposition 3. (Lyapunov Conditions) Let $\mathcal{D}_1 \subset \mathbb{R}^n$ and $\mathcal{D}_2 \subset \mathbb{R}^m$ be open sets such that $0 \in \mathcal{D}_1$ and $F(\mathcal{D}_1) \subseteq \mathcal{D}_2$. Furthermore, define $\tilde{x}_{2,0} = F(0)$. If there exists a function $\tilde{V} : \mathcal{D}_1 \times \mathcal{D}_2 \to \mathbb{R}$ and column vectors of functions $\lambda_1(\tilde{x}_1, \tilde{x}_2), \lambda_2(\tilde{x}_1, \tilde{x}_2), \sigma_1(\tilde{x}_1, \tilde{x}_2)$ and $\sigma_2(\tilde{x}_1, \tilde{x}_2)$ with appropriate dimensions such that

$$\tilde{V}(0, \tilde{x}_{2,0}) = 0, \tag{4-15}$$

$$\tilde{V}(\tilde{x}_1, \tilde{x}_2) - \lambda_1^T G_1(\tilde{x}_1, \tilde{x}_2) - \sigma_1^T(\tilde{x}_1, \tilde{x}_2) G_2(\tilde{x}_1, \tilde{x}_2) \geq \phi(\tilde{x}_1, \tilde{x}_2) \quad \forall (\tilde{x}_1, \tilde{x}_2) \in \mathcal{D}_1 \times \mathcal{D}_2, \tag{4-16}$$

$$- g(\tilde{x}_1, \tilde{x}_2) \left(\frac{\partial \tilde{V}}{\partial \tilde{x}_1} f_1(\tilde{x}_1, \tilde{x}_2) + \frac{\partial \tilde{V}}{\partial \tilde{x}_2} f_2(\tilde{x}_1, \tilde{x}_2) \right) - \lambda_2^T(\tilde{x}_1, \tilde{x}_2) G_1(\tilde{x}_1, \tilde{x}_2) - \sigma_2^T G_2(\tilde{x}_1, \tilde{x}_2) \geq 0 \quad \forall (\tilde{x}_1, \tilde{x}_2) \in \mathcal{D}_1 \times \mathcal{D}_2, \quad (4\text{-}17)$$

$$\sigma_1(\tilde{x}_1, \tilde{x}_2) \geq 0 \quad \forall (\tilde{x}_1, \tilde{x}_2) \in \mathbb{R}^{n+m}, \quad (4\text{-}18)$$

$$\sigma_2(\tilde{x}_1, \tilde{x}_2) \geq 0 \quad \forall (\tilde{x}_1, \tilde{x}_2) \in \mathbb{R}^{n+m}, \quad (4\text{-}19)$$

for some scalar function $\phi(\tilde{x}_1, \tilde{x}_2)$ with $\phi(\tilde{x}_1, F(\tilde{x}_1)) > 0 \quad \forall \tilde{x}_1 \in \mathcal{D}_1 \setminus \{0\}$, then $z = 0$ is a stable equilibrium of (4-9).

The above non-negativity conditions can be relaxed to appropriate sum of squares conditions so that they can be algorithmically verified using semidefinite programming. This will also lead the way to an algorithmic construction of the Lyapunov function V. Here we assume that $\mathcal{D}_1 \times \mathcal{D}_2$ is a semialgebraic set described by the following inequalities:

$$\mathcal{D}_1 \times \mathcal{D}_2 = \{(\tilde{x}_1, \tilde{x}_2) \in \mathbb{R}^n \times \mathbb{R}^m : G_D(\tilde{x}_1, \tilde{x}_2) \geq 0\},$$

where $G_D(\tilde{x}_1, \tilde{x}_2)$ is a column vector of polynomials and the inequality is satisfied entry-wise. With all this notation, the sum of squares conditions can be stated as follows.

Proposition 4. (Sum-of-Square Relaxation) Let the system (4-10)-(4-11) and the functions $F(\tilde{x}_2), G_1(\tilde{x}_1, \tilde{x}_2), G_2(\tilde{x}_1, \tilde{x}_2), G_D(\tilde{x}_1, \tilde{x}_2)$ and $g(\tilde{x}_1, \tilde{x}_2)$ be given. Define $\tilde{x}_{2,0} = F(0)$. If there exists a polynomial function $\tilde{V}(\tilde{x}_1, \tilde{x}_2)$, column vectors of polynomial functions $\lambda_1(\tilde{x}_1, \tilde{x}_2), \lambda_2(\tilde{x}_1, \tilde{x}_2)$, and column vectors of sum of squares polynomials $\sigma_1(\tilde{x}_1, \tilde{x}_2), \sigma_2(\tilde{x}_1, \tilde{x}_2), \sigma_3(\tilde{x}_1, \tilde{x}_2), \sigma_4(\tilde{x}_1, \tilde{x}_2)$ with appropriate dimensions such that

$$\tilde{V}(0, \tilde{x}_{2,0}) = 0, \quad (4\text{-}20)$$

$$\tilde{V}(\tilde{x}_1, \tilde{x}_2) - \lambda_1^T G_1(\tilde{x}_1, \tilde{x}_2) - \sigma_1^T(\tilde{x}_1, \tilde{x}_2) G_2(\tilde{x}_1, \tilde{x}_2) - \sigma_3^T(\tilde{x}_1, \tilde{x}_2) G_D(\tilde{x}_1, \tilde{x}_2) - \phi(\tilde{x}_1, \tilde{x}_2) \text{ is a sum of squares,} \quad (4\text{-}21)$$

$$-g(\tilde{x}_1, \tilde{x}_2) \left(\frac{\partial \tilde{V}}{\partial \tilde{x}_1} f_1(\tilde{x}_1, \tilde{x}_2) + \frac{\partial \tilde{V}}{\partial \tilde{x}_2} f_2(\tilde{x}_1, \tilde{x}_2) \right) - \lambda_2^T(\tilde{x}_1, \tilde{x}_2) G_1(\tilde{x}_1, \tilde{x}_2) - \sigma_2^T G_2(\tilde{x}_1, \tilde{x}_2) - \sigma_4^T G_D(\tilde{x}_1, \tilde{x}_2) \text{ is a sum of squares.} \quad (4\text{-}22)$$

for some scalar polynomial function $\phi(\tilde{x}_1, \tilde{x}_2)$ with $\phi(\tilde{x}_1, F(\tilde{x}_1)) > 0 \quad \forall \tilde{x}_1 \in \mathcal{D}_1 \setminus \{0\}$, then $z = 0$ is a stable equilibrium of (4-9).

4-3 Example with Trigonometric Function

Here we present one example relevant to the swing up problem of a pendulum. Other examples can be found in the Appendix.

Example: Whirling Pendulum

Consider the whirling pendulum [41] shown in Figure 4-1. It is a pendulum of length l_p whose suspension end is attached to a rigid arm of length l_a, with a mass m_b attached to its free end. The arm rotates with angular velocity $\dot{\theta}_a$. The pendulum can oscillate with angular velocity $\dot{\theta}_b$ in a plane normal to the arm, making an angle θ_p with the vertical in the instantaneous plane of motion. We will ignore frictional effects and assume that all links are slender so that their moment of inertia can be neglected.

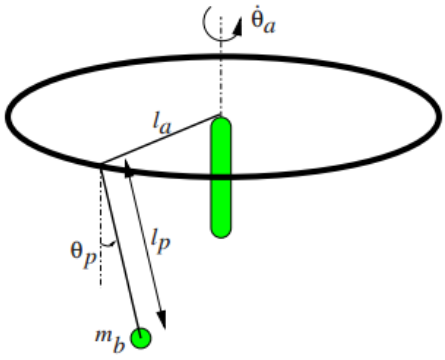


Figure 4-1: Whirling pendulum

Using $x_1 = \theta_p$, and $x_2 = \dot{\theta}_p$ as state variables, we obtain the following state equations for the system:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \dot{\theta}_a^2 \sin x_1 \cos x_1 - \frac{g}{l_p} \sin x_1 \end{cases} \tag{4-23}$$

The number and stability properties of equilibrium in this system depend on the value of $\dot{\theta}_a$. When the condition

$$\dot{\theta}_a^2 < \frac{g}{l_p} \tag{4-24}$$

is satisfied, the only equilibrium in the system are (x_1, x_2) satisfying $sinx_1 = 0, x_2 = 0$. One equilibrium corresponds to $x_1 = 0$, i.e., the pendulum is hanging vertically downward (stable), and the other equilibrium corresponds to $x_1 = \pi$, the vertically upward position (unstable). As $\dot{\theta}_a$ is increased beyond $\frac{g}{l_p}$, a supercritical pitchfork bifurcation of equilibrium occurs [42]. The $(x_1, x_2) = (0, 0)$ equilibrium becomes unstable, and two other equilibriums appear. These equilibriums correspond to $cosx_1 = \frac{g}{l_p \dot{\theta}_a^2}, x_2 = 0$.

We will now prove the stability of the equilibrium point at the origin for $\dot{\theta}_a$ satisfying (4-24) by constructing a Lyapunov function. Obviously the energy of this mechanical system can be used as a Lyapunov function, but since our purpose is to show that a Lyapunov function can be found using the SOS decomposition, we will assume that our knowledge is limited to

the state equations describing the system and that we know nothing about the underlying energy.

Since the vector field (4-23) is not polynomial, a transformation to a polynomial vector field must be performed before we are able to construct a Lyapunov function using the SOS decomposition. For this purpose, introduce $u_1 = sinx_1$ and $u_2 = cosx_1$ to get recasted system:

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = \dot{\theta}_a^2 u_1 u_2 - \frac{g}{l_p} u_1 \\ \dot{u_1} = x_2 u_2 \\ \dot{u_2} = -x_2 u_1 \end{cases} \tag{4-25}$$

In addition, we have the algebraic constraint

$$u_1^2 + u_2^2 - 1 = 0 \tag{4-26}$$

So the whirling pendulum system will now be described by (4-25) (4-26). Notice that all the functions here are polynomial, so that Proposition 4 can be used to prove stability.

Then we will perform the analysis with the parameters of the system set at some fixed values. Assume that $\dot{\theta}_a = 1, l_p = 1, g = 10$, for which condition (4-24) is satisfied. For a mechanical system like this, we expect that some trigonometric terms will be needed in the Lyapunov function. Thus, we will try to find a Lyapunov function of the following form:

$$\begin{aligned} V &= a_1 x_2^2 + a_2 u_1^2 + a_3 u_2^2 + a_4 u_2 + a_5 \\ &= a_1 x_2^2 + a_2 \sin^2 x_1 + a_3 \cos^2 x_1 + a_4 \cos x_1 + a_5 \end{aligned}$$

where the a_i are the unknown coefficients. And these coefficient must satisfy

$$a_3 + a_4 + a_5 = 0$$

for V to be equal to zero at $(x_1, x_2) = (0, 0)$. To guarantee that V is positive definite, by using SOSTOOLS, we search for V that satisfy

$$(V - W) \text{ is a sum of squares}$$

where $W = \epsilon_1(1 - u_2) + \epsilon_2 x_2^2$ with ϵ_1 and ϵ_2 are positive constants.

In other words, we search for V satisfy

$$V - W = V - \epsilon_1(1 - u_2) - \epsilon_2 x_2^2 \geq 0$$

Then V positive definiteness hold as

$$\epsilon_1(1 - u_2) + \epsilon_2 x_2^2 = \epsilon_1(1 - \cos x_1) + \epsilon_2 x_2^2 \geq 0$$

is a positive definite function in the space (x_1, x_2).

An example of Lyapunov function for this whirling pendulum system is given by,

$$V = 0.33445 x_2^2 + 1.4615 u_1^2 + 1,7959 u_2^2 - 6.689 u_2 + 4.8931$$

4-4 Partial Conclusions

In this chapter we have presented a methodology to analyze systems described by non-polynomial vector fields using the sum of squares decomposition and a recasting procedure. Using this recasting procedure, a non-polynomial system can be converted into a polynomial system with equality, inequality and integral constraints. In doing so, certain nonpolynomial nonlinearities can be handled, as shown in the examples and Lyapunov functions that are nonpolynomial in the state variables can be constructed.

Constructing Lyapunov functions has always been a challenging task and an important problem in dynamical systems and control theory. An algorithmic approach was developed recently to construct Lyapunov functions for dynamical systems with polynomial vector fields. This was based on a relaxation of the condition that a function is positive semidefinite to the condition that it is a sum of squares (SOS). Then an extension of the Lyapunov theorem in conjunction with the sum of squares decomposition and semidefinite programming can then be used to investigate the stability of the recasted system, the result of which can be used to infer the stability of the original system. Using the results of this chapter, the modified Policy Iteration Algorithm of Figure 3-2 can be straightforwardly extended to nonlinear systems with input saturation.

Chapter 5

Numerical Example

5-1 Test Case

In the following, we present a numerical example to illustrate the results obtained via the two optimal control methods. To evaluate and compare the performance of our algorithms, we apply them to the task of swinging up an inverted pendulum. The swing-up task was chosen because it is a low-dimensional, but challenging, highly nonlinear control problem. As the process has two states (the angle ϕ of the pendulum and the angular velocity $\dot{\phi}$) and one action (the torque u). It allows for easy visualization of the functions of interest (value function, control policy, phase plane). A photograph of this system is shown in Figure 5-1.

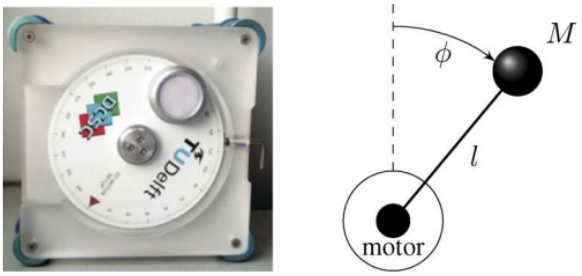


Figure 5-1: Inverted pendulum setup

The task is to swing up the pendulum from the pointing-down position to the upright position as quickly as possible and stabilize it in this position. The term "as quickly as possible" is quantified in term of a cost function to be minimized. The actuation signal u is limited with saturation $[-20, 20]$, $[-10, 10]$, $[-5, 5]$ respectively, making it impossible to directly move the pendulum to the upright position. For actor-critic algorithm, a continuous quadratic reward function ρ is used to define the swing-up task. This reward function has its maximum in the

upright position $[0\ 0]^T$ and quadratically penalizes nonzero values of ϕ, $\dot{\phi}$ and u.

$$r_k(x_{k-1}, u_{k-1}) = -x_{k-1}^T Q x_{k-1} - R u_{k-1}^2 \tag{5-1}$$

In continuous time, we define the cost function

$$J = \int_0^\infty L(x,u)dt = \int_0^\infty x'Qx + sat'(u(x))Rsat(u(x))dt \tag{5-2}$$

And in discrete time, we obtain the cumulative cost

$$J = \sum_{k=1}^{\infty} x_{k-1}^T Q x_{k-1} + R u_{k-1}^2 \tag{5-3}$$

with $Q = \begin{bmatrix} 5 & 0 \\ 0 & 0.1 \end{bmatrix}$, $R = 1$ in (5-1), (5-2), (5-3).

The motion equation of this system is

$$J\ddot{\phi} = Mgl\sin(\phi) - \left(b + \frac{K^2}{R}\right)\dot{\phi} + \frac{K}{R}u \tag{5-4}$$

where ϕ is the angle of the pendulum measured from the upright position. The (fully measurable) state x consists of the angle ϕ of the pendulum and the angular velocity $\dot{\phi}$ of the pendulum.

$$x = \begin{bmatrix} \phi \\ \dot{\phi} \end{bmatrix}$$

The model parameters are given in Table 5-1.

Table 5-1: Inverted Pendulum Model Parameters

Model Parameter	Symbol	Value	Units
Pendulum inertia	J	$1.91 \cdot 10^{-4}$	kgm^2
Pendulum mass	M	$5.50 \cdot 10^{-2}$	kg
Gravity	g	9.81	m/s^2
Pendulum length	l	$4.20 \cdot 10^{-2}$	m
Damping	b	$3 \cdot 10^{-6}$	Nms
Torque constant	K	$5.36 \cdot 10^{-2}$	Nm/A
Rotor resistance	R	9.50	Ω

The MBAC and Nolinear Policy Iteration algorithms were applied in simulation using the parameter settings in Table 5-2. Note that the discounted rate is chosen in such a way that $\gamma = e^{-\lambda T_s}$.

Table 5-2: The Parameter Settings for MBAC and Nonlinear Policy Iteration Methods

		MBAC	Nonlinear PI
sampling time(s)	T_s	0.02	-
reward discount rate (discrete)	γ	0.9980	-
discounted cost rate (continuous)	λ	-	0.1
control quantization	Δu	0.2	-
basis function type		triangular	-
number of basis function for x_1	n_1	30	-
number of basis function for x_2	n_2	30	-

Then the system is described as follows,

$$\dot{x}_1 = x_2 \tag{5-5}$$

$$\dot{x}_2 = \frac{Mgl}{J} sin(x_1) - \left(b + \frac{K^2}{R}\right) \frac{1}{J} x_2 + \frac{K}{JR} u \tag{5-6}$$

with $x_1 \in [-\pi, \pi], x_2 \in [-20, 20]$. However, we want to normalize the state $x_1 \in [-1, 1], x_2 \in [-1, 1]$. The advantage of the normalization is that all the monomials will be also between $[-1, 1]$ and the P matrix should be better conditioned. For this reason we define a new state which is

$$\bar{x}_1 = \frac{x_1}{\pi} \tag{5-7}$$

$$\bar{x}_2 = \frac{x_2}{2\pi^2} \tag{5-8}$$

Thus the system (5-5)-(5-6) becomes

$$\dot{\bar{x}}_1 = 2\pi \bar{x}_2 \tag{5-9}$$

$$\dot{\bar{x}}_2 = \frac{Mgl}{2J\pi^2} sin(\pi \bar{x}_1) - \left(b + \frac{K^2}{R}\right) \frac{1}{J} \bar{x}_2 + \frac{K}{2JR\pi^2} u \tag{5-10}$$

But this system contains the non-polynomial term of $\sin(\pi \bar{x}_1)$, so we transform it to the polynomial system by introducing $\bar{x}_3 = sin(\pi \bar{x}_1)$ that is between $[-1, 1]$, so there is no need to normalize it; and $\bar{x}_4 = \frac{cos(\pi \bar{x}_1) - 1}{2}$ that is in order to have equilibrium at 0. So the nonlinear system with polynomial term is

$$\dot{\bar{x}}_1 = 2\pi \bar{x}_2 \tag{5-11}$$

$$\dot{\bar{x}}_2 = \frac{Mgl}{2J\pi^2} \bar{x}_3 - \left(b + \frac{K^2}{R}\right) \frac{1}{J} \bar{x}_2 + \frac{K}{2JR\pi^2} u \tag{5-12}$$

$$\dot{\bar{x}}_3 = 2\pi^2 \bar{x}_2 (2\bar{x}_4 + 1) \tag{5-13}$$

$$\dot{\bar{x}}_4 = -\pi^2 \bar{x}_2 \bar{x}_3 \tag{5-14}$$

with equality constraint

$$\bar{x}_3^2 + (2\bar{x}_4 + 1)^2 = 1 \tag{5-15}$$

In addition, Q in (5-1), (5-2), (5-3) becomes

$$\bar{Q} = \begin{bmatrix} 5\pi^2 & 0 & 0 & 0 \\ 0 & 0.1(2\pi^2)^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and the initial state is

$$\bar{x}_0 = \begin{bmatrix} 0.75 \\ 0 \\ sin(0.75\pi) \\ cos(0.75\pi - 1)/2 \end{bmatrix}$$

In the following, we will show the figures of simulation results with input saturation $[-20, 20]$, saturation $[-10, 10]$ and saturation $[-5, 5]$ by using model based actor-critic algorithm and nonlinear policy iteration algorithm respectively. These figures will include

- Value function
- Control policy
- Phase plane with final controller
- Final trajectory
- Cost evolution

In addition, a table will show the cost improvement of nonlinear policy iteration algorithm compared to the model based actor-critic algorithm.

5-2 Simulation Results with Input-Saturation [-20,20]

5-2-1 Model based actor-critic algorithm

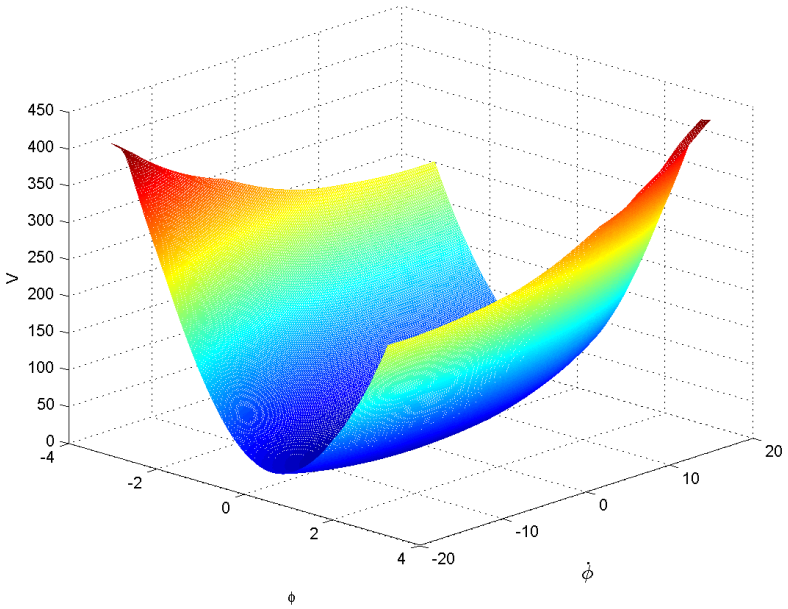


Figure 5-2: Value function with input-saturation [-20,20].

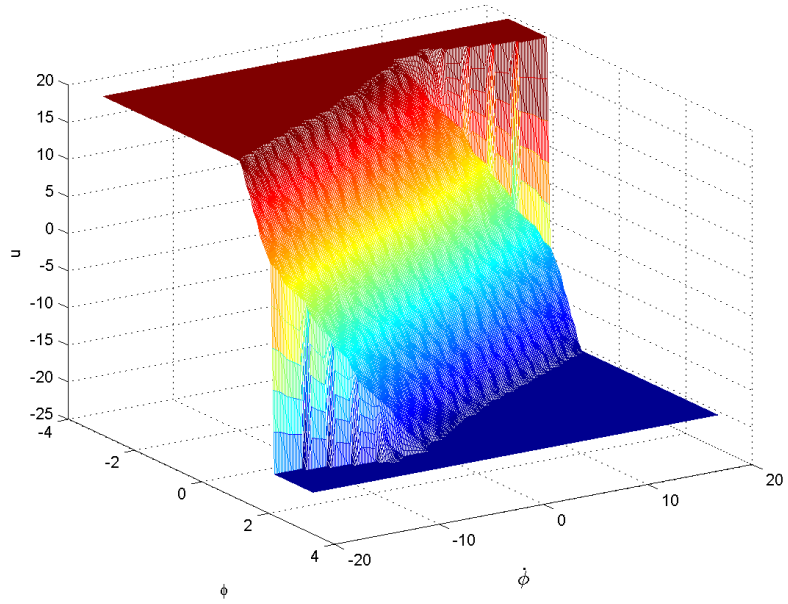


Figure 5-3: Control policy with input-saturation [-20,20].

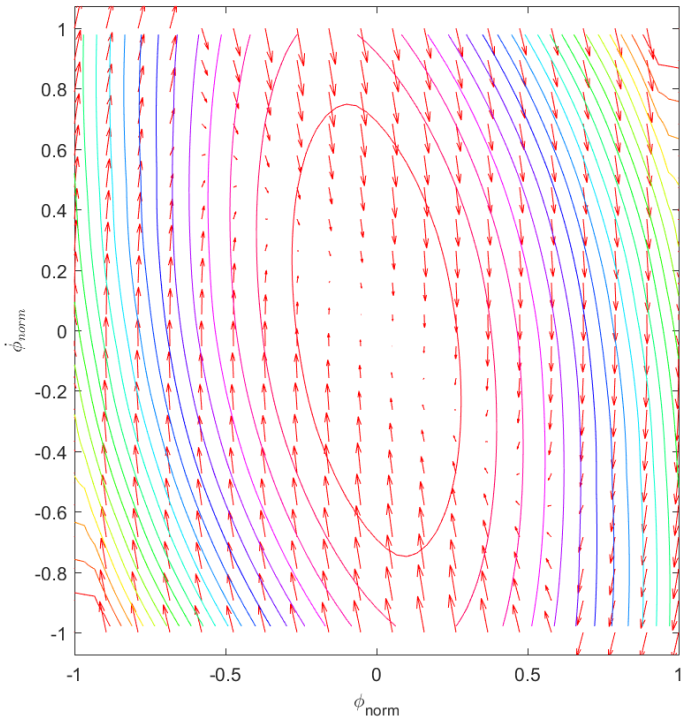


Figure 5-4: Phase phane with input-saturation [-20,20].

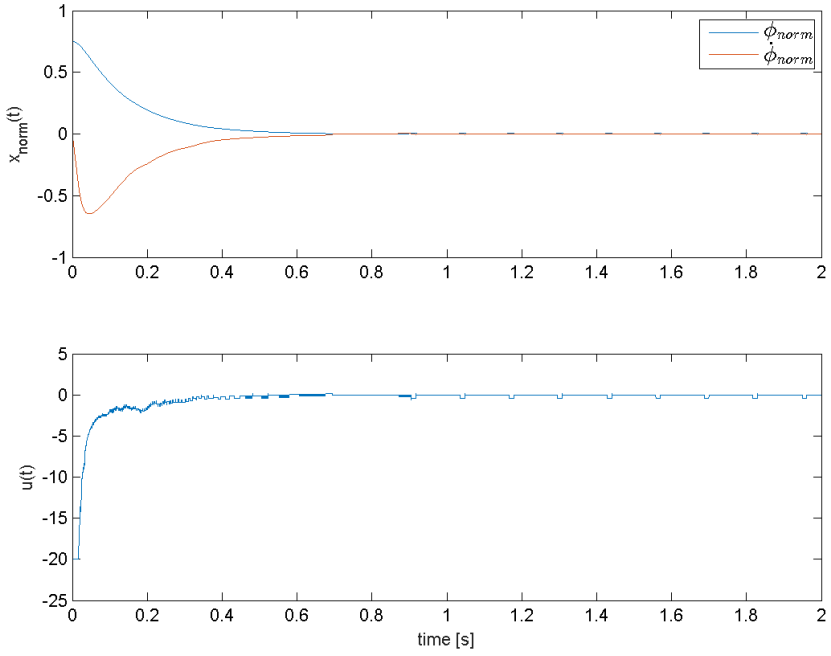


Figure 5-5: Final trajectory with input-saturation [-20,20].

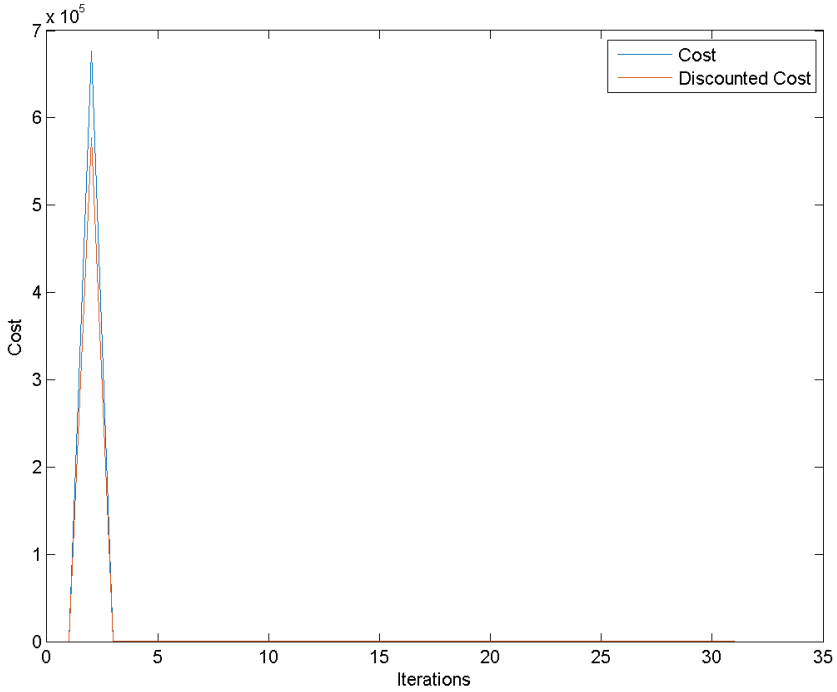


Figure 5-6: Cost with input-saturation [-20,20].

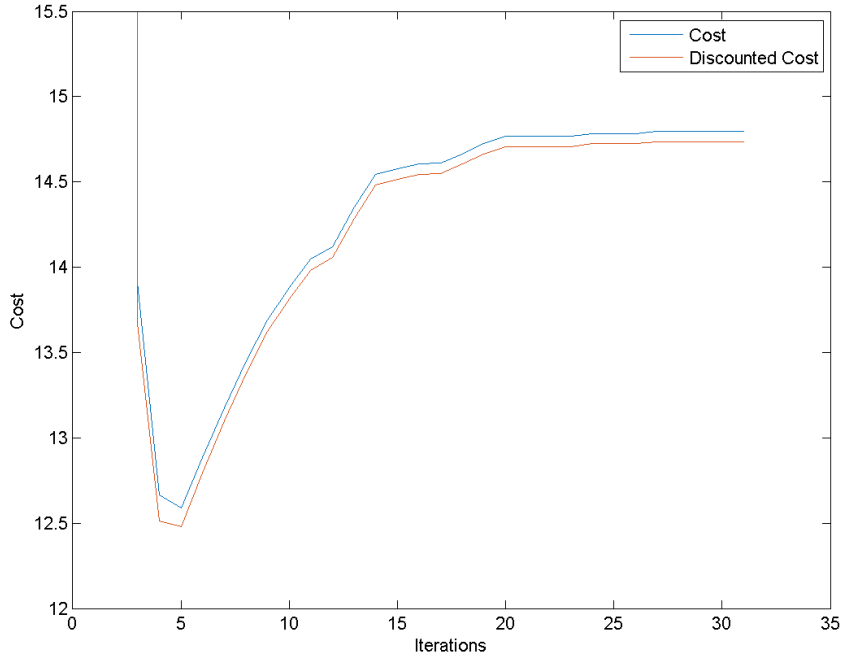


Figure 5-7: Cost (Zoom) with input-saturation [-20,20].

5-2-2 Nonlinear policy iteration algorithm

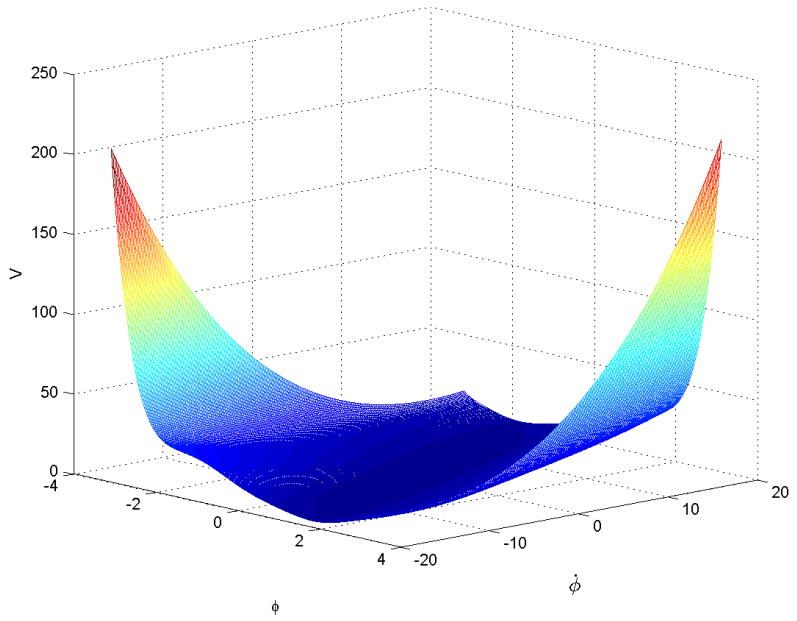


Figure 5-8: Value function with input-saturation [-20,20].

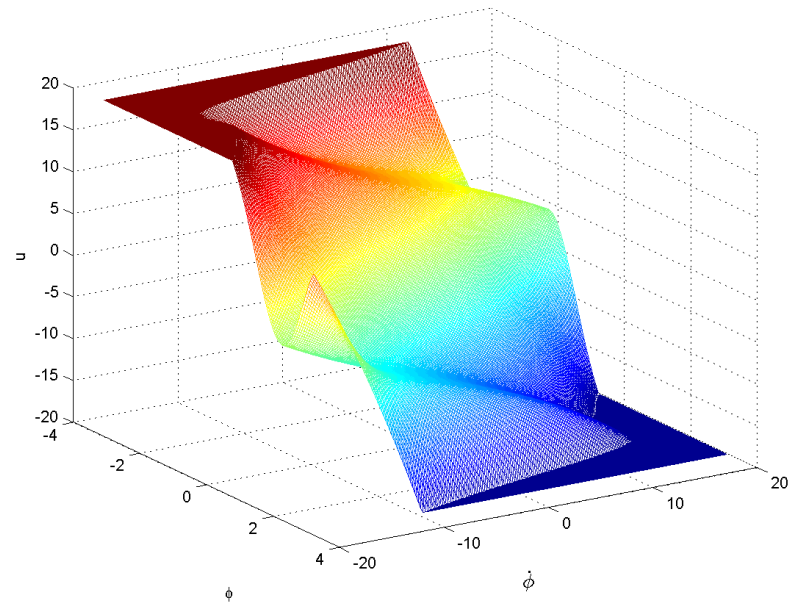


Figure 5-9: Control policy with input-saturation [-20,20].

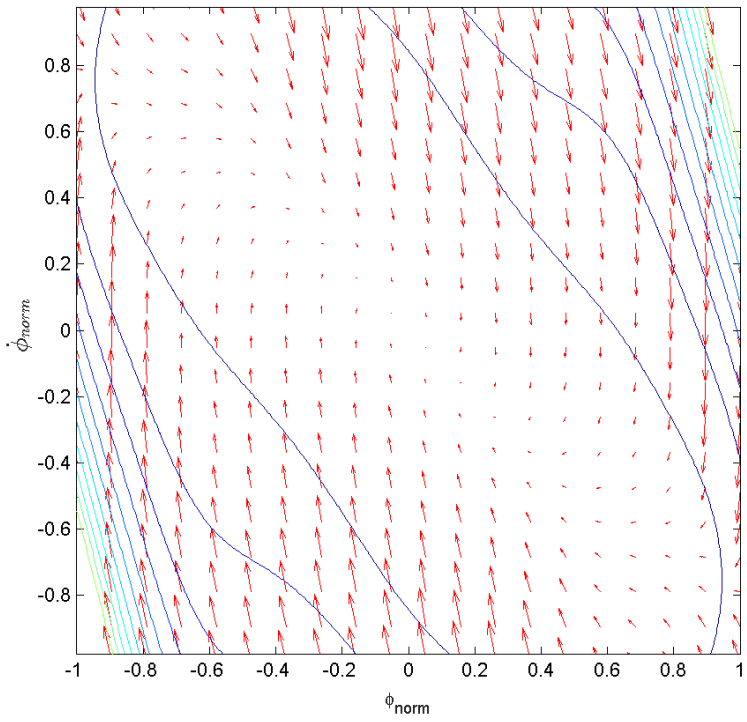


Figure 5-10: Phase phane with input-saturation [-20,20].

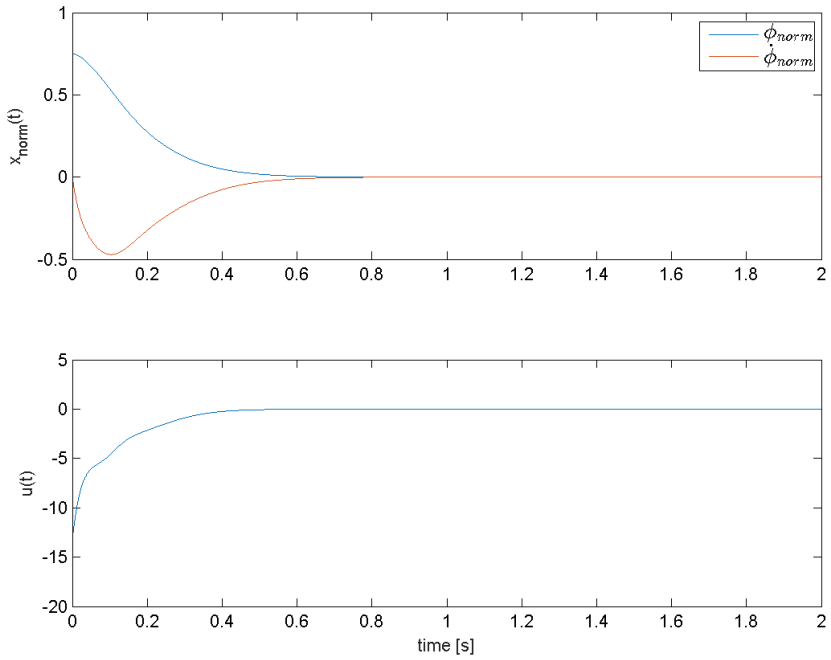


Figure 5-11: Final trajectory with input-saturation [-20,20].

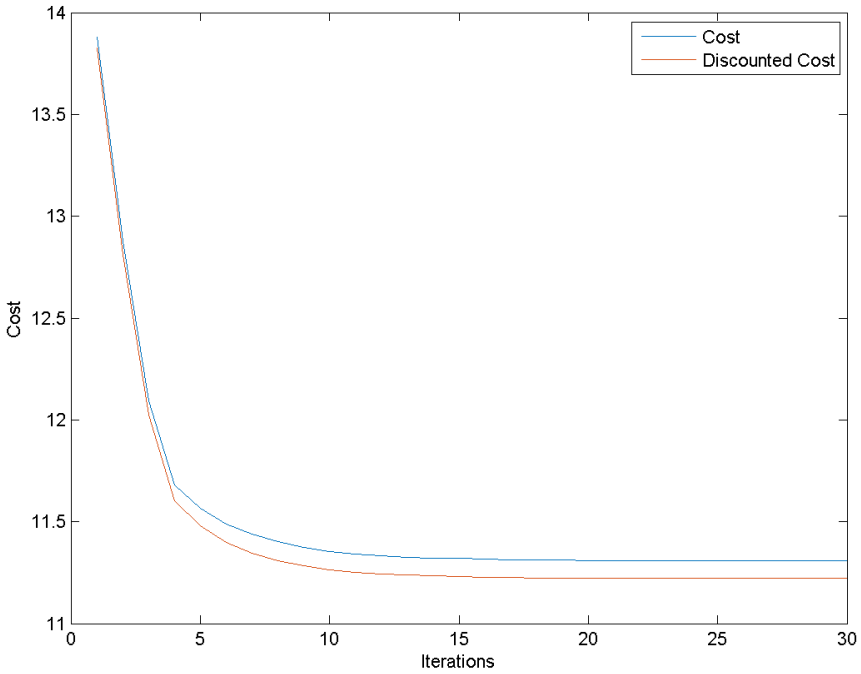


Figure 5-12: Cost with input-saturation [-20,20].

Table 5-3: Cost Comparison with Input-Saturation $[-20, 20]$

	Final discounted cost	Improvement
MBAC	14.75	
Nonlinear PI	11.25	23.7%

5-3 Simulation Results with Input-Saturation [-10,10]

5-3-1 Model based actor-critic algorithm

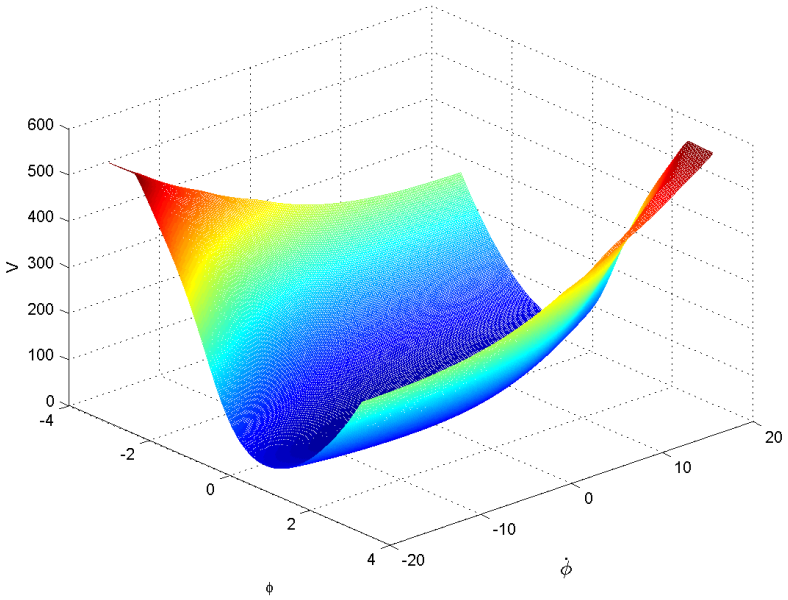


Figure 5-13: Value function with input-saturation [-10,10].

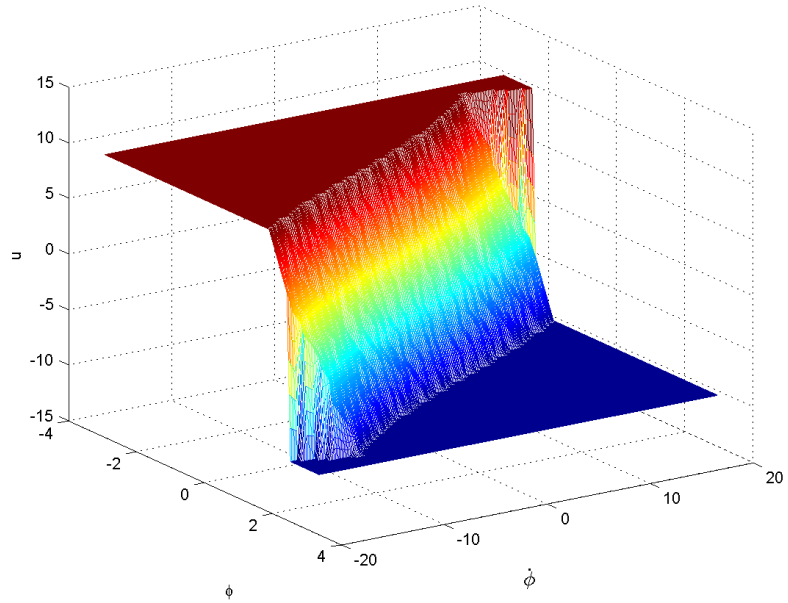


Figure 5-14: Control policy with input-saturation [-10,10].

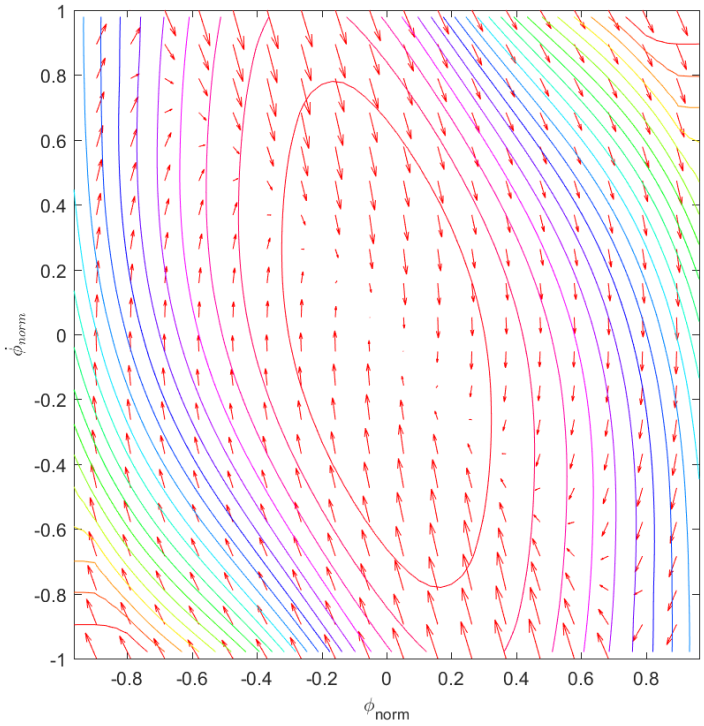


Figure 5-15: Phase phane with input-saturation [-10,10].

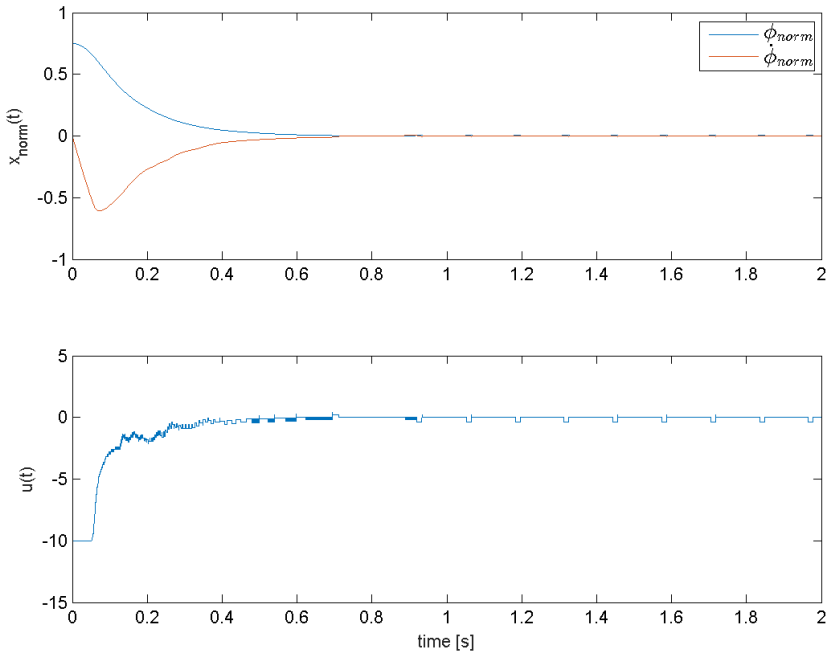


Figure 5-16: Final trajectory with input-saturation [-10,10].

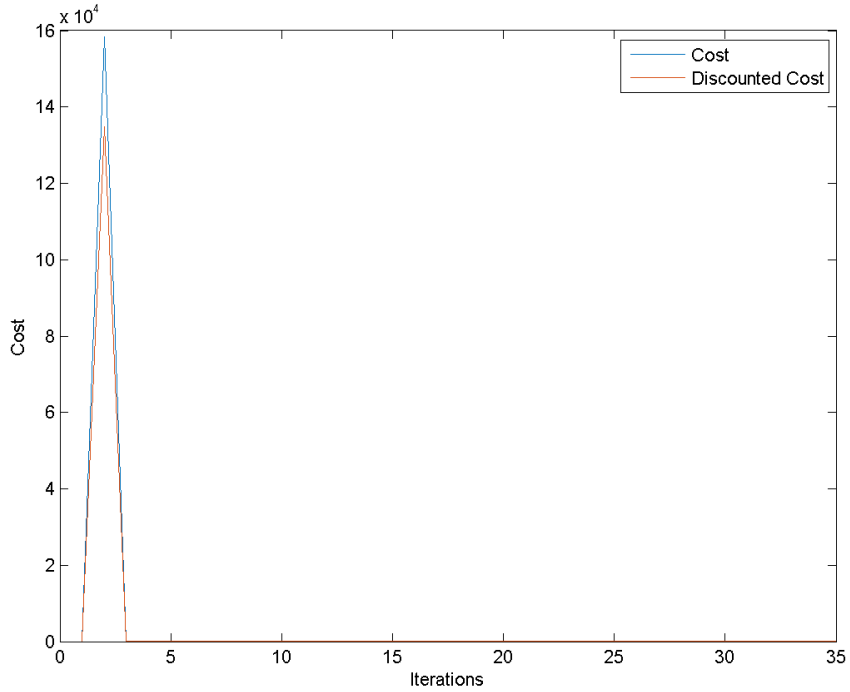


Figure 5-17: Cost with input-saturation [-10,10].

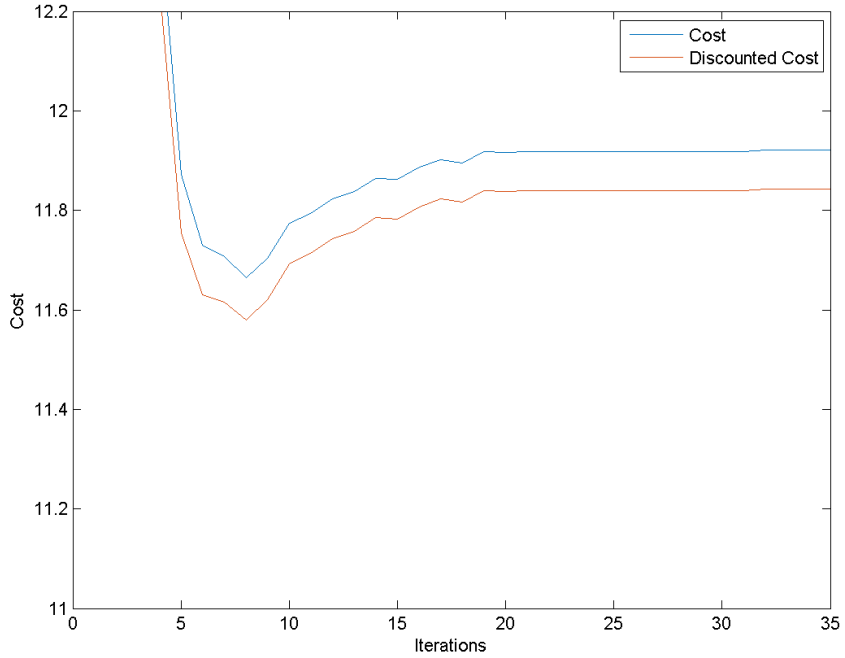


Figure 5-18: Cost (Zoom) with input-saturation [-10,10].

5-3-2 Nonlinear policy iteration algorithm

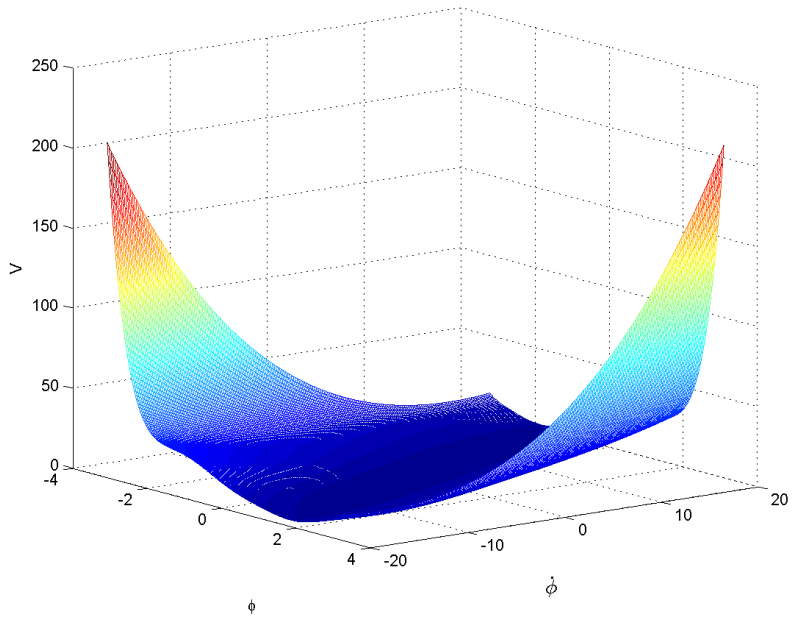


Figure 5-19: Value function with input-saturation [-10,10].

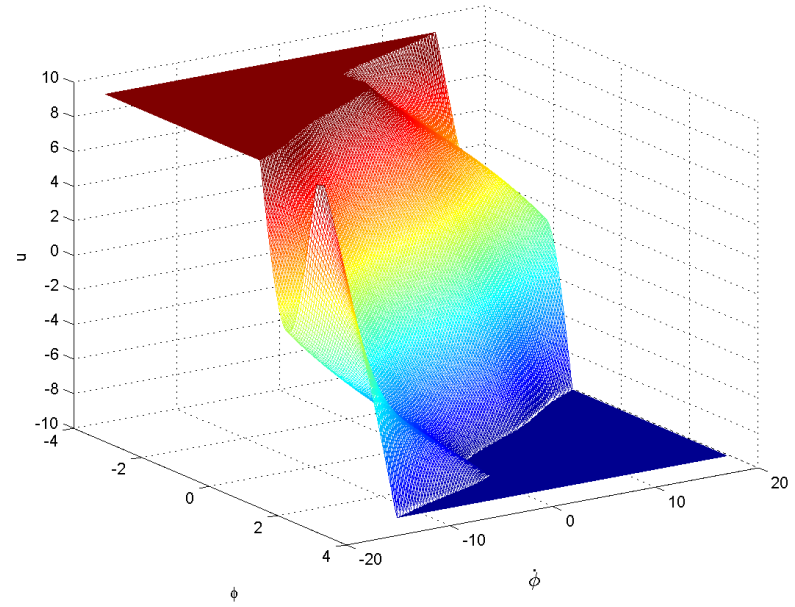


Figure 5-20: Control policy with input-saturation [-10,10].

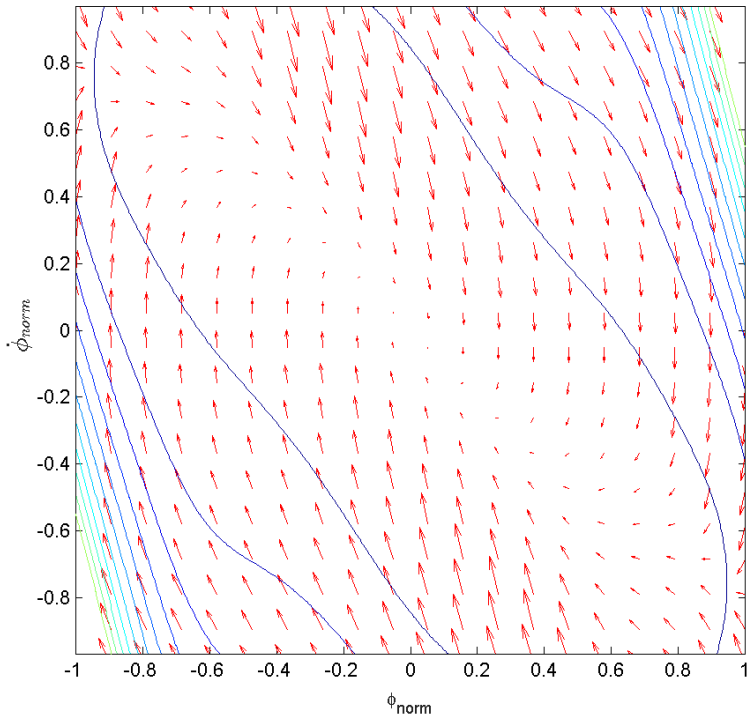


Figure 5-21: Phase phane with input-saturation [-10,10].

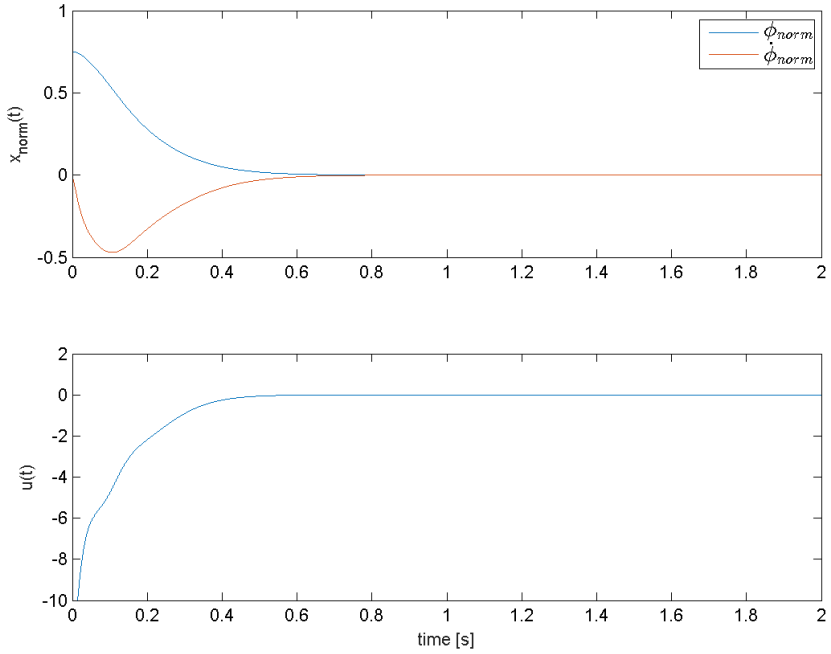


Figure 5-22: Final trajectory with input-saturation [-10,10].

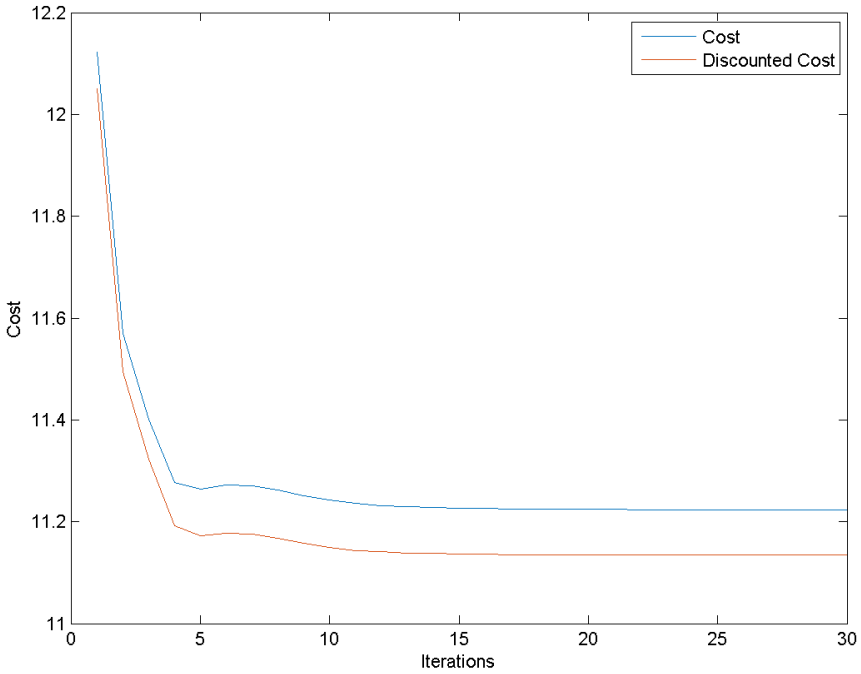


Figure 5-23: Cost with input-saturation [-10,10].

Table 5-4: Cost Comparison with Input-Saturation $[-10, 10]$

	Final discounted cost	Improvement
MBAC	11.82	
Nonlinear PI	11.18	5.4%

5-4 Simulation Results with Input-Saturation [-5,5]

5-4-1 Model based actor-critic algorithm

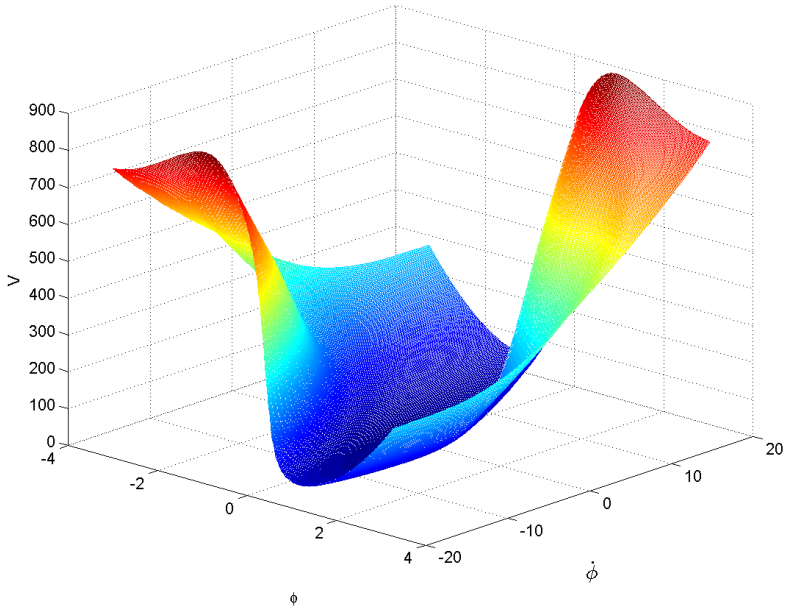


Figure 5-24: Value function with input-saturation [-5,5].

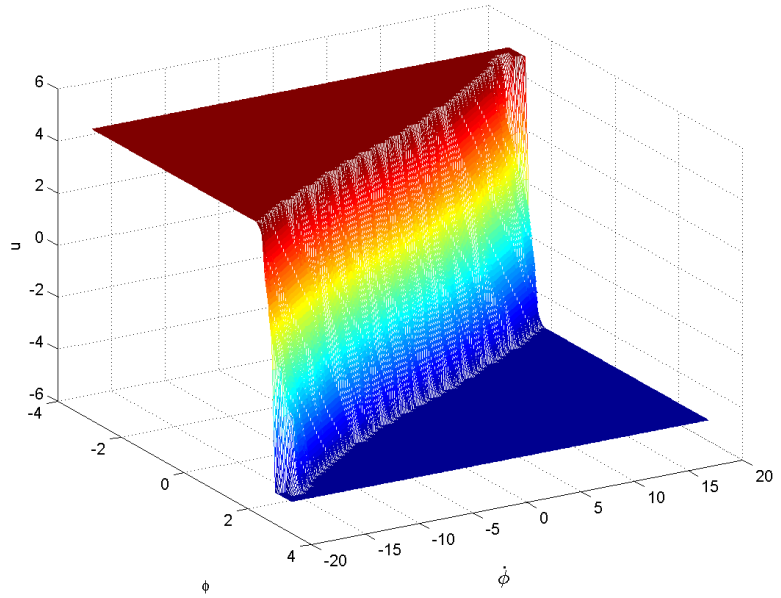


Figure 5-25: Control policy with input-saturation [-5,5].

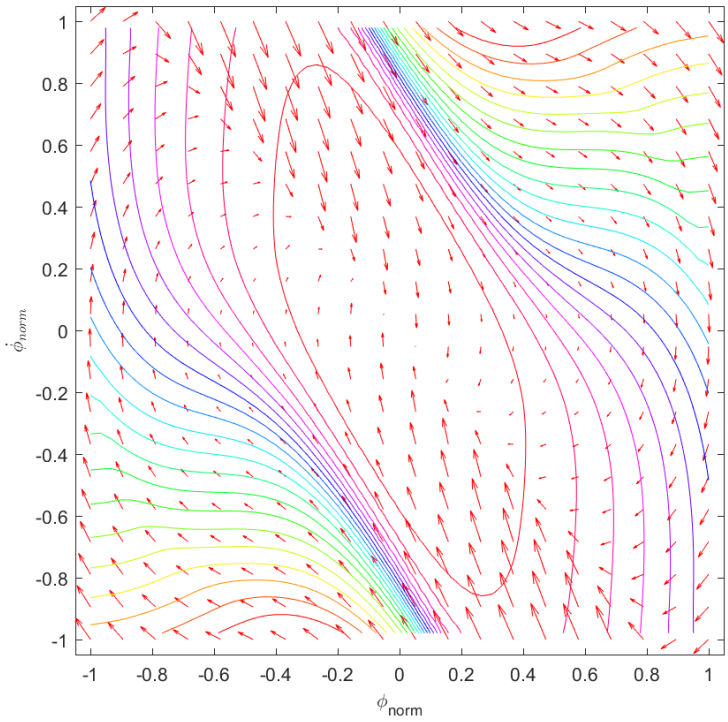


Figure 5-26: Phase phane with input-saturation [-5,5].

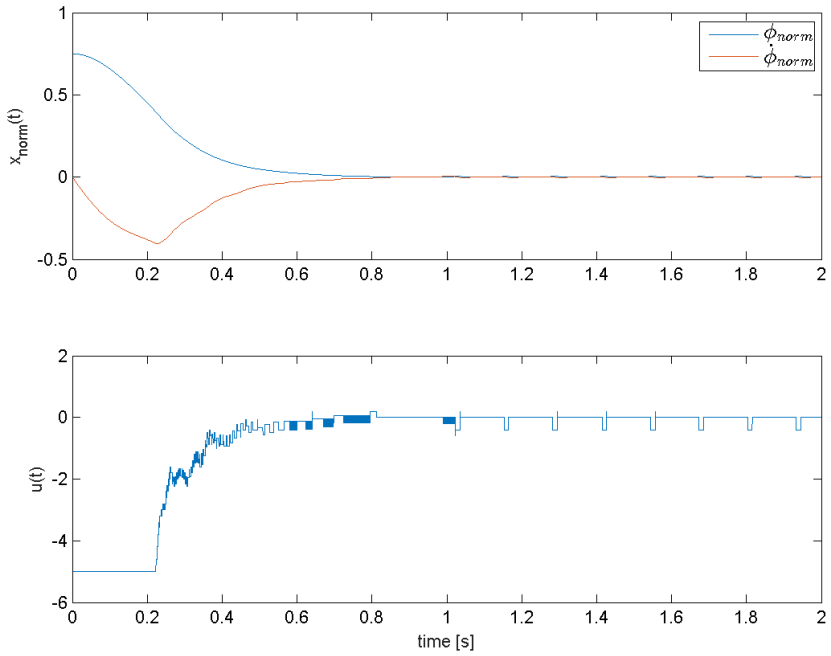


Figure 5-27: Final trajectory with input-saturation [-5,5].

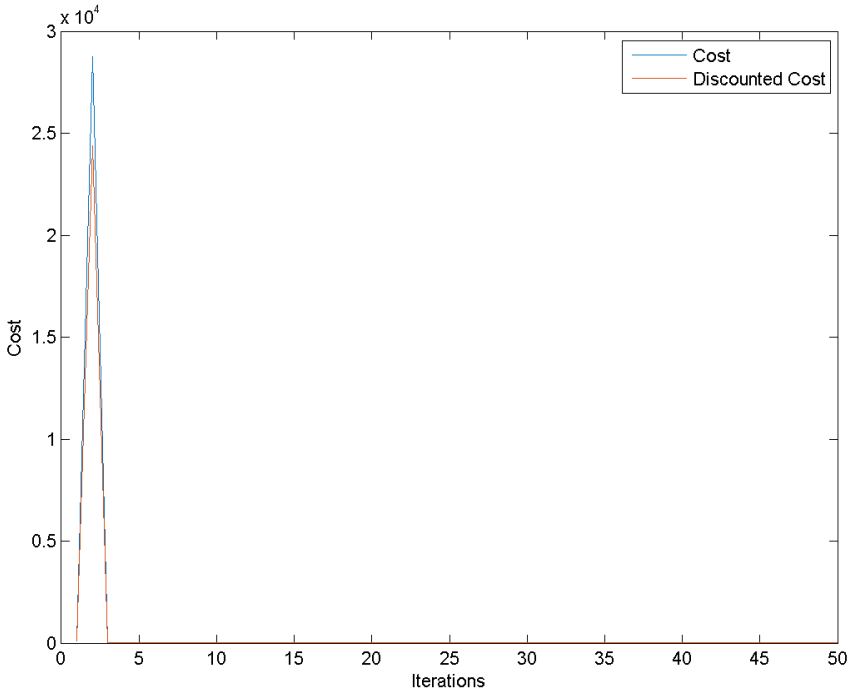


Figure 5-28: Cost with input-saturation [-5,5].

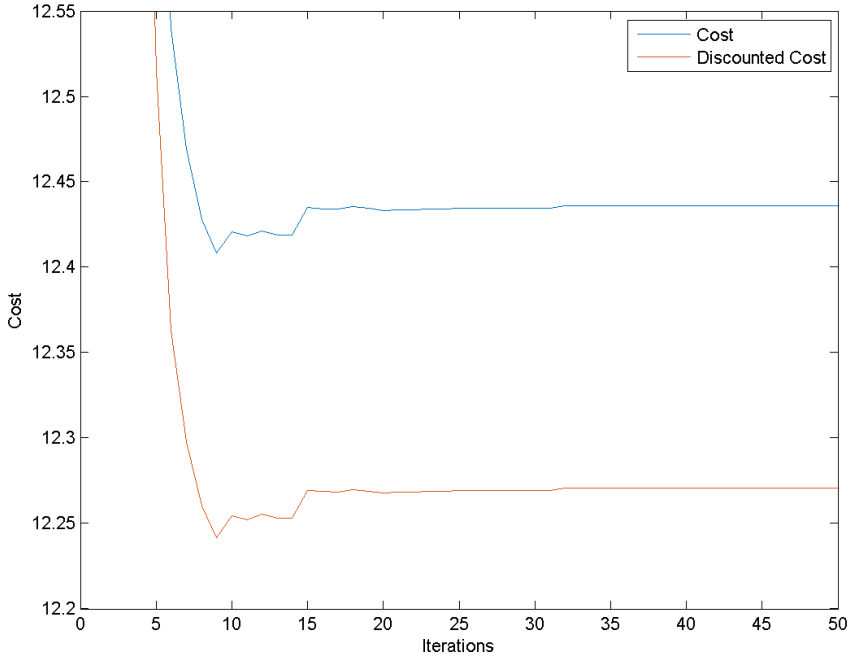


Figure 5-29: Cost (Zoom) with input-saturation [-5,5].

5-4-2 Nonlinear policy iteration algorithm

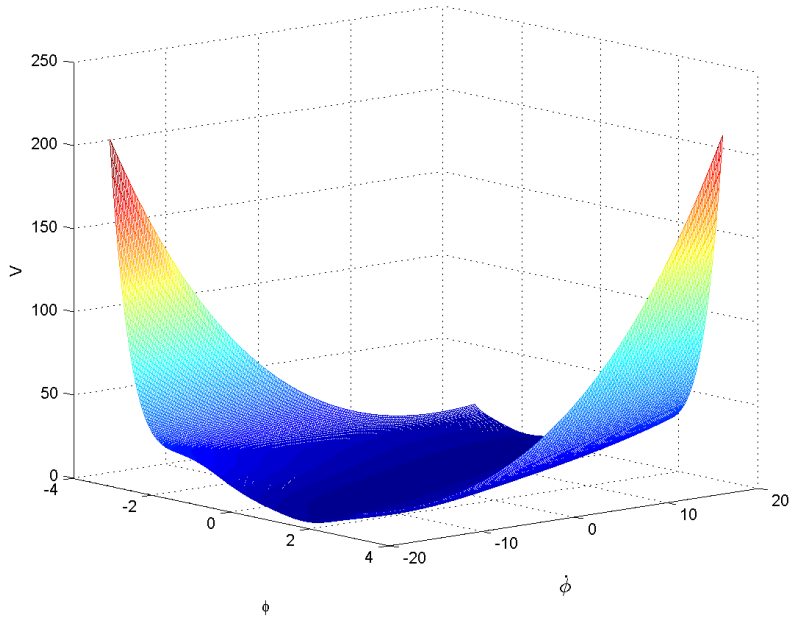


Figure 5-30: Value function with input-saturation [-5,5].

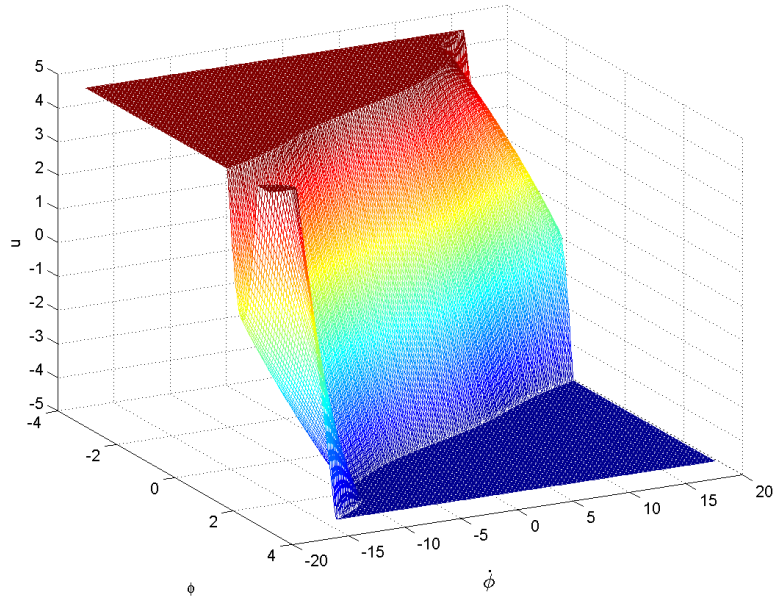


Figure 5-31: Control policy with input-saturation [-5,5].

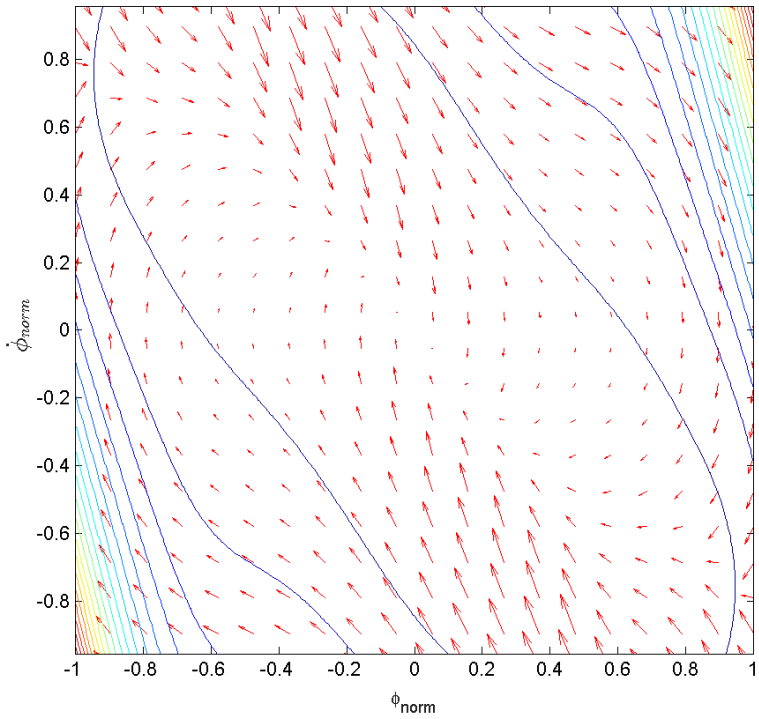


Figure 5-32: Phase phane with input-saturation [-5,5].

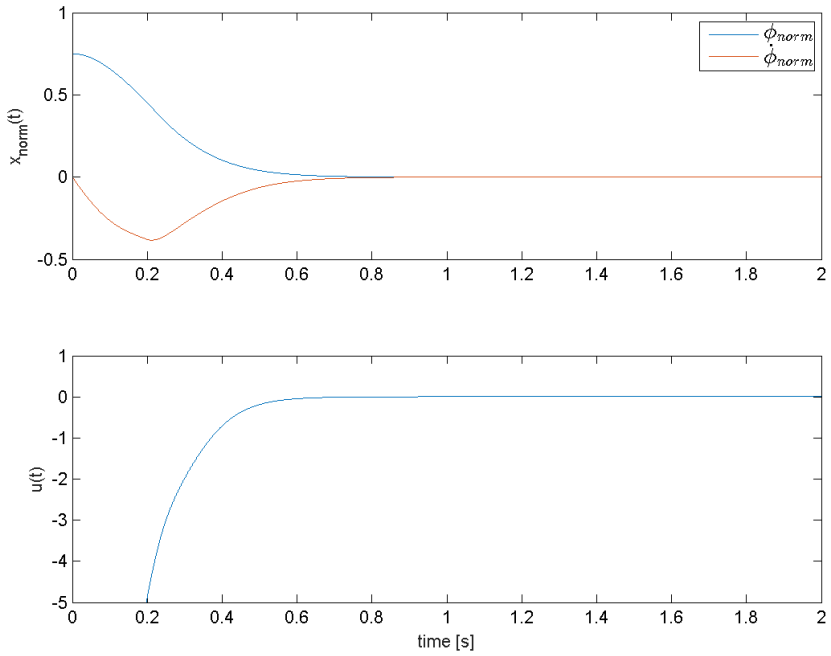


Figure 5-33: Final trajectory with input-saturation [-5,5].

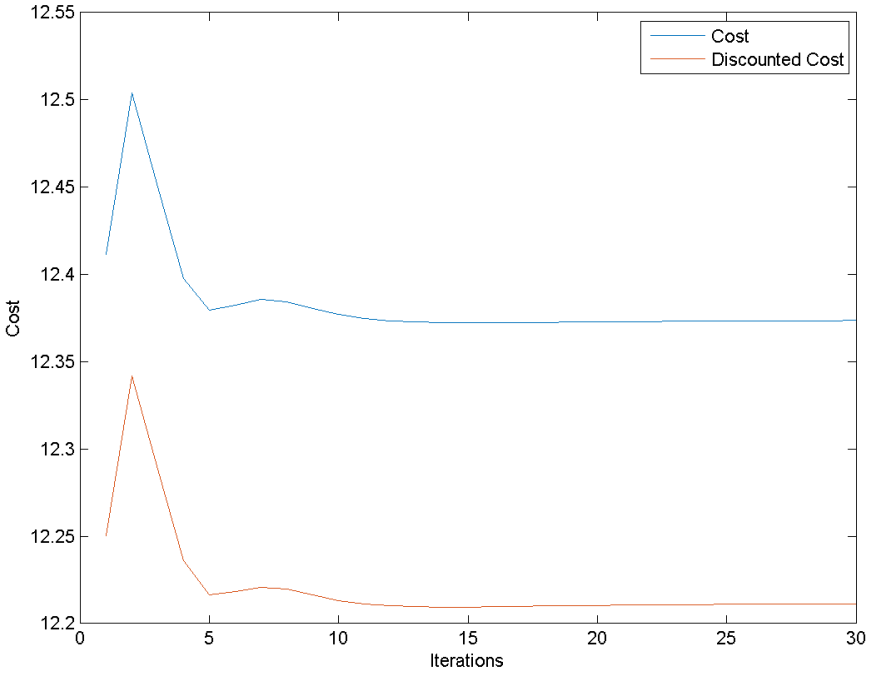


Figure 5-34: Cost with input-saturation [-5,5].

Table 5-5: Cost Comparison with Input-Saturation $[-5, 5]$

	Final discounted cost	Improvement
MBAC	12.28	
Nonlinear PI	12.21	0.6%

5-5 Comments on Simulation Results

Based on the results of Section 5.3, Section 5.4 and Section 5.5, the following comments can be made:

For model based actor-critic method:

- Even if ideally the NN approximation can approximate any smooth function to arbitrary precision, it looks like the MBAC algorithm is not always able to reach the optimum. This is potentially evident with saturation$[-20, 20]$, where the final result is at least 25% far from the optimal.
- The computational complexity of the MBAC approach depends on 2 factors: the discretization of the state and input; the number of NN used to approximate the value function and policy function. For a discretization of 0.2 and adopting 30×30 NNs (respectively in the $x_1 - x_2$ plane), the MBAC approach is computationally less expensive than the Nonlinear Policy Iteration approach. However, adopting 40×40 NNs, the approach was leading to OUT OF MEMORY problems.

For nonlinear policy iteration method:

- In contrast to the MBAC approach, where the value function is stored in a memory, in the nonlinear policy iteration approach the value function is a parameterized function. This leads to a control action which is in general smoother, while in the MBAC approach the policy must be calculated according to the state stored in the memory which is close to the current state. For this reason, the control action of the MBAC approach can result "bumpy". Some improvement can be achieved in the MBAC case by mixing the control action according to several neighbors to the current state.
- The nonlinear PI approach guarantees that every new control policy will be stabilizing (provided that the initial one is stabilizing). Furthermore, the nonlinear PI approach will generally lead to a monotonically decreasing cost (at least in the unsaturated region), whereas in the MBAC approach neither stabilization nor monotonic convergence can be guaranteed. In particular, in the MBAC approach, it is observed that the best value function is not always corresponding to the last one.
- The improvement of the nonlinear PI approach is generally decreasing as the saturation becomes tighter and tighter. This can be explained by that when the control authority decreases, the freedom to improve the cost becomes smaller and smaller.

Chapter 6

Conclusions and Suggestions for Future Work

6-1 Conclusions

In this work, two methodologies for optimal control of nonlinear systems have been slightly revised and applied to the optimal swing up control of a pendulum. Different saturation levels have been assumed to derived the corresponding optimal value and policy functions.

The first strategy considered in this thesis used model learning methods for actor-critic control, which is a novel model-based update rule for the actor parameters. The second strategy considered in this thesis applied a policy iteration procedure for the synthesis of optimal and global stabilizing control policies for Linear Time Invariant (LTI) systems with input-saturation. In this thesis, this strategy was extended to nonlinear polynomial systems via appropriate transformations.

In particular, the first approach has been slightly revised so as to exploit the information of the process model to update the policy. The second approach, which was originally formulated for linear systems, has been extended to nonlinear system via Sum-of-Squares decomposition (SOS). Since SOS methods are applicable to polynomial systems, appropriate transformations have been adopted to recast the pendulum model as a polynomial system.

Simulation results show that although ideally the NN approximation can approximate any smooth function to arbitrary precision, it looks like the MBAC algorithm is not always able to reach the optimum. The computational complexity of the MBAC approach depends on both the discretization of the state and input and the number of NN used to approximate the value function and policy function. In contrast to the MBAC approach, the nonlinear PI approach guarantees that every new control policy will be stabilizing and generally lead to a monotonically decreasing cost, whereas in the MBAC approach neither stabilization nor monotonic convergence can be guaranteed. In particular, in the MBAC approach, it is observed that the best value function is not always corresponding to the last one. However, the

improvement of the nonlinear PI approach is generally decreasing as the saturation becomes tighter and tighter.

It must be recognized that both methods are prone to computational complexity issues as the desired precision increases. In particular, the MBAC approach can lead to OUT OF MEMORY problems if the discretization of states and inputs is finer, or if the number of neural networks increases. On the other hand, the nonlinear policy iteration method will exhibit increased computational complexity as the degree of the introduced polynomial increase or as the state will become of larger and larger dimension.

6-2 Suggestions for Future Work

Suggestions for future work will include,

- Extending the piecewise polynomial policy iteration methodology to linear system with exponentially unstable modes or to nonlinear locally stabilization system. It is worth mentioning that for linear systems with exponentially unstable modes, it is not possible to achieve global stabilization with bounded-input. For this reason, an optimal value function and control policy can be found only locally.
- Including a generalized sector condition which is instrumental to compute estimations of the region of attraction.
- Further increase the level of saturation in such a way that a truly swing up action is required (eg, saturation $[-3, 3]$ or $[-1, 1]$). It has to be noticed that in such case a piecewise nonlinear control can be adopted: a swing-up and stabilization controller of an inverted pendulum system must be diverge the pendulum from the stable position, that is the hung down position, while a stabilization controller must stand the pendulum in the unstable position, that is the standing position. Therefore, the swing-up controller and the stabilization controller have to be designed individually in the case of using a control technique based on a linear model, for example, an optimal control theory.

Appendix A

Sum of Squares Decomposition

We will give a brief introduction to sum of squares (SOS) polynomials and show how the existence of an SOS decomposition can be verified using semidefinite programming [43]. A more detailed description can be found in [44] [45] and the references therein.

Definition. For $x \in \mathbb{R}^n$, a multivariate polynomial $p(x)$ is an SOS if there exist some polynomials $f_i(x), i = 1...M$ such that $p(x) = \sum_{i=1}^{M} f_i^2(x)$.

An equivalent characterization of SOS polynomials is given in the following proposition.

Proposition. A polynomial $p(x)$ of degree $2d$ is an SOS if and only if there exists a positive semidefinite matrix Q and a vector of monomials $Z(x)$ containing all monomials in x of degree $\leq d$ such that $p = Z(x)^T Q Z(x)$.

The proof of this proposition is based on the eigenvalue decomposition and can be found in [44]. In general, the monomials in $Z(x)$ are not algebraically independent. Expanding $Z(x)^T Q Z(x)$ and equating the coefficients of the resulting monomials to the ones in $p(x)$, we obtain a set of affine relations in the elements of Q. Since $p(x)$ being SOS is equivalent to $Q \geq 0$, the problem of finding a Q which proves that $p(x)$ is an SOS can be cast as a semidefinite program (SDP). This was observed by Parrilo in [44].

Note that $p(x)$ being an SOS implies that $p(x) \geq 0$ for all $x \in \mathbb{R}^n$. However, the converse is not always true. Not all nonnegative polynomials can be written as SOS, apart from three special cases: (i) when $n = 2$, (ii) when $deg(p) = 2$, and (iii) when $n = 3$ and $deg(p) = 4$. See [46] for more details. Nevertheless, checking nonnegativity of $p(x)$ is an NP-hard problem when the degree of $p(x)$ is at least 4 [47], whereas as argued in the previous paragraph, checking whether $p(x)$ can be written as an SOS is computationally tractable. It can be formulated as an SDP, which has worst-case polynomial time complexity. We will not entail in a discussion on how conservative the relaxation is, but there are several results suggesting that this is not too conservative [46]. Note that as the degree of $p(x)$ or its number of variables is increased, the computational complexity for testing whether $p(x)$ is an SOS increases. Nonetheless, the complexity overload is still a polynomial function of these parameters.

There is a close connection between sum of squares and robust control theory through Positivstellensatz, a central theorem in Real Algebraic Geometry [48]. This theorem allows us

to formulate a hierarchy of polynomial-time computable stronger conditions [44] for the S-procedure type of analysis [49]. To see how we will be using this result say we want to use the S-procedure to check that the set:

$$\{p(x) \geq 0 \quad \text{when } p_i(x) \geq 0 \text{ for } i = 1, ..., n\}$$

is non-empty. Instead of finding positive constant multipliers (the standard S-procedure), we search for SOS multipliers $h_i(x)$ so that

$$p(x) - \sum_i h_i(x)p_i(x) \text{ is a SOS.} \tag{A-1}$$

Since $h_i(x)$ and condition (A-1) is satisfied, for any x such that $p_i(x)$ we automatically have $p(x) \geq 0$, so sufficiency follows. This condition is at least as powerful as the standard S-procedure, and many times it is strictly better; it is a special instance of positivstellensatz. By putting an upper bound on the degree of h_i, we can get a nested hierarchy of polynomial-time checkable conditions.

Besides this, what is more interesting is the case in which the monomials in the polynomial $p(x)$ have unknown coefficients, and we want to search for some values of those coefficients such that $p(x)$ is a sum of squares. Since the unknown coefficients of $p(x)$ are related to the entries of Q via affine constraints, it is evident that the search for the coefficients that make $p(x)$ an SOS can also be formulated as an SDP (these coefficients are themselves decision variables). This observation is crucial in the construction of Lyapunov functions and other S-procedure type multipliers.

Construction of an equivalent SDP for computing SOS decomposition as in Proposition can be quite involved when the degree of the polynomials is high. For this reason, conversion of SOS conditions to the corresponding SDP has been automated in SOSTOOLS [50], a software package developed for this purpose. This software calls SeDuMi [51], an SDP solver to solve the resulting SDP, and converts the solutions back to the solutions of the original SOS programs.

Appendix B

Three examples of converting a non-polynomial system into a polynomial system

In the first example the vector field of the system includes a radical term. Such terms appear frequently when considering systems with saturation nonlinearities. The second example shows how one can analyze non-polynomial vector fields with irrational powers, which for example appear in models of biological systems. In the third example we analyze a system that appears frequently in chemical engineering, that of a diabatic Continuous Stirred Tank Reactor (CSTR) with a single first-order exothermic rreversible reaction $A \rightarrow B$. Furthermore, the sum of squares programs corresponding to these examples are converted into semidefinite programs using SOSTOOLS, and are solved using SDP3 or SeDuMi, a semidefinite solver.

Example 1: System with saturation Nonlinearity

For a general system, finding a global Lyapunov function is difficult, as one of polynomial form in the variables considered might not exist. However, if a term is expected to appear in a Lyapunov function, the search can be directed to include that term in the Lyapunov function expression sought, by changing variables and recasting the system in an equivalent one in terms of that variable, making use of inequality and equality constraints.

Consider the system

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -\varphi(x_1 + x_2) \end{aligned}$$

where the function φ is a saturation function of the following form:

$$\varphi(\sigma) = \frac{\sigma}{\sqrt{1+\sigma^2}}$$

This function has only one equilibrium point, namely the origin. First rewrite the above system in a polynomial form, as it has non-polynomial terms. For this purpose, introduce the

following auxiliary variables:

$$\begin{aligned} u_1 &= \sqrt{1+(x_1+x_2)^2}, \\ u_2 &= 1/u_1, \\ u_3 &= \sqrt{1+x_1^2}, \\ u_4 &= 1/u_3. \end{aligned}$$

Then the equations of motion for the above system become

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -(x_1+x_2)u_2, \\ \dot{u}_1 &= (x_1+x_2)(x_2-x_1u_2-x_2u_2)u_2, \\ \dot{u}_2 &= -(x_1+x_2)(x_2-x_1u_2-x_2u_2)u_2^3, \\ \dot{u}_3 &= x_1x_2u_4, \\ \dot{u}_4 &= -x_1x_2u_4^3. \end{aligned}$$

In addition, we have a number of equality and inequality constraints

$$\begin{aligned} u_1^2 &= 1+(x_1+x_2)^2, \\ u_1u_2 &= 1, \\ u_3^2 &= 1+x_1^2, \\ u_3u_4 &= 1, \\ u_i &\geq 0, \quad \text{for } i=1,2,3,4. \end{aligned}$$

Now the system is in the form (4-10) (4-11) with $\tilde{x}_1 = (x_1, x_2)$ and $\tilde{x}_2 = (u_1, u_2, u_3, u_4)$. The above constraints correspond to (4-12). The new representation allows us to use SOS decomposition to compute a Lyapunov function for this problem, using Proposition 4. Thus, for example, we may search for a Lyapunov function of the following form:

$$V = a_1 + a_2u_3 + a_3x_1^2 + a_4x_1x_2 + a_5x_2^2$$

where the $a_i's$ are the unknowns, with $a_1+a_2=0$, so that V is equal to zero at $(x_1, x_2) = (0, 0)$. To guarantee positive definiteness, we require V to satisfy

$$(V - \epsilon_1(u_3-1) - \epsilon_2x_1^2 - \epsilon_3x_2^2) \quad \text{is sum of squares.}$$

with $\epsilon_1, \epsilon_2, \epsilon_3$ being non-negative decision variables that satisfy, for example,

$$\begin{aligned} \epsilon_1+\epsilon_2 &\geq 0.1, \\ \epsilon_3 &\geq 0.1 \end{aligned}$$

Using this method, a Lyapunov function has been constructed for the system:

$$\begin{aligned} V &= -3.9364 + 3.9364u_3 + 0.0063889x_1^2 + 0.010088x_1x_2 + 2.0256x_2^2 \\ &= -3.9364 + 3.9364\sqrt{1+x_1^2} + 0.0063889x_1^2 + 0.010088x_1x_2 + 2.0256x_2^2 \end{aligned}$$

The level curves of this Lyapunov function are shown in Figure B-1. Arrows show vector field, solid lines show level curves of the Lyapunov function.

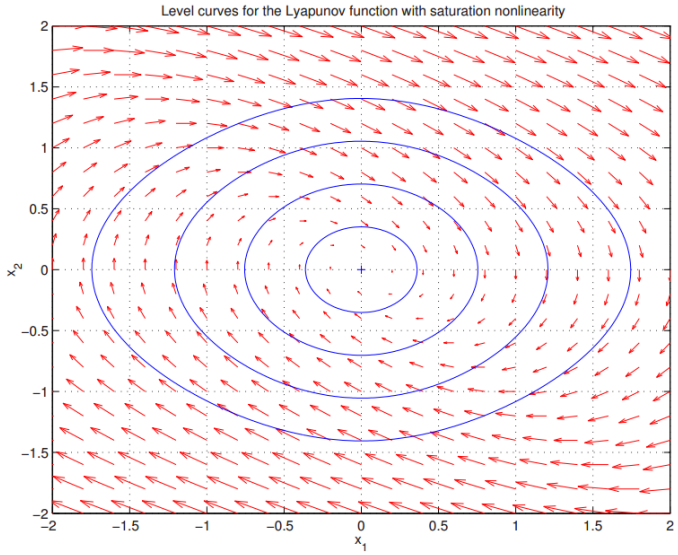


Figure B-1: Global Lyapunov function for the system with saturation nonlinearity present [3].

Example 2: System with an non-polynomial vector field

Consider a simple one dimensional system:

$$\dot{x} = x^{\alpha} - 1, \qquad x \in \mathbb{R}_{+}$$

where α is a parameter. The linearisation of this system about the equilibrium $x = 1$ is $\dot{x} = \alpha x$ which implies that the system is locally stable for $\alpha < 0$. Let us make a transformation $y = x - 1$ to the above system to put its equilibrium at the origin:

$$\dot{y} = (y + 1)^{\alpha} - 1$$

Further to this transformation, we introduce the transformation $z = (y+1)^{\alpha} - 1$ and embed the system into a second order system with a polynomial vector field and an equality constraint

$$\dot{y} = z \tag{B-1}$$

$$\dot{z} = \alpha \frac{(z + 1)z}{y + 1} \tag{B-2}$$

In addition, we have a constraint

$$z = (y + 1)^{\alpha} - 1 \tag{B-3}$$

The non-polynomial equality constraint (B-3) cannot be imposed in the sum of squares program in a similar manner as before. To proceed with the analysis, we will try to prove stability of the two dimensional system without the equality constraint. We will attempt to prove stability for

$$\alpha - \alpha_h \leq 0 \tag{B-4}$$
$$-y + y_l \leq 0 \tag{B-5}$$
$$-z + z_l \leq 0 \tag{B-6}$$

We set $\alpha_h = -0.1$ and $y_l = -0.9$. This dictates that $z \geq (y_l + 1)^{\alpha_h} - 1 \triangleq z_l$. We search for a 4th order Lyapunov function in y, z, but we do not require V to be positive definite in both y and z, by constructing $\phi(y, z)$ in (4-21) appropriately. In particular the two Lyapunov conditions become:

$$V(y, z; \alpha) - \phi(y, z) \geq 0,$$
$$-\frac{\partial V}{\partial y}\dot{y} - \frac{\partial V}{\partial z}\dot{z} \leq 0$$

for α, y, z satisfying (B-4)-(B-6), and additionally where

$$\phi(y, z) = \epsilon_1 y^2 + \epsilon_2 y^4 + \epsilon_3 z^2 + \epsilon_4 z^4,$$
$$\sum_{i=1}^{4} \epsilon_i \geq 0.01, \quad \epsilon_i \geq 0, \quad \forall i = 1, ..., 4.$$

The inequality constraints (B-4)-(B-6) can be adjoined to the two conditions and a sum of squares program can be written using SOSTOOLS as in the previous examples. Indeed, such a Lyapunov function was constructed which allows for stability to be concluded.

Example 3. Diabatic Continuous Stirred Tank Reactor

Chemical reactors are the most important unit operation in a chemical process. In this example, we consider the analysis of the dynamics of a perfectly mixed, diabatic, continuously stirred tank reactor (CSTR) [52]. We also assume a constant volume-constant parameter system for simplicity.

The reaction taking place in the CSTR is a first-order exothermic irreversible reaction $A \to B$. After balancing mass and energy, the reactor temperature T and the concentration of species A in the reactor C_A evolve as follows:

$$\dot{C}_A = \frac{F}{V}(C_{A_f} - C_A) - k_0 e^{-\frac{\Delta E}{RT}} C_A \tag{B-7}$$
$$\dot{T} = \frac{F}{V}(T_f - T) - \frac{\Delta H}{\rho c_p} k_0 e^{-\frac{\Delta E}{RT}} C_A - \frac{UA}{V \rho c_p}(T - T_j) \tag{B-8}$$

where F is the volumetric flow rate, V is the reactor volume, C_{A_f} is the concentration of A in the freestream, k_0 is the pre-exponential factor of Arrhenius law, ΔE is the reaction activation energy, R is the ideal gas constant, T_f is the feed temperature, $-\Delta H$ is the heat of reaction (exothermic), ρ is the density, c_p is the heat capacity, U is the overall heat transfer coefficient, A is the area for heat exchange, and T_j is the jacket temperature.

The equilibrium of the above system is given by $(C_{A_0}, T_0) = (8.5636, 311.171)$. We employ the following transformation:

$$x_1 = \frac{C_A}{C_{A_0}} - 1$$
$$x_2 = \frac{T}{T_0} - 1$$

This serves two purposes: firstly it moves the equilibrium to the origin, and secondly it rescales the state to avoid numerical ill-conditioning. Then the transformed system then becomes:

$$\dot{x}_1 = \frac{F}{V}\left(\frac{C_{A_f}}{C_{A_0}} - (x_1+1)\right) - k_0 e^{-\frac{\Delta E}{RT_0(x_2+1)}}(x_1+1) \tag{B-9}$$

$$\dot{x}_2 = \frac{F}{V}\left(\frac{T_f}{T_0} - (x_2+1)\right) - \frac{\Delta H C_{A_0}}{\rho c_p T_0} k_0 e^{-\frac{\Delta E}{RT_0(x_2+1)}}(x_1+1) - \frac{UA}{V\rho c_p}\left((x_2+1) - \frac{T_j}{t_0}\right) \tag{B-10}$$

Note that the system has an exponential term; the recasting will yield an indirect constraint. Define the state $x_3 = e^{-\frac{\Delta E}{RT_0(x_2+1)}} - 1$. Then an extra equation in the analysis would be

$$\dot{x}_3 = \frac{\Delta E}{RT_0(x_2+1)^2}(x_3+1)\dot{x}_2 \tag{B-11}$$

under the constraints that $x_3 > -1$. Then the full system, after we use the equilibrium relationship simplifies to:

$$\dot{x}_1 = -\frac{F}{V}x_1 - k_0 e^{-\frac{\Delta E}{RT_0}}(x_1x_3 + x_1 + x_3) \tag{B-12}$$

$$\dot{x}_2 = -\frac{F}{V}x_2 - \frac{\Delta H C_{A_0}}{\rho c_p T_0} k_0 e^{-\frac{\Delta E}{RT_0}}(x_1x_3 + x_1 + x_3) - \frac{UA}{V\rho c_p}x_2 \tag{B-13}$$

$$\dot{x}_3 = \frac{\Delta E}{RT_0(x_2+1)^2}(x_3+1)\dot{x}_2 \tag{B-14}$$

This system is now of the form (4-10)-(4-11), with $\tilde{x}_1 = (x_1, x_2)$ and $\tilde{x}_2 = x_3$. To proceed, we define the set $\mathcal{D}_1$ and $\mathcal{D}_2$ as:

$$\mathcal{D}_1 = \{(x_1, x_2) \in \mathbb{R}^2 : |x_1| \leq \gamma_1, |x_2| \leq \gamma_2\}$$

$$\mathcal{D}_2 = \{x_3 \in \mathbb{R} : (x_3 - e^{\frac{-\Delta E\gamma_2}{RT_0(-\gamma_2+1)}} - 1)(x_3 - e^{\frac{\Delta E\gamma_2}{RT_0(-\gamma_2+1)}} - 1) \leq 0\}$$

Then the system is ready for analysis as Proposition 4. For $\gamma_1 = 0.12$ and $\gamma_2 = 0.05$ a quartic Lyapunov function can be constructed for the system described by (B-12)-(B-14) using Proposition 4. Here the following $\phi(x)$ is used:

$$\phi(x) = \sum_{i=1}^{2}\sum_{j=2,4}\epsilon_{i,j}x_i^j + \sum_{j=1}^{4}\epsilon_{3,j}x_3^j$$

with

$$\epsilon_{1,2} + \epsilon_{1,4} - 0.1 \geq 0$$
$$\epsilon_{2,2} + \epsilon_{2,4} + \epsilon_{3,1} + \epsilon_{3,2} + \epsilon_{3,3} + \epsilon_{3,4} - 0.1 \geq 0.$$

The level curves of the constructed Lyapunov function are shown in Figure B-2

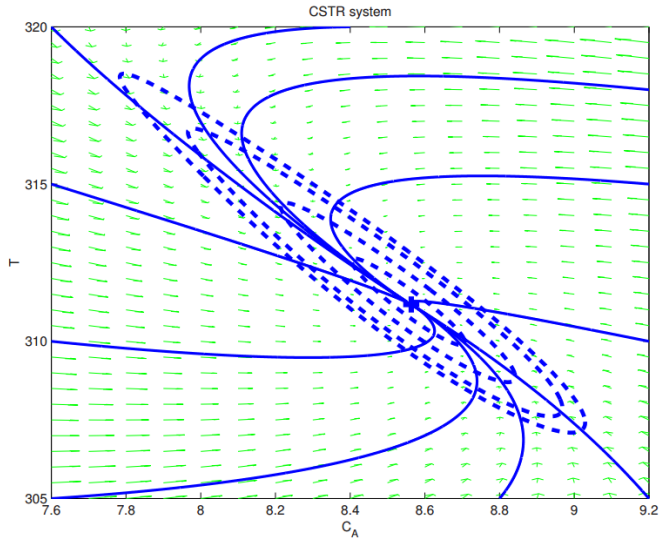


Figure B-2: Lyapunov function level curves for the system (B-7)-(B-8).

Bibliography

[1] Ivo Grondman, Maarten Vaandrager, Lucian Busoniu, Robert Babuska, and Erik Schuitema. Efficient model learning methods for actor–critic control. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(3):591–602, 2012.

[2] Simone Baldi, Giorgio Valmorbida, Antonis Papachristodoulou, and Elias B Kosmatopoulos. Piecewise polynomial policy iterations for synthesis of optimal control laws in input-saturated systems. In *American Control Conference (ACC), 2015*, pages 2850–2855. IEEE, 2015.

[3] Antonis Papachristodoulou and Stephen Prajna. Analysis of non-polynomial systems using the sum of squares decomposition. In *Positive polynomials in control*, pages 23–43. Springer, 2005.

[4] Kazunobu Yoshida. Swing-up control of an inverted pendulum by energy-based methods. In *Proceedings of the American control conference*, volume 6, pages 4045–4047, 1999.

[5] Shozo Mori, Hiroyoshi Nishihara, and Katsuhisa Furuta. Control of unstable mechanical system control of pendulum. *International Journal of Control*, 23(5):673–692, 1976.

[6] Meier Farwig, H Zu, and H Unbehauen. Discrete computer control of a triple-inverted pendulum. *Optimal Control Applications and Methods*, 11(2):157–171, 1990.

[7] Masaki Yamakita, Katsuhisa Furuta, Kazuteru Konohara, Junichi Hamada, and Hitoshi Kusano. Vss adaptive control based on nonlinear model for titech pendulum. In *Industrial Electronics, Control, Instrumentation, and Automation, 1992. Power Electronics and Motion Control., Proceedings of the 1992 International Conference on*, pages 1488–1493. IEEE, 1992.

[8] Alexander L Fradkov and Alexander Yu Pogromsky. Speed gradient control of chaotic continuous-time systems. *IEEE Transactions on Circuits and Systems I Fundamental Theory and Applications*, 43(11):907–913, 1996.

[9] Miroslav Krstić, Ioannis Kanellakopoulos, and Petar V Kokotović. Passivity and parametric robustness of a new class of adaptive systems. *Automatica*, 30(11):1703–1716, 1994.

[10] Johan Eker and Karl Johan Åström. A nonlinear observer for the inverted pendulum. In *Control Applications, 1996., Proceedings of the 1996 IEEE International Conference on*, pages 332–337. IEEE, 1996.

[11] Carl Fredrik Abelson. The effect of friction on stabilization of an inverted pendulum. *MSc Theses*, 1996.

[12] Katsuhisa Furuta, Masaki Yamakita, and Seiichi Kobayashi. Swing up control of inverted pendulum. In *Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON'91., 1991 International Conference on*, pages 2193–2198. IEEE, 1991.

[13] R Lozano and I Fantoni. Passivity based control of the inverted pendulum. In *Perspectives in Control*, pages 83–95. Springer, 1998.

[14] Qifeng Wei, Wijesuriya P Dayawansa, and WS Levine. Nonlinear controller for an inverted pendulum having restricted travel. *Automatica*, 31(6):841–850, 1995.

[15] John Guckenheimer. A robust hybrid stabilization strategy for equilibria. *IEEE Transactions on Automatic Control*, 40(2):321–326, 1995.

[16] Troy Shinbrot, Celso Grebogi, Jack Wisdom, and James A Yorke. Chaos in a double pendulum. *Am. J. Phys*, 60(6):491–499, 1992.

[17] Yu Jiang and Zhong-Ping Jiang. Global adaptive dynamic programming for continuous-time nonlinear systems. *Automatic Control, IEEE Transactions on*, 60(11):2917–2929, 2015.

[18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[19] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *Systems, Man and Cybernetics, IEEE Transactions on*, (5):834–846, 1983.

[20] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press, 2010.

[21] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999.

[22] Dietrich Wettschereck, David W Aha, and Takao Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11(1-5):273–314, 1997.

[23] Thomas Gabel and Martin Riedmiller. Cbr for state value function approximation in reinforcement learning. In *Case-Based Reasoning Research and Development*, pages 206–221. Springer, 2005.

[24] Jon Louis Bentley and Jerome H Friedman. Data structures for range searching. *ACM Computing Surveys (CSUR)*, 11(4):397–409, 1979.

[25] Richard S Sutton. Reinforcement learning architectures. *Proceedings ISKIT*, 92, 1992.

[26] Andrew W Moore and Christopher G Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, 1993.

[27] Leonid Kuvayev and Rich Sutton. Model-based reinforcement learning with an approximate, learned model. In *in Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*. Citeseer, 1996.

[28] Gerhard Neumann and Jan R Peters. Fitted q-iteration by advantage weighted regression. In *Advances in neural information processing systems*, pages 1177–1184, 2009.

[29] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pages 363–372. Springer, 2006.

[30] Jeffrey Forbes and David Andre. Representations for learning control policies. In *Proceedings of the ICML-2002 Workshop on Development of Representations*, pages 7–14, 2002.

[31] Richard E Bellman. *Adaptive control processes: a guided tour*. Princeton university press, 2015.

[32] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[33] Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pages 560–564. IEEE, 1995.

[34] Eduardo D Sontag. An algebraic approach to bounded controllability of linear systems. *International Journal of Control*, 39(1):181–188, 1984.

[35] Kirk Donald. Optimal control theory: An introduction. *Mineola, NY: Dover Publications, Inc*, 1970.

[36] George N Saridis and Chun-Sing G Lee. An approximation theory of optimal control for trainable manipulators. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(3):152–159, 1979.

[37] John J Murray, Chadwick J Cox, George G Lendaris, and Richard Saeks. Adaptive dynamic programming. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32(2):140–153, 2002.

[38] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[39] Michael A Savageau and Eberhard O Voit. Recasting nonlinear differential equations as s-systems: a canonical nonlinear form. *Mathematical biosciences*, 87(1):83–115, 1987.

[40] Vladimir Ivanovich Zubov and Leo F Boron. *Methods of AM Lyapunov and their Application*. Noordhoff Groningen, 1964.

[41] Katsuhisa Furuta, M Yamakita, and S Kobayashi. Swing-up control of inverted pendulum using pseudo-state feedback. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 206(4):263–269, 1992.

[42] Jerrold E Marsden and Tudor S Ratiu. Introduction to mechanics and symmetry. *Physics Today*, 48(12):65, 1995.

[43] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[44] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, Citeseer, 2000.

[45] Pablo A Parrilo and Bernd Sturmfels. Minimizing polynomial functions. *Algorithmic and quantitative real algebraic geometry, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 60:83–99, 2003.

[46] Bruce Reznick. Some concrete aspects of hilbert's 17th problem. *Contemporary Mathematics*, 253:251–272, 2000.

[47] Katta G Murty and Santosh N Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2):117–129, 1987.

[48] Jacek Bochnak, Michel Coste, and Marie-Françoise Roy. Real algebraic geometry, volume 36 of a series of modern surveys in mathematics, 1998.

[49] V Ao Yakubovich. S-procedure in nonlinear control theory. *Vestnik Leningrad University*, 1:62–77, 1971.

[50] Stephen Prajna, Antonis Papachristodoulou, and Pablo A Parrilo. Introducing sostools: A general purpose sum of squares programming solver. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 1, pages 741–746. IEEE, 2002.

[51] Jos F Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

[52] B Wayne Bequette and Wayne B Bequette. *Process dynamics: modeling, analysis, and simulation*. Prentice Hall PTR Upper Saddle River, NJ, 1998.

Glossary

List of Acronyms

LTI	Linear Time Invariant
ANCBI	Asymptotically Nullcontrollable with Bounded Inputs
MBAC	Model Based Actor-Critic
MLAC	Model Learning Actor-Critic
NN	Neural Network
PI	Policy Iteration
MDP	Markov decision process
HJB	Hamilton-Jacobi-Bellman
SOS	Sum-of-Squares
SDP	Semidefinite Programming