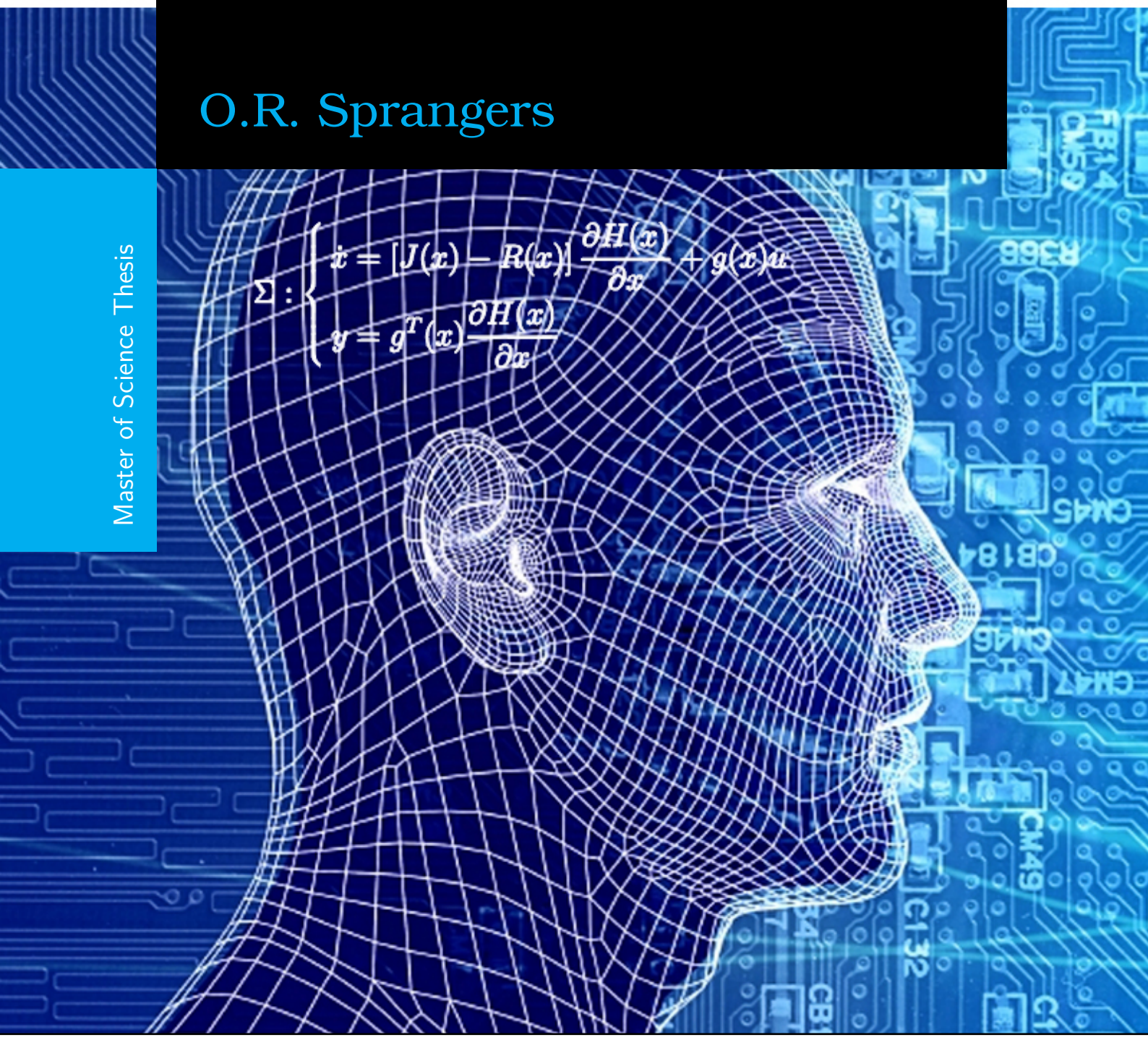
Embedding machine learning into passivity theory: a port-Hamiltonian approach

O.R. Sprangers

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Embedding machine learning into passivity theory: a port-Hamiltonian approach

Master of Science Thesis

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

O.R. Sprangers

May 7, 2012

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

Cover picture edited, original: https://cs.byu.edu/neural_networks_and_machine_learning

Delft University of Technology
Department of
Delft Center for Systems and Control (DCSC)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Embedding machine learning into passivity theory: a port-Hamiltonian approach

by

O.R. Sprangers

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: May 7, 2012

Supervisor(s):

Dr. G.A. Delgado Lopes

Prof.dr. R. Babuška

Reader(s):

Dr.ing. D. Jeltsema

Prof.dr.ir. J.M.A. Scherpen

Ir. I. Grondman

Abstract

Passivity-based control (PBC) is a control methodology that achieves its control objective by rendering a system passive with respect to a desired storage function. A key feature of PBC is that it exploits structural properties of the system. In this thesis, the PBC of systems endowed with a special structure, called port-Hamiltonian (PH) systems, has been investigated. The geometric structure of PH systems allows reformulating the PBC problem in terms of solving a generally complex partial differential equation (PDE).

Reinforcement learning (RL), on the other hand, is a learning control method that can solve complex nonlinear (stochastic) control problems without the need for a process model or explicitly solving a set of equations. In RL the controller receives an immediate numerical reward as a function of the process state and possibly control action. The goal is to find an optimal control policy that maximizes the cumulative long-term rewards, which corresponds to maximizing a value function. In this thesis, actor-critic techniques have been used, which are a class of RL methods in which a separate actor (the control law) and critic (a 'memory') function are learned. A disadvantage of RL is that without having a process model it can be slow at learning and computationally expensive.

In this thesis, the goal was to design a theoretical framework using PBC techniques subject to control saturation, in which knowledge about the PH system can be incorporated and in which (optimal) control policies can be learned using actor-critic reinforcement learning. Therefore, Standard Actor-Critic (S-AC) RL has been combined with Energy-Balancing Passivity-Based Control (EB-PBC). This combination, called Energy-Balancing Actor-Critic (EBAC), showed its effectiveness in the pendulum swing-up problem, which was used as a benchmark test. The advantages of the EBAC method from a PBC perspective are that no PDE has to be explicitly solved, control saturation can be incorporated, the geometric structure of the PH system is preserved, local stability can be numerically demonstrated using passivity theory and the learned controllers can be interpreted in terms of energy shaping strategies. From a RL perspective, the learning is speeded up because model knowledge is available.

Table of Contents

List of Figures

List of Tables

Acknowledgements

During the eleven months trajectory that completing my MSc thesis has taken I have incurred many debts, most of which I will try to acknowledge here. First of all, I would like to thank Gabriel, my daily supervisor, for providing me with this interesting thesis topic, but most of all for our meetings, in which he always showed his faith in this project and in which he gave me useful suggestions concerning other relevant research areas, background information and in which he provided help on the math. Of course I would also like to thank Robert, my other supervisor, for his unlimited enthusiasm, for the ability to transfer that enthusiasm and for his critical opinion. However, the group that has certainly incurred the largest debt over the past months are my friends and family. I would like to thank them for their unlimited support, especially my mother, father and my girlfriend, for I often had the urge to tell them about 'finding the correct parameters', 'the difference between model-free and model-based learning' and many, many more interesting topics that I encountered in this project. Finally, I would like to thank my fellow students at DCSC for sharing thoughts on the MSc thesis, which was very helpful. My apologies for those that should be mentioned here but are omitted - it's not my intention. Also, there will probably still be some errors in this thesis, although I hope that the main message will still persuade new students to take an interest in the exciting world of learning control and nonlinear systems. Thank you all!

Delft, University of Technology — O.R. Sprangers
May 7, 2012

Chapter 1

Introduction

In recent years there has been a shift from approaching control design problems from a signal-processing point of view to a design methodology in which information about (the structure of) the system is exploited. A popular methodology to do so, first proposed in [39], is called *Passivity-Based Control (PBC)*, which exploits the property of passivity to achieve the control objective by rendering the closed-loop system passive with respect to a desired storage function and by injecting damping. The interested reader is referred to [38, 36, 41, 53] for variations on PBC and to [49, 12, 11] for examples of applications in robotics of PBC. In this thesis, the control of dynamical systems endowed with a particular structure, called *port-Hamiltonian (PH)* systems, will be investigated. It will be shown that these PH models have a very natural and intuitive way of representing a PBC problem.

On the other hand, today's expanding field of robotics is dealing with challenges such as unstructured or unknown environments and interaction with these environments (such as humans). Moreover, robots need to handle more complex tasks that cannot always be predefined, such that some sort of adaptation or learning control is required. A well-known and currently popular technique is called *Reinforcement Learning (RL)*, a machine learning technique that has a close similarity with how humans and animals learn and which has been used in many applications in robotics to this date, examples can be found in [50, 47, 35, 7, 24]. For more information about RL, the reader is referred to [50, 4, 10, 19, 25]. In this thesis, RL will be used as adaptive/learning method to combine the advantages of PBC and the PH framework with.

There are advantages to both fields in control:

- PBC techniques, especially of the form Control by Interconnection (more on that in chapter 2), used in combination with the PH system, have a very natural and intuitive way of representing a control design problem in terms of shaping the energy the dynamical system exchanges with its environment.

- Certain properties, such as (cyclo-)passivity, can be guaranteed by PBC techniques, which make them suitable for use in environments where multiple dynamical systems interact with each other.
- Reinforcement learning techniques are useful to obtain control policies when no or unstructured information about the environment and/or the to-be controlled (nonlinear) dynamical system is available.

However, there are also disadvantages:

- Deriving control laws in PBC for PH systems requires solving a Partial Differential Equation (PDE) for which it is hard to find nontrivial closed-loop solutions such that the PDE has to be calculated numerically, which is computationally expensive.
- Most of the PBC methods for PH systems in literature do not consider input saturation, which makes these methods less applicable to real-life environments where actuators commonly have a limited operating range.
- Enhancing the speed of learning and ensuring quality of control policies can be difficult in reinforcement learning, especially when dealing with high dimensional systems, where computer speeds can be a limiting factor and in practice convergence to a (local) optimum is not always guaranteed.

The problem is that as of today, there does not exist a method combining the advantages while relaxing/eliminating the disadvantages of both techniques.

1-1 Research Goals and Objectives

The general research goal of this thesis is to design a methodology that combines the major advantages of the two control paradigms mentioned above: incorporating knowledge about the system or its structure from the port-Hamiltonian framework, and learning (optimal) control policies via reinforcement learning.

The general goal is split into several objectives:

- The first objective of this thesis is to design a theoretical framework (methodology) using PBC techniques subject to control saturation that incorporates knowledge about a (possibly nonlinear) port-Hamiltonian system and that is able to learn (optimal) control policies (or closed-loop energy landscapes) via reinforcement learning. In this thesis, Actor-Critic (AC) RL will be used. To the author's best knowledge, no such method has yet been developed.
- The second objective is to test and verify the developed methodology using simulations and a real-life set-up. Therefore, the problem of swinging up an inverted

pendulum is studied, both in simulations and in experiments using a physical set-up. This is a low-dimensional but highly nonlinear control problem commonly used as a benchmark test in RL [20] but it has also been studied in PBC [2]. The developed methodology will be evaluated based on a set of criteria including the speed of learning (i.e. convergence speed), quality of the found policy (i.e. stability properties, robustness to parameter variations) and the set of stabilizable PH systems to which the developed methodology can be applied.

- The third objective is to compare the developed framework with current methods of choice, both in the field of PBC as well as in RL. More about the current methods of choice for both these fields can be found in Chapters 2-3.

1-2 Outline of Thesis

To make this thesis self-contained, a theoretical introduction to the two major control fields considered in this thesis is given in Chapters 2-3. Then, in Chapter 4, the developed method, called Energy-Balancing Actor-Critic (EBAC), is introduced in the form of a paper that will be submitted to *Automatica*[1]. After that, in Chapter 5 an extensive review of simulations and experiments using the proposed method is given to gain further insights and understand the behaviour of the EBAC method. Finally, based on the results of these last two chapters, conclusions and recommendations will be given in Chapter 6.

[1] *Automatica* is a journal of the International Federation of Automatic Control (IFAC), © 2012 Elsevier.

Chapter 2

Port-Hamiltonian Systems

In this chapter, the port-Hamiltonian (PH) framework is introduced as well as methods to control port-Hamiltonian systems. The port-Hamiltonian framework allows to describe a physical system in terms of its energy exchange and of ports modeling the interaction between the basic elements of the system and its environment. As such, it allows for a broader class of dynamical systems to which PBC can be applied to. First, necessary definitions, the general framework and some important properties are given. Then, methods to control the port-Hamiltonian system are discussed after which a choice for a particular control type for this thesis is made.

2-1 Dissipative and Passive Systems

In this section, the most important definitions that will come back throughout the thesis are stated. Given the time-invariant state space system:

$$\Sigma : \begin{cases} \dot{x} &= f(x,u) \ , \quad u \in U \\ y &= h(x,u) \ , \quad y \in Y \end{cases} \tag{2-1}$$

where $U \in \mathbb{R}^m$ and $Y \in \mathbb{R}^p$, respectively, and $\mathcal{X} \in \mathbb{R}^n$ the state space manifold containing the local coordinates $x = (x_1, x_2, \ldots, x_n)$.

Definition 2-1.1 (Dissipative systems)**.** *A state space system Σ is said to be* dissipative *with respect to a supply rate s defined as:*

$$s : \ U \times Y \to \mathbb{R} \qquad \textit{where} \qquad \mathbb{R}^+ = [0, \infty) \tag{2-2}$$

if there exists a function $S : \ \mathcal{X} \to \mathbb{R}^+$, called the storage *function, such that for all $x_0 \in \mathcal{X}$, all $t_1 \geq t_0$ and all input functions u:*

$$S(x(t_1)) \leq S(x(t_0)) + \int_{t_0}^{t_1} s(u(t), y(t)) dt \tag{2-3}$$

The expression (2-3) states that the stored energy $S(x(t_1))$ is at most equal to the stored energy $S(x(t_0))$ at time t_0 plus the total supplied energy (given by the integral in (2-3)) during the interval $[t_0, t_1]$. Thus, internal creation of energy is not possible, only internal *dissipation*. [53, 59]

Using the notion of dissipative systems it is possible to define a passive system.

Definition 2-1.2 (Passive systems [53])**.** *A state space system Σ with $U = Y = \mathbb{R}^m$ is called passive if it is dissipative with respect to the supply rate $s(u, y) = u^T y$. Σ is* strictly input passive *if there exists $\delta > 0$ such that Σ is dissipative with respect to $s(u, y) = u^T y - \delta\|u\|^2$. Σ is* strictly output passive *if there exists $\epsilon > 0$ such that Σ is dissipative with respect to $s(u, y) = u^T y - \epsilon\|y\|^2$. Σ is* conservative *if it is lossless with respect to $s(u, y) = u^T y$.*

2-2 Port-Hamiltonian Systems

The general framework of PH systems was introduced in [33] and was formalized in [54, 53]. A review of the application of PH systems to PBC can be found in [44, 43, 53, 42]. The general input-state-output port-Hamiltonian system reads:

$$\Sigma : \begin{cases} \dot{x} = [J(x) - R(x)] \dfrac{\partial H(x)}{\partial x} + g(x)u \\ y = g^T(x) \dfrac{\partial H(x)}{\partial x} \end{cases} \tag{2-4}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$, $m \leq n$ is the control input, $J(x)$, $R(x) : \mathbb{R}^n \mapsto \mathbb{R}^{n \times n}$ with $J(x) = -J(x)^T$ and $R(x) = R(x)^T \geq 0$ are the interconnection and damping matrix, respectively, $H(x) : \mathbb{R}^n \mapsto \mathbb{R}$ the Hamiltonian which is the stored energy in the system, $u, y \in \mathbb{R}^m$ are conjugated variables whose product has units of power and $g(x) : \mathbb{R}^n \mapsto \mathbb{R}^{n \times m}$ is the input matrix assumed to be full rank. For the remainder of this thesis, denote the matrix $F(x) : \mathbb{R}^n \mapsto \mathbb{R}^{n \times n}$,

$$F(x) := J(x) - R(x) \tag{2-5}$$

which satisfies $F(x) + F^T(x) = -2R(x) \leq 0$. The system (2-4) satisfies the *power-balance* equation:

$$\begin{aligned} \dot{H}(x) &= \frac{\partial^T H(x)}{\partial x} \dot{x} && (2\text{-}6) \\ &= \frac{\partial^T H(x)}{\partial x} \left([J(x) - R(x)] \frac{\partial H(x)}{\partial x} + g(x)u \right) && (2\text{-}7) \\ &= -\frac{\partial^T H(x)}{\partial x} R \frac{\partial H(x)}{\partial x} + u^T y && (2\text{-}8) \\ &\leq u^T y \end{aligned}$$

which is referred to as the *cyclo-passivity* inequality [41] if the Hamiltonian is not bounded from below nor is it positive-definite and the *passivity* inequality if it is

bounded from below and positive-semidefinite. Intuitively, cyclo-passivity means that the system cannot create energy over closed paths in the state space. Hence, every passive system is cyclo-passive.
There are some key advantages of working with PH systems for PBC: first, the choice of storage function becomes more natural and second, these models allow for a clear distinction between the structural elements in the system, such as the interconnection, the storage elements and dissipation elements. Also, a large class of dynamical systems can be written in PH form, which makes it a suitable representation of complex dynamical systems.

An important dynamical property of PH systems is the existence of dynamical invariants independent of the Hamiltonian $H(x)$ of the system, called *Casimir* functions [53, 32]. For dynamical invariants the following set of partial differential equations is considered:

$$\frac{\partial^T C(x)}{\partial x} J(x) = 0, \quad x \in \mathcal{X} \tag{2-9}$$

in the unknown function $C : \mathcal{X} \to \mathbb{R}$. If (2-9) has a solution C, then the time derivative along the PH system satisfies:

$$\begin{aligned} \frac{dC}{dt} &= \frac{\partial^T C(x)}{\partial x} J(x) \frac{\partial H(x)}{\partial x} + \frac{\partial^T C(x)}{\partial x} g(x) u \\ &= \frac{\partial^T C(x)}{\partial x} g(x) u \end{aligned} \tag{2-10}$$

For the input $u = 0$ or for arbitrary inputs if also $\frac{\partial^T C(x)}{\partial x} g(x) = 0$ the function $C(x)$ remains constant along the trajectories of the PH system. A function $C(x)$ satisfying (2-9) is called a *Casimir* function. For PH systems with dissipation, functions $C : \mathcal{X} \to \mathbb{R}$ satisfying the set of equations:

$$\frac{\partial^T C(x)}{\partial x} [J(x) - R(x)] = 0, \quad x \in \mathcal{X} \tag{2-11}$$

are considered, which can be shown is equal to:

$$\begin{aligned} \frac{\partial^T C(x)}{\partial x} J(x) &= 0 \\ \frac{\partial^T C(x)}{\partial x} R(x) &= 0 \end{aligned} \tag{2-12}$$

A $C(x)$ satisfying (2-11) is a Casimir function for both geometric structures defined by $J(x)$ and $R(x)$.
An important consequence of the existence of Casimir functions is that if $C_1(x)$, $C_2(x)$, $\ldots$, $C_r(x)$ are Casimir functions, then not only $\frac{dH(x)}{dt} = 0$ for $u = 0$, but also:

$$\frac{d}{dt}(H + H_{\mathrm{a}}(C_1, C_2, \ldots, C_r))(x(t)) = 0 \tag{2-13}$$

for any function $H_{\mathrm{a}}(x) : \mathbb{R}^r \to \mathbb{R}$. This means that even though $H(x)$ is not positive definite at an equilibrium x^*, the function $H(x) + H_{\mathrm{a}}(C_1, C_2, \ldots, C_r)(x)$ is possibly

positive definite at the equilibrium point by appropriately choosing $H_{\mathrm{a}}(x)$ and thus may serve as a candidate Lyapunov function for stability analysis. This method is called the *Energy-Casimir* method and it has various applications in the control of port-Hamiltonian systems.

2-3 Passivity-Based Control of Port-Hamiltonian Systems

The general goal in PBC of PH systems is to reach the control objective by rendering the closed-loop system passive with respect to a desired storage function. One of the key features of PH systems is that by using a power-conserving interconnection between two PH systems, the resulting system is also a PH system. This approach suggests energy-transfer or energy-balancing strategies, where energy is transferred from one part of the (total) system to the other. This leads to a different approach for control problems by thinking in terms of shaping the behaviour of the system at the interaction port by, for example, addition of a controller that is also in the form of a PH system.

The control of PH systems can be divided into 4 types: Control by Damping Injection (CbDI), Control by Interconnection (CbI), Interconnection and Damping Assignment Passivity-Based Control (IDA-PBC) and Standard Passivity-Based Control (SPBC); each subsequent control type applicable to a larger set of stabilizable PH plants than the previous. The difference between each type is which target closed-loop system is desired. In this thesis, CbDI, Energy-Balancing Passivity-Based Control (EB-PBC) (which is a form of CbI) and IDA-PBC will be used to achieve the research objective, the reasons for which will be explained further on. First, the SPBC problem is explained such that later on, other PBC types can be viewed as a special case of SPBC.

2-3-1 Standard Passivity-Based Control

If the open-loop PH system satisfies the power-balance equation (2-8) then the control action $u = \beta(x) + v$ is said to solve the *Standard Passivity-Based Control* (SPBC) problem [42, 41] if the closed-loop system satisfies the desired power-balance equation:

$$\dot{H}_{\mathrm{d}}(x) = v^T z - d_{\mathrm{d}}(x) \tag{2-14}$$

with $H_{\mathrm{d}} : \mathbb{R}^n \to \mathbb{R}^+$ the desired energy function, $d_{\mathrm{d}}(x) : \mathbb{R}^n \to \mathbb{R}^+$ the desired damping and $z \in \mathbb{R}^m$ a new passive output. An added energy function is defined as:

$$H_{\mathrm{a}}(x) := H_{\mathrm{d}}(x) - H(x) \tag{2-15}$$

Now, a state feedback is said to be *energy-balancing* if the added energy $H_{\mathrm{a}}(x)$ is equal to the energy supplied to the system by the environment,

$$\dot{H}_{\mathrm{a}} = -\beta(x)^T y \tag{2-16}$$

which implies that the desired energy function $H_{\mathrm{d}}(x)$ in that case is the difference between the stored and supplied energy.

The PH system (2-4) in closed-loop with $u = \beta(x) + v$ satisfies (2-14) iff

$$\frac{\partial^T H_{\mathrm{d}}(x)}{\partial x}\left([J(x) - R(x)]\frac{\partial H(x)}{\partial x} + g(x)\beta(x)\right) = -d_{\mathrm{d}}(x) \tag{2-17}$$

$$z(x) = g^T(x)\frac{\partial H_{\mathrm{d}}(x)}{\partial x} \tag{2-18}$$

for some function $d_{\mathrm{d}}(x)$.

2-3-2 Control by Damping Injection

In CbDI closed-loop asymptotic stability[1] is achieved by injecting damping on the passive output y via:

$$u = -K(x)y \tag{2-19}$$

with $K(x) > 0, K \in \mathbb{R}^m$ a damping matrix on the passive output such that the open-loop to closed-loop target dynamics can be characterized as follows:

$$\begin{aligned} \dot{x} &= [J(x) - R(x)]\frac{\partial H(x)}{\partial x} + g(x)u \\ &\Downarrow \\ \dot{x} &= [J(x) - R_{\mathrm{d}}(x)]\frac{\partial H(x)}{\partial x} \end{aligned} \tag{2-20}$$

with $R_{\mathrm{d}}(x) = R(x) + g(x)K(x)g^T(x)$ the desired damping matrix for the closed-loop system. However, the result of stabilization by damping injection is only applicable if the desired equilibrium is a minimum of the storage function or Hamiltonian $H(x)$.

2-3-3 Control by Interconnection

In light of the results of CbDI, an extra degree of freedom in the controller design can be created by *shaping* the closed-loop Hamiltonian, and there are several approaches to do so [42]. Motivated by the fact that interconnections of PH systems again yield PH systems, a controller is defined as the PH system:

$$\Sigma_{\mathrm{C}} : \begin{cases} \dot{\zeta} = [J_{\mathrm{C}}(\zeta) - R_{\mathrm{C}}(\zeta)]\dfrac{\partial H_{\mathrm{C}}(\zeta)}{\partial \zeta} + g_{\mathrm{C}}(\zeta)u_{\mathrm{C}}\ ,\ x \in \mathcal{X}_{\mathrm{C}},\ u_{\mathrm{C}} \in \mathbb{R}^m \\ y_{\mathrm{C}} = g_{\mathrm{C}}^T(\zeta)\dfrac{\partial H_{\mathrm{C}}(\zeta)}{\partial \zeta}\ ,\ y_{\mathrm{C}} \in \mathbb{R}^m \end{cases} \tag{2-21}$$

with $J_{\mathrm{C}}(\zeta) = -J^T(\zeta), R_{\mathrm{C}}(\zeta) = R_{\mathrm{C}}^T(\zeta) \geq 0$ and $H_{\mathrm{C}}(\zeta)$ the controller Hamiltonian. With the power-conserving interconnection:

$$\Sigma_I : \left\{ \begin{bmatrix} u \\ u_C \end{bmatrix} = \begin{bmatrix} 0 & -I_m \\ I_m & 0 \end{bmatrix} \begin{bmatrix} y \\ y_{\mathrm{C}} \end{bmatrix} + \begin{bmatrix} v \\ v_{\mathrm{C}} \end{bmatrix} \right. \tag{2-22}$$

[1] It should be noted that the system should also be zero-state detectable.

with v, v_{C} external inputs, the following closed-loop system is obtained:

$$\begin{aligned}\begin{bmatrix}\dot{x}\\ \dot{\zeta}\end{bmatrix} &= \begin{bmatrix}J(x)-R(x) & -g(x)g_{\mathrm{C}}^T(\zeta)\\ g_{\mathrm{C}}(\zeta)g^T(x) & J_{\mathrm{C}}(\zeta)-R_{\mathrm{C}}(\zeta)\end{bmatrix}\begin{bmatrix}\frac{\partial H(x)}{\partial x}\\ \frac{\partial H_{\mathrm{C}}(\zeta)}{\partial \zeta}\end{bmatrix}+\begin{bmatrix}g(x) & 0\\ 0 & g_{\mathrm{C}}(\zeta)\end{bmatrix}\begin{bmatrix}v\\ v_{\mathrm{C}}\end{bmatrix}\\ \begin{bmatrix}y\\ y_{\mathrm{C}}\end{bmatrix} &= \begin{bmatrix}g(x) & 0\\ 0 & g_{\mathrm{C}}(\zeta)\end{bmatrix}\begin{bmatrix}\frac{\partial H(x)}{\partial x}\\ \frac{\partial H_{\mathrm{C}}(\zeta)}{\partial \zeta}\end{bmatrix}\end{aligned} \tag{2-23}$$

which is still a port-Hamiltonian system with Hamiltonian $H(x)+H_{\mathrm{C}}(\zeta)$. By employing the Energy-Casimir method, energy-based Lyapunov functions of the following form are searched for:

$$L(x,\zeta) = H(x) + H_{\mathrm{C}}(\zeta) + C(x,\zeta) \tag{2-24}$$

Now, the idea, established in [44], is to look for Casimir functions of the form:

$$C(x,\zeta) = G(x) - \zeta + \kappa \tag{2-25}$$

with κ a level constant. Equation (2-25) implies looking for solutions of the set of PDE's (2-11):

$$\begin{bmatrix}\frac{\partial^T G(x)}{\partial x} & -I_m\end{bmatrix}\begin{bmatrix}J(x)-R(x) & -g(x)g_{\mathrm{C}}^T(\zeta)\\ g_{\mathrm{C}}(\zeta)g^T(x) & J_{\mathrm{C}}(\zeta)-R_{\mathrm{C}}(\zeta)\end{bmatrix} = 0 \tag{2-26}$$

Now, (2-25) satisfies (2-26) if the following set of equalities is satisfied [44]:

$$\frac{\partial^T G(x)}{\partial x} J(x) \frac{\partial G(x)}{\partial x} = J_{\mathrm{C}}(\zeta) \tag{2-27}$$

$$R(x)\frac{\partial G(x)}{\partial x} = 0 \tag{2-28}$$

$$R_{\mathrm{C}}(\zeta) = 0 \tag{2-29}$$

$$\frac{\partial^T G(x)}{\partial x} J(x) = g_{\mathrm{C}}(\zeta) g^T(x) \tag{2-30}$$

In this case, the closed-loop dynamics are reduced to the set:

$$\Omega = \{(x,\zeta) | \zeta = G(x) + \kappa\} \tag{2-31}$$

which means that the closed-loop dynamics are restricted to a subspace of the extended state space[2] (x,ζ). Combining now the first equation of (2-21) with (2-28)-(2-29) results in the x-dynamics (with $v, v_{\mathrm{C}} = 0$):

$$\dot{x} = [J(x) - R(x)]\left(\frac{\partial H(x)}{\partial x} + \frac{\partial G(x)}{\partial x}\frac{\partial H_{\mathrm{C}}(\zeta)}{\partial \zeta}\right) \tag{2-32}$$

$$= [J(x) - R(x)]\frac{\partial H_{\mathrm{s}}(x)}{\partial x} \tag{2-33}$$

[2]It should be noted that this result applies to systems in which *all* the controller state variables ζ are related to the system state variables x. Otherwise, a more generalized result holds. (which is quite similar)

with $H_s(x) = H(x) + H_C(G(x) + \kappa)$ the *shaped* Hamiltonian of the system. Equation (2-33) has a clear interpretation in terms of Energy-Balancing Passivity-Based Control[3]:

$$\frac{dH_s(x)}{dt} = \frac{dH(x)}{dt} + \frac{dH_C(x)}{dt} = \frac{dH(x)}{dt} - u^T y \tag{2-34}$$

which means that the shaped Hamiltonian $H_s(x)$ is equal to the difference between the (original) plant Hamiltonian $H(x)$ and the supplied energy by the control system (2-21). Furthermore, the shaped Hamiltonian is given by:

$$H_s(x) = H(x) + H_C(G(x) + \kappa) \tag{2-35}$$

which means that the described method generates Casimir functions by appropriately choosing $H_C(x)$. In [41], the following controller[4] Hamiltonian system is chosen:

$$\Sigma_C : \begin{cases} \dot{\zeta} = u_C \\ y_C = \dfrac{\partial H_C(\zeta)}{\partial \zeta} \end{cases} \tag{2-36}$$

and using the power-preserving interconnection (2-22) with $v_C = 0$ the set of equalities that need to be satisfied for the existence of Casimir functions (2-26) reduce to:

$$\frac{\partial^T G(x)}{\partial x}[J(x) - R(x)] = g^T(x) \tag{2-37}$$

$$\frac{\partial^T G(x)}{\partial x} g(x) = 0 \tag{2-38}$$

Then, (2-24) is such that $\dot{L}(x, \zeta) \leq v^T y$, which implies cyclo-passivity. This form (i.e. choosing (2-36) as controller PH system) of Control by Interconnection will be abbreviated with CbI. It has been shown in [41] that CbI can be extended to increase the set of stabilizable plants and that common types of Passivity-Based Control for PH systems, such as EB-PBC [42, 41], Power-Shaping Passivity-Based Control (PS-PBC) [37, 16, 15] and Basic Interconnection and Damping Assignment Passivity-Based Control [40] are forms of CbI.

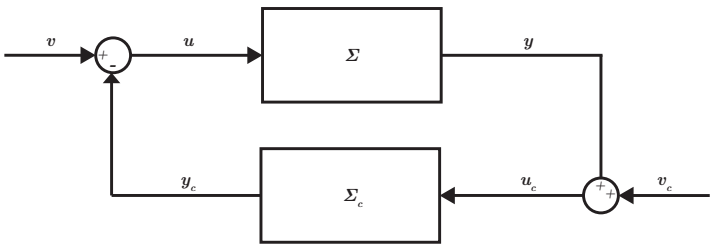


Figure 2-1: Control by Interconnection of a port-Hamiltonian system Σ by interconnection with a controller Hamiltonian system Σ_C.

[3] See also Section 2-3-3.

[4] This is actually (nonlinear) integral control [23].

Energy-Balancing Passivity-Based Control

The existence of Casimir functions can be relaxed using *control as a state-modulated (SM) source* [42, 45, 41]. The nonlinear integral controller of (2-36) is again considered as the controller Σ_{C} which the system will be interconnected with. Furthermore, the unitary feedback is now replaced by a static state dependent feedback $u = \alpha(x)$ such that:

$$\Sigma_I^{SM} : \left\{ \begin{bmatrix} u \\ u_{\mathrm{C}} \end{bmatrix} = \begin{bmatrix} 0 & -\alpha(x) \\ \alpha^T(x) & 0 \end{bmatrix} \begin{bmatrix} y \\ y_{\mathrm{C}} \end{bmatrix} + \begin{bmatrix} v \\ 0 \end{bmatrix} \right. \tag{2-39}$$

The closed-loop system with the new feedback interconnection and the controller Σ_{C} yields:

$$\begin{bmatrix} \dot{x} \\ \dot{\zeta} \end{bmatrix} = \begin{bmatrix} J(x) - R(x) & -g(x)\alpha(x) \\ \alpha^T(x)g^T(x) & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial H(x)}{\partial x} \\ \frac{\partial H_{\mathrm{C}}(\zeta)}{\partial \zeta} \end{bmatrix} + \begin{bmatrix} g(x) \\ 0 \end{bmatrix} v \tag{2-40}$$

such that the set of equalities that need to be satisfied for the existence of Casimir functions (2-26) reduce to:

$$\frac{\partial^T G(x)}{\partial x}[J(x) - R(x)] = \alpha^T(x)g^T(x) \tag{2-41}$$

$$\frac{\partial^T G(x)}{\partial x} g(x) = 0 \tag{2-42}$$

In [41], $\alpha(x) : \mathbb{R}^n \to \mathbb{R}^{m \times m}$ is defined as:

$$\alpha(x) = -(g^T(x)g(x))^{-1} g^T(x)[J(x) - R(x)]^T \frac{\partial G(x)}{\partial x} \tag{2-43}$$

such that (2-41)-(2-42) reduce to:

$$g^{\perp}(x)[J(x) - R(x)]^T \frac{\partial G(x)}{\partial x} = 0 \tag{2-44}$$

$$\frac{\partial^T G(x)}{\partial x} g(x) = 0 \tag{2-45}$$

with $g^{\perp}(x)$ the full-rank left annihilator of $g(x)$, i.e. $g^{\perp}(x)g(x) = 0$. With this choice the cyclo-passivity inequality (2-8) with storage function (2-24) is satisfied. This form of CbI is also called Energy-Balancing Passivity-Based Control (EB-PBC). Usually, EB-PBC is derived from a SPBC point of view. If, in (2-17),

$$d_{\mathrm{d}}(x) = d = -\frac{\partial^T H(x)}{\partial x}[J(x) - R(x)]\frac{\partial H(x)}{\partial x} \tag{2-46}$$

the control law:

$$\alpha(x) = -(g^T(x)g(x))^{-1} g^T(x)[J(x) - R(x)]\frac{\partial H_{\mathrm{a}}(x)}{\partial x} \tag{2-47}$$

solves the SPBC problem with $H_{\mathrm{a}}(x)$ a solution of:

$$g^{\perp}(x)[J(x)-R(x)]\frac{\partial H_{\mathrm{a}}(x)}{\partial x}=0 \tag{2-48}$$

$$g^{T}(x)\frac{\partial H_{\mathrm{a}}(x)}{\partial x}=0 \tag{2-49}$$

and the controller is energy-balancing as in (2-16)[5]. Hence, EB-PBC is equivalent to the state-modulated CbI for $\frac{\partial G(x)}{\partial x}=\frac{\partial H_{\mathrm{a}}(x)}{\partial x}$ (compare (2-48)-(2-49) with (2-44)-(2-45)). The target dynamics obtained in EB-PBC are:

$$\begin{aligned}\dot{x}&=[J(x)-R(x)]\frac{\partial H(x)}{\partial x}+g(x)u\\ &\Downarrow\\ \dot{x}&=[J(x)-R(x)]\frac{\partial H_{\mathrm{d}}(x)}{\partial x}\end{aligned} \tag{2-50}$$

Usually, damping is then injected according to (2-19) to asymptotically stabilize the closed-loop system at the desired equilibrium. It is now possible to constitute the main reasons for choosing EB-PBC as one of the Passivity-Based Control (PBC) control techniques for this thesis.

- In EB-PBC, the resulting control law (2-47) can be interpreted in terms of energy-balancing, which means that the added energy is equal to the energy supplied to the system by the environment. Hence, in the closed-loop system the energy is *shaped*, which means that by applying EB-PBC, it is possible to generate a closed-loop energy landscape. The advantage is that this type of control has a physical interpretation, e.g. in mechanical systems the change in energy will be the power (force times velocity) supplied to the system.
- From a CbI perspective, the controller can be viewed and formulated in terms of interconnecting subsystems in PH form with a controller PH system. The advantage is that it is possible to interpret the control in terms of 'building blocks' of PH systems interconnected in a power-preserving way with each other yielding another PH system. This modular approach allows breaking down complex control problems by interconnecting simpler subsystems.

Dissipation Obstacle

A disadvantage of EB-PBC however, is that it suffers from the *dissipation obstacle*. From equation (2-28) it is also possible to write:

$$R(x)\frac{\partial H_{\mathrm{C}}(G(x))}{\partial x}=0 \tag{2-51}$$

[5]The proof can be constructed by filling in (2-47) in the general to-be solved PDE (2-17).

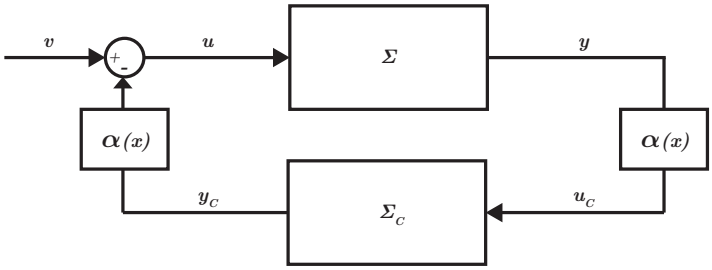


Figure 2-2: Control by state-modulated interconnection such that the existence of Casimir functions is relaxed.

which means that Casimir functions satisfying (2-28) are independent of those coordinates affected by damping in the system. In other words, $H_{\mathrm{C}}(x)$ can only depend on those coordinates where there is no natural damping and thus energy can only be shaped for those coordinates. This is known as the *dissipation obstacle* [42]. In mechanical systems, this is not an issue because damping is associated with velocities and generally, the control objective is to regulate the position (i.e. shape the potential energy), which causes the power to be driven to zero (and thus extracting a finite amount of energy of the controller). However, in for example RLC circuits it is possible that the power, which is the product of current and voltage, is nonzero in the equilibrium. In that case, an infinite amount of energy from the controller is necessary, which means that there is infinite dissipation. Hence, the dissipation obstacle decreases the set of stabilizable plants when applying EB-PBC[6].

2-3-4 Interconnection and Damping Assignment Passivity-Based Control

To overcome the dissipation obstacle and increase the set of stabilizable plants various techniques have been proposed, such as Power-Shaping Passivity-Based Control (PS-PBC) and Interconnection and Damping Assignment Passivity-Based Control (IDA-PBC). In these techniques, the target dynamics are of the form

$$\begin{aligned} \dot{x} &= F(x)\frac{\partial H(x)}{\partial x} + g(x)u \\ &\Downarrow \\ \dot{x} &= F_{\mathrm{d}}(x)\frac{\partial H_{\mathrm{d}}(x)}{\partial x} \end{aligned} \tag{2-52}$$

with $F_{\mathrm{d}}(x) = [J_{\mathrm{d}}(x) - R_{\mathrm{d}}(x)]$ containing the desired interconnection and damping matrices satisfying

$$F_{\mathrm{d}}(x) + F_{\mathrm{d}}(x)^T \leq 0 \tag{2-53}$$

Thus, not only the energy is shaped, but also the interconnection and damping matrices are changed. In this thesis, IDA-PBC[7] [45] is considered, which is the most general form

[6] Equations (2-45) and (2-49) imply that the dissipation obstacle is present in EB-PBC.

[7] The definition of IDA-PBC provided here is not exactly the definition by [45], because Ortega et al. consider a broader class of systems with control also acting on the interconnection structure (i.e. $J(x,u)$) and

of PBC techniques that change the interconnection and damping matrices. Define the desired interconnection and damping matrices $J_{\mathrm{d}}(x) = -J_{\mathrm{d}}^T(x)$ and $R_{\mathrm{d}}(x) = R_{\mathrm{d}}^T(x) \geq 0$. Now, assume that functions $\beta(x), J_{\mathrm{a}}(x), R_{\mathrm{a}}(x)$ and a vector function $D(x)$ can be found such that:

$$[J(x) + J_{\mathrm{a}}(x) - (R(x) + R_{\mathrm{a}}(x))]D(x) = -[J_{\mathrm{a}}(x) - R_{\mathrm{a}}(x)]\frac{\partial H(x)}{\partial x} + g(x)\beta(x) \quad (2\text{-}54)$$

and satisfying the following properties:

1. Structure preservation:

$$J_{\mathrm{d}}(x) := J(x) + J_{\mathrm{a}}(x) = -[J(x) + J_{\mathrm{a}}(x)]^T \quad (2\text{-}55)$$
$$R_{\mathrm{d}}(x) := R(x) + R_{\mathrm{a}}(x) = [R(x) + R_{\mathrm{a}}(x)]^T \geq 0 \quad (2\text{-}56)$$

2. Integrability: $D(x)$ is the gradient of a scalar function:

$$\frac{\partial D(x)}{\partial x} = \left[\frac{\partial D(x)}{\partial x}\right]^T \quad (2\text{-}57)$$

3. Equilibrium assignment: $D(x)$ at x^* satisfies:

$$D(x^*) = -\frac{\partial H(x^*)}{\partial x} \quad (2\text{-}58)$$

4. Lyapunov stability: The Jacobian of $D(x)$ at x^* satisfies the bound:

$$\frac{\partial D(x)}{\partial x^*} > -\frac{\partial^2 H(x^*)}{\partial x^2} \quad (2\text{-}59)$$

Then, the closed-loop system $u = \beta(x)$ is a PH system with:

$$H_{\mathrm{d}}(x) := H(x) + H_{\mathrm{a}}(x) \quad (2\text{-}60)$$
$$\frac{\partial H_{\mathrm{a}}(x)}{\partial x} = D(x) \quad (2\text{-}61)$$

and x^* will be a locally stable equilibrium of the closed-loop system (via LaSalle's invariance principle asymptotic stability can be proven). It can be concluded that the solvability of the IDA-PBC problem comes down to solving (2-54), which can equivalently be rewritten as:

$$g^{\perp}(x)\left([J_{\mathrm{d}}(x) - R_{\mathrm{d}}(x)]\frac{\partial H_{\mathrm{a}}(x)}{\partial x}\right) = -g^{\perp}(x)\left([J_{\mathrm{a}}(x) - R_{\mathrm{a}}(x)]\frac{\partial H(x)}{\partial x}\right) \quad (2\text{-}62)$$

with $g^{\perp}(x)$ the left annihilator of $g(x)$, i.e. $g^{\perp}(x)g(x) = 0$. Equation (2-62) is called the *matching equation* of IDA-PBC [36]. To solve (2-62), there are three different methods [36]:

constant source inputs ($g(x, u)$). This leads to almost the same results, but for the sake of consistency here IDA-PBC is explained with respect to the PH system given in (2-4).

1. Non-parameterized IDA-PBC [45]: fix $J_{\mathrm{d}}(x)$ and $R_{\mathrm{d}}(x)$ and compute $H_{\mathrm{d}}(x)$. Then, the control law can be explicitly computed:

$$\begin{aligned}\beta(x) =&[g^T(x)g(x)]^{-1}g^T(x)\Big\{[J_{\mathrm{d}}(x)+R_{\mathrm{d}}(x)]\frac{\partial H_{\mathrm{d}}(x)}{\partial x}\\ &-[J(x)+R(x)]\frac{\partial H(x)}{\partial x}\Big\}\end{aligned} \tag{2-63}$$

2. Algebraic IDA-PBC: fix the desired Hamiltonian $H_{\mathrm{d}}(x)$ and compute $R_{\mathrm{d}}(x)$, $J_{\mathrm{d}}(x)$ and $g^{\perp}(x)$.

3. Parameterized IDA-PBC: Take advantage of the structure of the PDE (2-62), such as in [40, 45, 1, 17].

IDA-PBC solves the SPBC problem if, in (2-14),

$$d_{\mathrm{d}}(x) = -\frac{\partial^T H_{\mathrm{d}}(x)}{\partial x}F_{\mathrm{d}}(x)\frac{\partial H_{\mathrm{d}}(x)}{\partial x} \tag{2-64}$$

and the control law $\beta(x)$ as in (2-63). It is now possible to constitute the main reasons for choosing IDA-PBC as one of the PBC control techniques for this thesis.

- IDA-PBC does not suffer from the dissipation obstacle, and hence allows for a larger set of plants to be stabilized.
- IDA-PBC has complete freedom in choosing $F_{\mathrm{d}}(x)$ (the only constraint is (2-53)), which makes it more general than e.g. PS-PBC, where $F_{\mathrm{d}}(x)$ is the result of solving a particular PDE [41].
- It has been widely applied to mechanical and electromechanical systems, and several extensions have been made to even further increase its applicability [1, 31, 34, 9].

2-4 Control Saturation

Control saturation is a very important aspect of control design, since most physical applications have limited actuators, such that not every calculated control action can be sent to the system. Therefore, it is important to incorporate control saturation. However, there has been little account of incorporating control saturation in PBC for PH systems. Escobar et al. [13] incorporate control saturation in PBC in an Euler-Lagrange setting, which had been previously studied by Loria et al. in [30]. A more extensive discussion on saturation for Euler-Lagrange systems in PBC is given in [38]. However, these methods do not incorporate any form of adaptation or learning. Furthermore, the later developed PH system is a framework that can be applied to a

broader class of systems than those described in [13, 30, 38]. In this thesis, the following general saturation function will be considered:

$$u_{\text{sat}}(x) = \varsigma(u(x)) \tag{2-65}$$

with $\varsigma : \mathbb{R}^m \mapsto S,\ S \subset \mathbb{R}^m$ a saturation function such that:

$$\varsigma(u(x)) \in S\ \forall u \tag{2-66}$$

Examples of functions ς are the hyperbolic tangent function (tanh) and the arctangent function (arctan).

2-5 Conclusion

The PH framework is an intuitive way of representing dynamical systems in terms of energy exchange and ports modeling the interaction between the basic elements of a system. PBC of PH systems is based on stabilization by passivation, and the two PBC techniques that will be used in this thesis are EB-PBC and IDA-PBC. EB-PBC is useful because of its physical interpretation, in terms of energy shaping and in terms of Control by Interconnection. It suffers from the dissipation obstacle however, which is why IDA-PBC is also considered as a more general form of PBC. Control saturation is generally not incorporated in PBC for PH systems. Only for Euler-Lagrange systems, which are a subclass of systems of the form (2-4), research has been done on control design incorporating control saturation. However, this research often does not include a form of adaptivity or learning.

Finally, EB-PBC and IDA-PBC can be summarized by their open-loop to closed-loop target dynamics (2-50) resp. (2-52) and the PDE that has to be solved:

- EB-PBC: Solve (2-48)-(2-49):

$$\begin{bmatrix} g^{\perp}(x)F(x) \\ g^T(x) \end{bmatrix} \frac{\partial H_{\text{a}}(x)}{\partial x} = 0 \tag{2-67}$$

- IDA-PBC: Solve (2-62):

$$g^{\perp}(x)F_{\text{d}}(x)\frac{\partial H_{\text{a}}(x)}{\partial x} = -g^{\perp}(x)(F_{\text{d}}(x) - F(x))\frac{\partial H(x)}{\partial x} \tag{2-68}$$

Hence, the challenge in this thesis is to be able to find a solution to these Partial Differential Equations (PDE's) using learning techniques while control saturation is present without explicitly having to solve a PDE, which is generally difficult. Therefore, Reinforcement Learning will be used, which is the topic of the next chapter.

Chapter 3

Reinforcement Learning

In this chapter, the RL problem is explained and Actor-Critic (AC) methods are introduced. This class of Reinforcement Learning (RL) methods will be used to learn the PBC control laws without having to explicitly solve a PDE.

3-1 Introduction

In the reinforcement learning problem [50], the goal is that an agent learns and takes *actions* (i.e. control strategy) in some state so as to maximize some numerical *reward*, while interacting with the *environment*. The agent adapts its behaviour based on the reward it receives from the environment. The rewards received due to interaction with the environment are stored in a *value* function that determines for the agent how 'good' it is to be in a given state. The actions to be taken by the agent are based on a function called the *policy*.

3-2 The Reinforcement Learning Framework

The general framework will be described in discrete-time, because most RL methods deal with discrete-time[1]. The following ingredients constitute an RL-problem [50, 4]:

- **Agent**: The agent is the learning controller and decision maker in the RL-framework. The key idea is that the agent learns the optimal mapping from states to actions via the policy.

[1]For continuous-time RL the interested reader is referred to [10] for an extensive introduction to the topic and for recent work in this field the reader is referred to [56, 6, 52].

- **Environment**: Everything outside the agent. If the state of the environment has a one-to-one relation with the observed state by the agent, it is called fully observable. Otherwise, it is called partially observable. The state of the environment gives rise to the reward the agent receives.
- **States**: A state $x_k \in \mathcal{X}$ is the output of the environment as observed by the agent in time instant k. Based on the observation of the state, the agent takes an action.
- **Actions**: An action $u_k \in \mathcal{U}(x_k)$ that the agent takes with $\mathcal{U}(x_k)$ the set of actions corresponding to the state x_k.
- **Reward**: The reward is the numerical return on how good or how bad the action taken in a particular state is as returned by the environment.
- **Policy**: The policy is the optimal mapping from states to actions ($x_k \mapsto u_k$) that the agent tries to find.

3-2-1 Markov Decision Process

In RL, the environment is described by the mapping $x_{k+1} = f(x_k, u_k)$, which means that the environment depends on the previous state and action. In general, the environment at time instant $k+1$ depends on everything that has happened prior to this time instant. Therefore, the probability distribution of the mapping $x_k, u_k \rightarrow (x_{k+1}, r_{k+1})$, that is, the probability of ending in state x_{k+1} and receiving reward r_{k+1} while undertaking action u_k in state x_k is defined as:

$$P\{x_{k+1}, r_{k+1} | x_k, u_k, r_k, x_{k-1}, u_{k-1}, \ldots, r_1, x_0, u_0\} \tag{3-1}$$

which is a general state-action transition model of the environment. Ideally however, the state signal is such that it contains a compact summary of all past sensations, yet it does retain all the relevant past information. If the state signal retains all relevant information, it is said to satisfy the *Markov property* [50]. In this case, the probability distribution reads:

$$P\{x_{k+1}, r_{k+1} | x_k, u_k\} \tag{3-2}$$

An RL problem that satisfies this property is called a Markov Decision Process (MDP). In an MDP, one-step dynamics of the environment are sufficient to predict the next state and immediate reward. For any state x, action u, the state transition probability function to a next state x' is given as:

$$\mathcal{P}^u_{xx'} = P\{x_{k+1} = x' | x_k = x \ u_k = u\} \tag{3-3}$$

and the expected value of the next reward is given by:

$$\mathcal{R}^u_{xx'} = E\{r_{k+1} | x_k = x, \ u_k = u, \ x_{k+1} = x'\} \tag{3-4}$$

These two functions completely specify the most important dynamics of a finite MDP. When dealing with continuous state spaces, it is only possible to define a probability

that a region of the state space is reached [19]. Then, the probability of reaching a state x_{k+1} in the region $X_{k+1} \subseteq \mathcal{X}$ from a state x_k by applying action u_k is given as:

$$\mathcal{P}^u_{xx'} = P\{x_{k+1} \in X_{k+1} | x_k,\ u_k\} = \int_{X_{k+1}} f(x_k, u_k, x')dx' \tag{3-5}$$

with f the state transition probability function. In practice, it is not always possible for the agent to completely observe the environment due to, for example, sensor limitations or the specific RL problem the agent is dealing with. In this case, the environment may still have the Markov property but since the agent only observes part of it, it is called a Partially Observable Markov Decision Process (POMDP).

3-2-2 Reward Functions

The goal of RL is that the agent tries to maximize an expected cumulative or total reward described as some function of immediate rewards expected while following policy π:

$$J(\pi) = E\{R_k|\pi\} = E\{f(\dots, r_{k-1}, r_k, r_{k+1}, \dots)|\pi\} \tag{3-6}$$

In most cases, two reward settings f are considered [51], the *discounted sum* and *average reward*. In this thesis, the discounted sum reward setting is considered, which means the reward at time instant k is given as a discounted sum of future rewards:

$$R_k = \sum_{n=0}^{N} \gamma^n r_{k+n+1} \tag{3-7}$$

with γ the *discount rate*. If $\gamma = 1$ the reward is just a sum of all immediate rewards following the actions that lead to the goal. Furthermore, if N is finite, the task is called *episodic*. When it is not episodic, $N = \infty$ and γ is chosen $0 \leq \gamma < 1$ such that the sum remains bounded and future rewards are discounted. Using a discounted reward, the expected cumulative reward as a function of the policy π can be given as the *cost function*[2]:

$$J(\pi) = E\left\{\sum_{n=0}^{N} \gamma^n r_{k+n+1} | x_0, \pi\right\} \tag{3-8}$$

3-2-3 Value Functions

To maximize the total expected reward, the agent makes use of *value functions*, which define for each state or state-action pair how good it is to be in that state following policy π. This means that the value function for the discounted reward function for state $x_k = x$ is defined as:

$$V^\pi(x) = E^\pi\{R_k|x_k = x\} = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{k+n+1} | x_k = x\right\} \tag{3-9}$$

[2]Also as a normalized expected return [46], but since this will not be used it has been left out.

where E_π denotes the expected reward in a state $x_k = x$ when the agent follows policy π. Similarly, a state-action value function for state $x_k = x$ and action $u_k = u$ can be defined as:

$$Q^\pi(x,u) = E^\pi\{R_k|x_k = x,\ u_k = u\} = E\left\{\sum_{k=0}^{\infty}\gamma^k r_{k+n+1}|x_k = x,\ u_k = u\right\} \tag{3-10}$$

which defines the expected return starting from state $x_k = x$, applying action $u_k = u$ and following policy π. Similarly, for the average reward setting, the value functions are given by [19]:

$$V^\pi(x) = E\left\{\sum_{k=0}^{\infty} r_{k+n+1} - J(\pi)|x_k = x\right\} \tag{3-11}$$

$$Q^\pi(x,u) = E\left\{\sum_{k=0}^{\infty}\gamma^k r_{k+n+1} - J(\pi)|x_k = x,\ u_k = u\right\} \tag{3-12}$$

Now, the optimal policy π^* is given as the policy that maximizes the value functions:

$$V^*(x) = \max_\pi V^\pi(x) \tag{3-13}$$

$$Q^*(x,u) = \max_\pi Q^\pi(x,u) \tag{3-14}$$

with $V^*(x),\ Q^*(x,u)$ the optimal state value function and state-action value function, respectively. In classical RL methods, either one of these two is used. Thus, the reinforcement learning problem can be viewed as learning a policy that maximizes either of these two functions.

3-2-4 Policy

As said before, the policy is the optimal mapping from state to actions that the agent tries to find. A deterministic policy is denoted by $\pi_k(x)$, whereas a policy that defines a probability distribution over states and actions is denoted by $\Pi_k(x,u)$. A stochastic policy is a mapping from state-action pairs to probabilities that the action will be chosen in that state. In most RL methods, the policy is deterministic. In the last section, the optimal policy was given as the policy that maximizes either one of the value functions. If during learning the agent assigns to each state or state-action pair a policy that maximizes the value functions, it is called a *greedy policy.*

3-2-5 Exploration-Exploitation Dilemma

If a complete model of the environment is known, the agent can always take greedy actions, thus following the greedy-policy. However, since in most applications no complete model of the environment is known, the agent should not always select its actions greedily. Instead, the agent should sometimes choose to select an arbitrary or random action in order to visit some other part of the state space in which possibly better

solutions can be found. On the other hand, if the agent always chooses its actions randomly, it will never learn. Therefore, a mix between greedy actions (experience) and random actions (exploration) has to be made. This constitutes the exploration-exploitation dilemma. Methods to balance the exploration-exploitation problem are, for example, the ε-greedy method (i.e. selecting a greedy action with probability ε and an exploratory action with probability $1-\varepsilon$) and the Boltzmann-Gibbs method [4].

3-3 Solution Methods

Currently, a convenient way to classify solution methods for the reinforcement learning problem is into *critic-only*, *actor-only* and *Actor-Critic (AC)* [25, 19] methods. The latter two are a subdivision of *policy gradient based* reinforcement learning techniques. In this thesis, an AC method is used, since these methods combine advantages of actor-only and critic-only and they were developed to deal with continuous state and action spaces, which make them more suitable for applications in robotics. In the next section, the different solution methods are briefly discussed with a focus on AC methods.

3-3-1 Critic-only

In critic-only RL methods, the idea is to find the optimal value function first and then derive the optimal policy from this value function. A short overview is given below.

- The solution techniques that incorporate full knowledge of the environment are called Dynamic Programming (DP) methods [50, 4]. In these methods, *Bellman optimality equations* are solved using *policy iteration* and *value iteration*. The interested reader is referred to [50, 4] for an elaborate explanation of these techniques.
- If no perfect model knowledge is available, most critic-only methods use Temporal Difference (TD) learning [50], for which an elementary 1-step update rule for the value function can be given as:

$$V(s_k) \leftarrow V(s_k) + \alpha_k \underbrace{[r_{k+1} + \gamma V(s_{k+1}) - V(s_k)]}_{\text{TD error } \delta_k} \tag{3-15}$$

  with $\alpha_k > 0$ the learning rate (which can be a constant) and δ_k the Temporal Difference (TD)-error. Well-known critic-only methods that use TD learning are Q-learning [50] and $SARSA$ [50], and more recently QV-learning [57] and related QV algorithms, for which the interested reader is referred to [58].

3-3-2 Actor-only

In actor-only methods, an optimal policy is searched for immediately (as opposed to critic-only, where first an optimal value function is searched for to compute the optimal

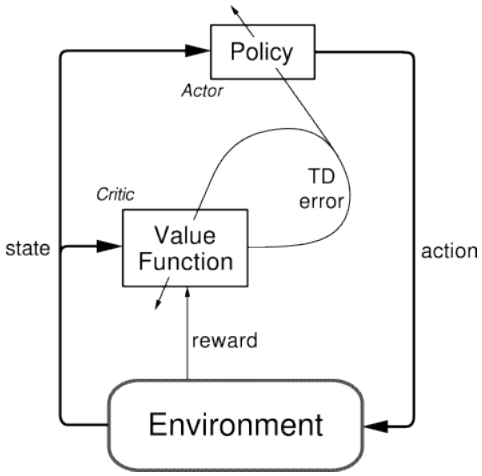


Figure 3-1: A general structure of actor-critic methods. After an action, the action is evaluated using the critic and the TD-error is used to update the actor. Figure adopted from [50].

policy with), which means that the algorithm searches in the policy space (and hence the search space should be restricted). Typically, a class of policies is parameterized by some parameter vector ϑ [19]. In this way, prior knowledge about the task can be incorporated but the drawback is that if no information on the task is available a priori, the class of policies is difficult to define. Examples of actor-only algorithms are Williams's REINFORCE [60] and also evolutionary algorithms have been applied to search for policies directly [21].

Basic Policy Gradient Methods

Basic *policy gradient* methods are actor-only methods in which the policy is parameterized and a search over the policy space is done directly using a gradient. Suppose the differentiable parameterized policy π_ϑ is given with ϑ the parameter vector, then the gradient of the cost function $\nabla J(\pi)$ (3-8) can be immediately given

$$\frac{\partial J}{\partial \pi}\frac{\partial \pi}{\partial \vartheta} = \frac{\partial J}{\partial \vartheta} \tag{3-16}$$

The gradient of $J(\pi(\vartheta))$ can be estimated by simulation and the update law for the parameter vector ϑ can be easily computed using a gradient ascent learning rule:

$$\vartheta_{k+1} = \vartheta_k + \alpha\frac{\partial J}{\partial \vartheta} \tag{3-17}$$

with $\alpha > 0$ small enough such that $J(\pi(\vartheta_{k+1})) \geq J(\pi(\vartheta_k))$.

3-3-3 Actor-Critic Methods

The term actor-critic originates from [5] and a first study on these methods was given in [25]. In actor-critic methods, the policy (actor: selects the actions) and value function (critic: criticizes the actions taken by the actor) are improved separately. This

property makes AC methods very useful in continuous state and action tasks, where the policy and value function can be separately approximated. Also, it allows for an easy incorporation of model knowledge by suitably defining an actor parameterization, so as to speed up learning.

A general overview of an actor-critic method is given in Figure 3-1. Recent work in the field of AC methods focuses on approximating the cost function gradient (3-16). An important contribution has been the work on the Policy Gradient Theorem [51, 25] and the development of Natural Actor-Critic (NAC) algorithms [46, 22]. The interested reader is referred to [19] for an overview of current work in AC methods. In this thesis, the Standard Actor-Critic (S-AC) from [20], which will be explained below, is used as a basis AC method to combine with PBC techniques.

Standard Actor-Critic Define the approximated policy:

$$\hat{\pi} : \mathbb{R}^n \mapsto \mathbb{R}^m \tag{3-18}$$

and the approximated value function:

$$\hat{V} : \mathbb{R}^n \mapsto \mathbb{R} \tag{3-19}$$

Denote the parameterization of the actor by:

$$\vartheta \in \mathbb{R}^p \tag{3-20}$$

and of the critic by:

$$\theta \in \mathbb{R}^q \tag{3-21}$$

Then, a basis function approximation linear in its parameters for the actor can be defined as:

$$\hat{\pi}(x, \vartheta) = \vartheta^T \phi_{\mathrm{a}}(x) \tag{3-22}$$

with $\phi_{\mathrm{a}}(x) \in \mathbb{R}^p$ the actor's basis functions and for the critic as:

$$\hat{V}(x, \theta) = \theta^T \phi_{\mathrm{c}}(x) \tag{3-23}$$

with $\phi_{\mathrm{c}}(x) \in \mathbb{R}^q$ the critic's basis functions. Next, the update equations for the actor and the critic can be defined.

- **Critic** The update equations for the critic (value function) are governed by a TD error (3-15), which consists of the error in the approximation of the critic and the immediate reward received by the environment:

  $$\delta_{k+1} = r_{k+1} + \gamma \hat{V}(x_{k+1}, \theta_k) - \hat{V}(x_k, \theta_k) \tag{3-24}$$

  and the update for the parameter vector θ is based on a gradient ascent learning rule [4, 19]:

  $$\theta_{k+1} = \theta_k + \alpha_{\mathrm{c}} \delta_{k+1} \nabla_\theta \hat{V}(x_k, \theta_k) \tag{3-25}$$

with $\alpha_c \in (0, 1]$ the learning rate of the critic. Eligibility traces are then used to update the value function not only for the previous state, but also for past states according to some decaying schedule. The eligibility trace defined as $e_k(x) \in \mathbb{R}^+$ is updated at each time step by a factor $\gamma\lambda$, with λ the trace-decay parameter [50, 4] and γ still the discount factor. In the S-AC method, *replacing traces* is used, which updates $e_k(x)$ in the following manner for the current state x_k:

$$e_k(x) = \begin{cases} \gamma\lambda e_{k-1} & \text{if } x \neq x_k \\ 1 & \text{if } x = x_k \end{cases} \tag{3-26}$$

such that the parameter vector update (3-25) becomes

$$e_{k+1} = \gamma\lambda e_k(x) + \nabla_\theta \hat{V}(x_k, \theta_k) \tag{3-27}$$
$$\theta_{k+1} = \theta_k + \alpha_c \delta_{k+1} e_{k+1} \tag{3-28}$$

- **Actor** The action selection is based on an exploration-exploitation balance achieved by perturbing the action u_k with a zero-mean exploration term Δu_k:

$$u_k = \hat{\pi}(x_k, \vartheta_k) + \Delta u_k \tag{3-29}$$

such that the update rule for the actor becomes

$$\vartheta_{k+1} = \vartheta_k + \alpha_a \underbrace{\delta_{k+1} \Delta u_k \nabla_\vartheta \hat{\pi}(x_k, \vartheta_k)}_{\nabla_\vartheta J(x_k)} \tag{3-30}$$

with $\alpha_a \in (0, 1]$ the learning rate of the actor and $\nabla_\vartheta J(x_k)$ is the gradient of cost function (3-16), which is heuristically estimated [18].

The basic idea of the S-AC method can thus be viewed as that an action is taken and the TD-error is computed by the critic. If this error is positive (negative) the policy is updated towards (away from) the perturbation Δu_k. An overview of the algorithm is given in Algorithm 3.1.

3-4 Conclusion

RL is a learning control method that can solve complex nonlinear (stochastic) control problems without the need for a process model or explicitly solving a set of equations. A disadvantage of RL methods is that without having model knowledge, learning can be slow and computationally expensive. In particular, AC methods provide an intuitive way of representing a control policy and value function such that these functions can be approximated using standard function approximators. This is convenient when working with continuous state and action spaces, such as in complex robotic systems. The S-AC method is an easy and intuitive AC method in which the policy gradient can be easily derived from the actor parameterization. This property is useful when combining PBC techniques with AC methods. Hence, the combination of the S-AC method with a PBC technique, EB-PBC, will be the topic of the next chapter.

Algorithm 3.1 Standard Actor-Critic

Input: λ, γ, α_{a}, α_{c}.

$e_0(x) = 0 \qquad \forall x$
Initialize x_0, θ_0, ϑ_0
$k \leftarrow 1$
loop
    Execute: Draw action $u_k \sim \hat{\pi}(x_k, \vartheta_k)$, observe next state x_{k+1} and reward $r_{k+1} = \rho(x_k, u_k)$
    Critic Update:
        $\delta_{k+1} = r_{k+1} + \gamma \hat{V}(x_{k+1}, \theta_k) - \hat{V}(x_k, \theta_k)$
        $e_{k+1} = \gamma \lambda e_k + \nabla_\theta \hat{V}(x_k, \theta_k)$
        $\theta_{k+1} = \theta_k + \alpha_{\mathrm{c}} \delta_{k+1} e_{k+1}(x)$
    Actor update:
        $\vartheta_{k+1} = \vartheta_k + \alpha_{\mathrm{a}} \delta_{k+1} \Delta u_k \nabla_\vartheta \hat{\pi}(x_k, \vartheta_k)$
end loop

Chapter 4

Paper

In Chapter 2 all the relevant information on PBC and PH systems was introduced, whereas in Chapter 3 an introduction to AC RL was given. Hence, all the ingredients to combine a PBC technique with AC RL are present, thus the main result of this thesis can be introduced, which is the topic of this chapter and has the form of a paper.

Energy-balancing passivity-based control through reinforcement learning[1]

Abstract— Passivity-based control for port-Hamiltonian systems provides an intuitive way of achieving stabilization by rendering a system passive with respect to a desired storage function. However, in most instances the control law has to be calculated by solving a complex partial differential equation (PDE). This paper considers energy-balancing passivity-based control (EB-PBC), which is a form of PBC in which the closed-loop energy is equal to the difference between the stored and supplied energies. We propose a method to parameterize EB-PBC such that the PDE that has to be solved is split into two terms: a fixed term that satisfies a matching condition following from the EB-PBC framework and a term that can be parameterized, such that control saturation can be incorporated. The parameters of the control law are then found using actor-critic reinforcement learning, enabling learning near-optimal control policies satisfying a desired closed-loop energy landscape. The advantages are that no PDE has to be solved, near-optimal controllers can be generated using energy shaping techniques and the solutions learned can be interpreted in terms of energy shaping and damping injection, which makes it possible to numerically assess stability using passivity theory. From the reinforcement learning perspective, our proposal allows for the class of port-Hamiltonian systems to be incorporated in the actor-critic framework, speeding up the learning thanks to the resulting parameterization of the policy. The method has been

[1]To be submitted to *Automatica*.

successfully applied to the pendulum swing-up problem in simulations and real-life experiments.

4-1 Introduction

Passivity-based control (PBC) [39] is a methodology that achieves the control objective by rendering a system passive with respect to a desired storage function [41]. Different forms of PBC have been successfully applied to design robust controllers [53] for mechanical systems and electrical circuits [41]-[42]. A key feature of PBC is that it exploits structural properties of the system. In this paper, we are interested in the passivity-based control of systems endowed with a special structure, called port-Hamiltonian (PH) systems. PH systems have been widely used in PBC applications [49, 12]. Their geometric structure allows reformulating a PBC problem in terms of solving a set of partial differential equations (PDE's). Much research in the literature concerns solving or simplifying such generally complex PDE's [41].

Passivity-based control of port Hamiltonian systems strongly relies on models. Other control techniques have been developed when no models are known. One such example is reinforcement learning (RL) [50]. RL is a semi-supervised learning control method that can solve optimal (stochastic) control problems for nonlinear systems, without the need for a process model or for explicitly solving complex equations. In RL the controller receives an immediate numerical reward as a function of the process state and possibly control action. The goal is to find an optimal control policy that maximizes the cumulative long-term rewards, which corresponds to maximizing a value function [50]. In this paper, we use actor-critic techniques [25], which are a class of RL methods in which a separate actor and critic are learned. The critic approximates the value function and the actor the policy (control law). Actor-critic reinforcement learning is suitable for problems with continuous state and action spaces. A general disadvantage of RL is that the progress of learning can be very slow and non-monotonic. However, by incorporating (partial) model knowledge, learning can be sped up [20].

In this paper we address three important issues: First, we propose a learning control structure within the PH framework that avoids solving complex PDE's while retaining important properties of the PBC. To this end, first a parameterization of a particular type of PBC, called energy-balancing passivity-based control (EB-PBC), is proposed such that the PDE arising in EB-PBC can be split in a non-assignable part satisfying a matching condition following from the EB-PBC framework and an assignable part that can be parameterized. Then, by applying actor-critic reinforcement learning the parameterized part can be learned, such that the PDE does not have to be explicitly solved. Second, we incorporate control input saturation. To the authors' best knowledge, there has been little account of either incorporating control saturation in PBC for PH systems [13, 2] or facilitating learning for PH systems [14]. Third, from a learning point of view, we present a systematic way of incorporating a priori knowledge into a RL problem. Thus, this work combines the advantages of both aforementioned control techniques, PBC and RL, and mitigates some of their respective disadvantages. The

approach proposed in this paper yields, after learning, an input-saturated controller that can be interpreted in terms of energy shaping control strategies.

The theoretical background on PH systems and actor-critic reinforcement learning is described in Section 4-2 and Section 4-3, respectively. In Section 4-4, our proposal for a parameterization of input-saturated EB-PBC control, compatible with actor-critic reinforcement learning, is introduced. Section 4-6 provides simulation and experimental results for the problem of swinging up an input-saturated inverted pendulum. Section 4-7 relates our results to the literature and Section 4-8 concludes the paper.

4-2 Port-Hamiltonian Systems

Port-Hamiltonian (PH) systems are a natural way of representing a physical system in terms of its energy exchange and of ports modeling the interaction between the basic elements of the system and its environment [42]. The general framework of PH systems was introduced in [33] and was formalized in [54, 53]. In this paper, we consider the input-state-output representation of the PH system which is of the form[2]:

$$\Sigma : \begin{cases} \dot{x} = [J(x) - R(x)]\, \nabla_x H(x) + g(x)u \\ y = g^T(x) \nabla_x H(x) \end{cases} \tag{4-1}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$, $m \leq n$ is the control input, $J, R : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ with $J(x) = -J(x)^T$ and $R(x) = R(x)^T \geq 0$ are the interconnection and damping matrix, respectively, $H : \mathbb{R}^n \to \mathbb{R}$ the Hamiltonian which is the stored energy in the system, $u, y \in \mathbb{R}^m$ are conjugated variables whose product has the unit of power and $g : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ is the input matrix assumed to be full rank. For the remainder of this paper, we denote

$$F(x) \coloneqq J(x) - R(x) \tag{4-2}$$

This matrix satisfies $F(x) + F^T(x) = -2R(x) \leq 0$. System (4-1) satisfies the power-balance equation [41]:

$$\begin{aligned} \dot{H}(x) &= (\nabla_x H(x))^T \dot{x} \\ &= -(\nabla_x H(x))^T R(x) \nabla_x H(x) + u^T y \end{aligned} \tag{4-3}$$

Since $R(x) \geq 0$, we obtain:

$$\dot{H}(x) \leq u^T y \tag{4-4}$$

which is called the passivity inequality, if $H(x)$ is positive semi-definite, and cyclo-passivity inequality, if $H(x)$ is not positive semi-definite nor bounded from below [41]. Hence, systems satisfying (4-4) are called (cyclo-)passive systems.

The goal is to obtain the target closed-loop system:

$$\Sigma_{\text{cl}} : \dot{x} = [J(x) - R_{\text{d}}(x)] \nabla_x H_{\text{d}}(x) \tag{4-5}$$

[2]We use the notation $\nabla_x \coloneqq \partial / \partial x$. Furthermore, all (gradient) vectors are column vectors.

through *energy shaping* using EB-PBC and *damping injection*, such that $H_{\mathrm{d}}(x)$ is the desired closed-loop energy which has a minimum at the desired equilibrium x^* and satisfies:

$$\dot{H}_{\mathrm{d}}(x) = -\left(\nabla_x H_{\mathrm{d}}(x)\right)^T R_{\mathrm{d}}(x) \nabla_x H_{\mathrm{d}}(x) \tag{4-6}$$

which implies (cyclo-)passivity according to (4-3)-(4-4) if the desired damping $R_{\mathrm{d}}(x) \geq 0$. Hence, the control objective is achieved by rendering the closed-loop system passive with respect to the desired storage function $H_{\mathrm{d}}(x)$.

4-2-1 Energy Shaping

Define the added energy function:

$$H_{\mathrm{a}}(x) \coloneqq H_{\mathrm{d}}(x) - H(x) \tag{4-7}$$

A state-feedback law $u_{\mathrm{es}}(x)$ is said to satisfy the energy-balancing property if it satisfies:

$$\dot{H}_{\mathrm{a}}(x) = -u_{\mathrm{es}}^T(x) y \tag{4-8}$$

If (4-8) holds, the desired energy $H_{\mathrm{d}}(x)$ is the difference between the stored and supplied energy. Assuming $g(x) \in \mathbb{R}^{n \times m},\ m < n$, rank $\{g(x)\} = m$, the control law:

$$u_{\mathrm{es}}(x) = g^{\dagger}(x) F(x) \nabla_x H_{\mathrm{a}}(x) \tag{4-9}$$

with $g^{\dagger}(x) = (g^T(x) g(x))^{-1} g^T(x)$ solves the EB-PBC problem with $H_{\mathrm{a}}(x)$ a solution of the following set of PDE's [41]:

$$\begin{bmatrix} g^{\perp}(x) F^T(x) \\ g^T(x) \end{bmatrix} \nabla_x H_{\mathrm{a}}(x) = 0 \tag{4-10}$$

with $g^{\perp}(x) \in \mathbb{R}^{(n-m) \times n}$ the full rank left-annihilator of $g(x)$, i.e. $g^{\perp}(x) g(x) = 0$.

4-2-2 Damping Injection

Damping is injected by feeding back the (new) passive output $g^T(x) \nabla_x H_{\mathrm{d}}(x)$,

$$u_{\mathrm{di}}(x) = -K(x) g^T(x) \nabla_x H_{\mathrm{d}}(x) \tag{4-11}$$

with $K(x) \in \mathbb{R}^{m \times m}, K(x) = K^T(x) \geq 0$ such that:

$$R_{\mathrm{d}}(x) = R(x) + g(x) K(x) g^T(x) \tag{4-12}$$

Hence, the full control law consists of an energy shaping part and a damping injection part:

$$\begin{aligned} u(x) &= u_{\mathrm{es}}(x) + u_{\mathrm{di}}(x) \\ &= g^{\dagger}(x) F(x) \nabla_x H_{\mathrm{a}}(x) \\ &\quad - K(x) g^T(x) \nabla_x H_{\mathrm{d}}(x) \end{aligned} \tag{4-13}$$

4-2-3 Control Saturation

In this paper, control saturation is incorporated in (4-13) by setting:

$$u_{\text{sat}}(x) = \varsigma(u(x)) \tag{4-14}$$

with $\varsigma : \mathbb{R}^m \to S,\ S \subset \mathbb{R}^m$, a saturation function such that:

$$\varsigma(u(x)) \in S\ \forall u \tag{4-15}$$

Hence, the target dynamics will not be exactly equal to (4-5) but:

$$\dot{x} = \begin{cases} [J(x) - R_{\text{d}}(x)]\,\nabla_x H_{\text{d}}(x) \text{ if } u_{\text{sat}}(x) = u(x) \\ [J(x) - R(x)]\,\nabla_x H(x) + g(x)u_{\text{sat}}(x) \text{ otherwise} \end{cases} \tag{4-16}$$

We will show in Section 4-6-3 that it is possible to numerically assess stability based on passivity for the regions where (4-5) holds. Before presenting our main result, we first introduce the second control paradigm used in this paper: actor-critic reinforcement learning.

4-3 Actor-Critic Reinforcement Learning

In reinforcement learning, the system to be controlled (called the 'environment' in the RL literature) is modeled as a Markov decision process (MDP). In a deterministic setting this MDP is defined by the tuple $M(X, U, f, \rho)$, where X is the state space, U the action space and $f : X \times U \to X$ the state transition function that describes the process to be controlled that returns the state x_{k+1} after applying action u_k in state x_k. The vector x_k is obtained by applying a zero-order hold discretization $x_k = x(kT_s)$ with T_s the sampling time. The reward function is defined by $\rho : X \times U \to \mathbb{R}$ and returns a scalar reward $r_{k+1} = \rho(x_{k+1}, u_k)$ after each transition. The goal of RL is to find an optimal control policy $\pi : X \to U$ by maximizing an expected cumulative or total reward described as some function of the immediate rewards expected. In this paper, we consider a discounted sum of rewards. The value function $V^\pi : X \to \mathbb{R}$,

$$\begin{aligned} V^\pi(x) &= \sum_{i=0}^{\infty} \gamma^i r^\pi_{k+i+1} \\ &= \sum_{i=0}^{\infty} \gamma^i \rho(x_{k+i+1}, \pi(x_{k+i})), \quad x = x_k \end{aligned} \tag{4-17}$$

approximates this discounted sum during learning while following policy π where $\gamma \in [0, 1)$ is the discount factor.

When dealing with large and/or continuous state and action spaces it is necessary to approximate the value function and policy. Actor-critic (AC) algorithms [5, 25] learn a separate actor (policy π) and critic (value function V^π). The critic approximates and updates (improves) the value function. Then, the actor's parameters are updated

in the direction of that improvement. The actor and critic are usually defined by a differentiable parameterization such that gradient ascent can be used to update the parameters. This is beneficial when dealing with continuous action spaces [51].

In this paper, the temporal-difference based Standard Actor-Critic (S-AC) algorithm from [20] is used. Define the approximated policy $\hat{\pi} : \mathbb{R}^n \to \mathbb{R}^m$ and the approximated value function as $\hat{V} : \mathbb{R}^n \to \mathbb{R}$. Denote the parameterization of the actor by $\vartheta \in \mathbb{R}^p$ and of the critic by $\theta \in \mathbb{R}^q$. The temporal difference [50]:

$$\delta_{k+1} = r_{k+1} + \gamma \hat{V}(x_{k+1}, \theta_k) - \hat{V}(x_k, \theta_k) \tag{4-18}$$

is used to update the critic parameters using the following gradient ascent update rule:

$$\theta_{k+1} = \theta_k + \alpha_c \delta_{k+1} \nabla_\theta \hat{V}(x_k, \theta_k) \tag{4-19}$$

in which $\alpha_c > 0$ is the learning rate. Eligibility traces $e_k \in \mathbb{R}^q$ [50] can be used to speed up learning by including reward information about previously visited states. The update for the critic parameters becomes:

$$e_{k+1} = \gamma \lambda e_k + \nabla_\theta \hat{V}(x_k, \theta_k) \tag{4-20}$$
$$\theta_{k+1} = \theta_k + \alpha_c \delta_{k+1} e_{k+1} \tag{4-21}$$

with $\lambda \in [0, 1)$ the trace-decay rate. The policy approximation can be updated in a similar fashion, as described below.

RL needs exploration in order to visit new, unseen parts of the state-action space so as to possibly find better policies. This is achieved by perturbing the policy with a zero-mean random exploration term Δu_k. The overall control action now becomes:

$$u_k = \hat{\pi}(x_k, \vartheta_k) + \Delta u_k \tag{4-22}$$

The policy update is such that the policy parameters are updated towards (away from) Δu_k if the temporal difference (4-18) is positive (negative). This leads to the following policy update rule:

$$\vartheta_{k+1} = \vartheta_k + \alpha_a \delta_{k+1} \Delta u_k \nabla_\vartheta \hat{\pi}(x_k, \vartheta_k) \tag{4-23}$$

with $\alpha_a > 0$ the actor learning rate.

4-4 Energy-Balancing Actor-Critic

In this section we present our main results. Our approach is that we will use the PDE (4-10) and split it into an assignable, parameterizable part and an unassignable part that satisfies a matching condition. In this way, it is possible to parameterize the desired closed-loop Hamiltonian $H_d(x)$ and simultaneously satisfy (4-10). After that, we parameterize the damping matrix $K(x)$. The two parameterized variables - the desired closed-loop energy $H_d(x)$ and damping $K(x)$ - are then suitable for Actor-Critic RL by defining two actors for these variables.

First, we reformulate the PDE (4-10) in terms of the desired closed-loop energy $H_{\mathrm{d}}(x)$ by applying (4-7):

$$\underbrace{\begin{bmatrix} g^{\perp}(x)F^T(x) \\ g^T(x) \end{bmatrix}}_{A(x)} (\nabla_x H_{\mathrm{d}}(x) - \nabla_x H(x)) = 0 \tag{4-24}$$

and we denote the kernel of $A(x)$ as:

$$\ker(A(x)) = \{N(x) \in \mathbb{R}^{n \times b} : A(x)N(x) = 0\} \tag{4-25}$$

such that (4-24) reduces to:

$$\nabla_x H_{\mathrm{d}}(x) - \nabla_x H(x) = N(x)a \tag{4-26}$$

with $a \in \mathbb{R}^b$. Suppose that (an example is given further on) the state vector x can be split:

$$x = \begin{bmatrix} w \\ z \end{bmatrix} \tag{4-27}$$

where $z \in \mathbb{R}^c$ and $w \in \mathbb{R}^d$, $c + d = n$ corresponding to the zero and non-zero elements of $N(x)$ such that:

$$\begin{bmatrix} \nabla_w H_{\mathrm{d}}(x) \\ \nabla_z H_{\mathrm{d}}(x) \end{bmatrix} - \begin{bmatrix} \nabla_w H(x) \\ \nabla_z H(x) \end{bmatrix} = \begin{bmatrix} N_w(x) \\ 0 \end{bmatrix} a \tag{4-28}$$

We assume that the matrix $N_w(x)$ is rank d, which is always true for mechanical systems (see section 4-5). It is clear that $\nabla_z H_{\mathrm{d}}(x) = \nabla_z H(x)$, which we call the matching condition, and hence $\nabla_z H_{\mathrm{d}}(x)$ cannot be chosen freely. Thus, only the desired closed-loop energy gradient vector $\nabla_w H_{\mathrm{d}}(x)$ is free for assignment for which we use a differentiable basis function approximation linear in its parameters:

$$\nabla_w \hat{H}_{\mathrm{d}}(x, \xi) = \left(\xi^T \frac{\partial \phi_H(w)}{\partial w} \right)^T + \frac{\partial \bar{H}_{\mathrm{d}}(x)}{\partial w} \tag{4-29}$$

with $\xi \in \mathbb{R}^e$ a parameter vector and $\frac{\partial \phi_H(w)}{\partial w}$ the gradient of a basis function $\phi_H(w) \in \mathbb{R}^e$, with e chosen sufficiently large to represent the assignable desired closed-loop energy. The term $\frac{\partial \bar{H}_{\mathrm{d}}(x)}{\partial w}$ is the gradient of the unassignable part $\bar{H}_{\mathrm{d}}(x)$ of the approximated desired closed-loop energy (we show in section 4-5 that for mechanical systems the term $\bar{H}_{\mathrm{d}}(x)$ can be the system's kinetic energy). The approximated desired closed-loop energy consists of the sum of the unassignable part and an assignable parameterized part:

$$\hat{H}_{\mathrm{d}}(x, \xi) = \bar{H}_{\mathrm{d}}(x) + \xi^T \phi_H(w) + C \tag{4-30}$$

with C chosen to render $\hat{H}_{\mathrm{d}}(x, \xi)$ non-negative. The elements of the desired damping matrix $K(x)$ of (4-13), denoted $\hat{K}(x, \Psi)$, can be parameterized in a similar way:

$$[\hat{K}(x, \Psi)]_{ij} = \sum_{l=1}^{f} [\Psi]_{ijl} [\phi_K(x)]_l \tag{4-31}$$

with $\Psi \in \mathbb{R}^{m\times m\times f}$ and

$$[\Psi]_{ijl} = [\Psi]_{jil} \tag{4-32}$$

a parameter vector such that $\hat{K}(x,\Psi) = \hat{K}^T(x,\Psi)$, $(i,j) = 1,\ldots,m$ and $\phi_K(x) \in \mathbb{R}^f$ basis functions. We purposefully do not impose $\hat{K}(x,\Psi) \geq 0$ to allow the injection of energy in the system via the damping term. Although this breaches the passivity criterion of (4-4) we shall see that local stability can still be numerically demonstrated using passivity analysis in Section 4-6-3.

The control law (4-13) now becomes:

$$\begin{aligned} u(x,\xi,\Psi) &= g^\dagger(x)F(x)\begin{bmatrix}\nabla_w \hat{H}_\mathrm{d}(x,\xi) - \nabla_w H(x)\\ 0\end{bmatrix} \\ &\quad - \hat{K}(x,\Psi)g^T(x)\nabla_x \hat{H}_\mathrm{d}(x,\xi) \\ &= g^\dagger(x)F(x)\begin{bmatrix}\left(\xi^T\frac{\partial \phi_H(w)}{\partial w}\right)^T + \frac{\partial \bar{H}_\mathrm{d}(x)}{\partial w} - \nabla_w H(x)\\ 0\end{bmatrix} \\ &\quad - \hat{K}(x,\Psi)g^T(x))\begin{bmatrix}\left(\xi^T\frac{\partial \phi_H(w)}{\partial w}\right)^T + \frac{\partial \bar{H}_\mathrm{d}(x)}{\partial w}\\ \nabla_z H(x)\end{bmatrix} \end{aligned} \tag{4-33}$$

Now, we are ready to introduce the update equations for the parameter vectors $\xi,\ [\Psi]_{ij}$. Denote by $\xi_k,\ [\Psi_k]_{ij}$ the value of the parameters at the discrete time step k. The policy $\hat{\pi}$ of the actor-critic reinforcement learning algorithm is chosen equal to the control law parameterized by (4-33):

$$\hat{\pi}(x_k,\xi_k,\Psi_k) \coloneqq u(x_k,\xi_k,\Psi_k) \tag{4-34}$$

Taking the saturation into account, the control action with exploration (4-22) becomes:

$$u_k = \varsigma\left(\hat{\pi}(x_k,\xi_k,\psi_k) + \Delta u_k\right) \tag{4-35}$$

The exploration term to be used in the actor update (4-23) must be adjusted to respect the saturation:

$$\Delta \bar{u}_k = u_k - \hat{\pi}(x_k,\xi_k,\psi_k) \tag{4-36}$$

Furthermore, we obtain the following gradients of the saturated policy:

$$\nabla_\xi \varsigma(\hat{\pi}) = \nabla_{\hat{\pi}}\varsigma(\hat{\pi})\nabla_\xi \hat{\pi} \tag{4-37}$$

$$\nabla_{[\Psi]_{ij}} \varsigma(\hat{\pi}) = \nabla_{\hat{\pi}}\varsigma(\hat{\pi})\nabla_{[\Psi]_{ij}} \hat{\pi} \tag{4-38}$$

Although not explicitly indicated in the previous equations, the (lack of) differentiability of the saturation function ς has to be considered for the problem at hand. Finally, the actor parameters $\xi_k, [\Psi_k]_{ij}$ are updated according to (4-23), respecting the saturated policy gradients. For the parameters of the desired Hamiltonian we obtain:

$$\xi_{k+1} = \xi_k + \alpha_{\mathrm{a},\xi}\delta_{k+1}\Delta\bar{u}_k\nabla_\xi \varsigma\left(\hat{\pi}(x_k,\xi_k,\Psi_k)\right) \tag{4-39}$$

Algorithm 4.1 Energy-Balancing Actor-Critic

Input: System (4-1), λ, γ, α_{a} for each actor, α_{c}.

$e_0(x) = 0 \qquad \forall x$
Initialize x_0, θ_0, ξ_0, Ψ_0
$k \leftarrow 1$
loop
  Execute:
    Draw $\Delta u_k \sim \mathcal{N}(0, \sigma^2)$, calculate action $u_k = \varsigma\left(\hat{\pi}(x_k, \xi_k, \psi_k) + \Delta u_k\right)$, $\Delta \bar{u}_k = u_k - \hat{\pi}(x_k, \xi_k, \psi_k)$
    Observe next state x_{k+1} and calculate reward $r_{k+1} = \rho(x_{k+1}, u_k)$
  Critic:
    Temporal difference: $\delta_{k+1} = r_{k+1} + \gamma \hat{V}(x_{k+1}, \theta_k) - \hat{V}(x_k, \theta_k)$
    Eligibility trace: $e_{k+1} = \gamma \lambda e_k + \nabla_\theta \hat{V}(x_k, \theta_k)$
    Critic update: $\theta_{k+1} = \theta_k + \alpha_{\mathrm{c}} \delta_{k+1} e_{k+1}$
  Actors:
    Actor 1 ($\hat{H}_{\mathrm{d}}(x, \xi)$): $\xi_k + \alpha_{\mathrm{a},\xi} \delta_{k+1} \Delta \bar{u}_k \nabla_\xi \varsigma\left(\hat{\pi}(x_k, \xi_k, \Psi_k)\right)$
    Actor 2 ($\hat{K}(x, \Psi)$):
      for $i, j = 1, \ldots, m$ **do**
        $[\Psi_{k+1}]_{ij} = [\Psi_k]_{ij} + \alpha_{\mathrm{a},[\Psi]_{ij}} \delta_{k+1} \Delta \bar{u}_k \nabla_{[\Psi_k]_{ij}} \varsigma\left(\hat{\pi}(x_k, \xi_k, \Psi_k)\right)$
      end for
end loop

and for the parameters of the desired damping we have:

$$\begin{aligned}[\Psi_{k+1}]_{ij} = [\Psi_k]_{ij} + \\ \alpha_{\mathrm{a},[\Psi]_{ij}} \delta_{k+1} \Delta \bar{u}_k \nabla_{[\Psi_k]_{ij}} \varsigma\left(\hat{\pi}(x_k, \xi_k, \Psi_k)\right)\end{aligned} \tag{4-40}$$

where $(i, j) = 1, \ldots, m$, while observing (4-32).

Algorithm 4.1 gives the entire Energy-Balancing Actor-Critic algorithm.

4-5 Mechanical Systems

To illustrate the method, consider a fully actuated mechanical system of the form:

$$\Sigma_{\mathrm{m}} : \begin{cases} \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -I & -\bar{R} \end{bmatrix} \begin{bmatrix} \nabla_q H(q,p) \\ \nabla_p H(q,p) \end{bmatrix} + \begin{bmatrix} 0 \\ G \end{bmatrix} u \\ y = \begin{bmatrix} 0 & G \end{bmatrix} \begin{bmatrix} \nabla_q H(q,p) \\ \nabla_p H(q,p) \end{bmatrix} \end{cases} \tag{4-41}$$

with $q \in \mathbb{R}^{\bar{n}}$, $p \in \mathbb{R}^{\bar{n}}$ ($\bar{n} = \frac{n}{2}$, n even) the generalized positions and momenta, respectively, $G = I$ and $\bar{R} \in \mathbb{R}^{\bar{n} \times \bar{n}}$ the damping matrix. The system admits (4-1) with $\bar{R} > 0$ and the Hamiltonian:

$$H(q,p) = \frac{1}{2} p^T M^{-1}(q) p + P(q) \tag{4-42}$$

with $M(q) = M^T(q) > 0$ the inertia matrix and $P(q)$ the potential energy. For the system (4-41) it holds that rank $\{g(x)\} = \bar{n}$ and the state vector can be split into part $w = [q_1, q_2, \ldots, q_{\bar{n}}]^T$ and part $z = [p_1, p_2, \ldots, p_{\bar{n}}]^T$. Since $g(x) = [0\ I]^T$ its annihilator can be written as $g^{\perp}(x) = [\bar{g}(x)\ 0]$, for an arbitrary matrix $\bar{g}(x)$. Equation (4-24) can then be written as:

$$\begin{bmatrix} 0 & -\bar{g}(x) \\ 0 & I \end{bmatrix} (\nabla_x H_{\mathrm{d}}(x) - \nabla_x H(x)) = 0 \tag{4-43}$$

resulting in the following expression where $N_q(x)$ is of rank $\bar{n}$:

$$\begin{bmatrix} \nabla_q H_{\mathrm{d}}(x) \\ \nabla_p H_{\mathrm{d}}(x) \end{bmatrix} - \begin{bmatrix} \nabla_q H(x) \\ \nabla_p H(x) \end{bmatrix} = \begin{bmatrix} N_q(x) \\ 0 \end{bmatrix} a \tag{4-44}$$

This means that only the potential energy can be shaped, which is widely known in EB-PBC for mechanical systems. The approximated desired closed-loop energy (4-30) reads:

$$\begin{aligned} \hat{H}_{\mathrm{d}}(x, \xi) &= \frac{1}{2} p^T M^{-1}(q) p + \xi^T \phi_H(q) \\ &= \bar{H}_{\mathrm{d}}(x) + \xi^T \phi_H(w) \end{aligned} \tag{4-45}$$

where the unassignable part $\bar{H}_{\mathrm{d}}(x)$ represents the kinetic energy part of the system Hamiltonian (4-42) and $\xi^T \phi_H(w)$ the assignable desired potential energy.

Control law (4-33) becomes:

$$\begin{aligned} u(x, \xi, \Psi) = g^{\dagger}(x) F(x) &\begin{bmatrix} \left(\xi^T \frac{\partial \phi_H(q)}{\partial q}\right)^T + \frac{\partial \bar{H}_{\mathrm{d}}(x)}{\partial q} - \nabla_q H(x) \\ 0 \end{bmatrix} \\ - \hat{K}(x, \Psi) g^T(x)) &\begin{bmatrix} \left(\xi^T \frac{\partial \phi_H(q)}{\partial q}\right)^T \\ \nabla_p H(x) \end{bmatrix} \end{aligned} \tag{4-46}$$

and the actor updates can be defined for each parameter according to (4-39)–(4-40). For underactuated mechanical systems, i.e. $G = [0\ I]^T$, the split state vector z is enlarged with those q-coordinates that cannot be actuated directly because these coordinates correspond to the zero elements of $N(x)$, (i.e. the matrix $N_q(x)$ is no longer rank $\bar{n}$).

4-6 Example: Pendulum Swing-up

To validate our method, the problem of swinging up an inverted pendulum subject to control saturation is studied in simulation and using the actual physical setup depicted in Fig. 4-1.

The pendulum swing-up is a low-dimensional, but highly nonlinear control problem commonly used as a benchmark in the RL literature [20] and it has also been studied

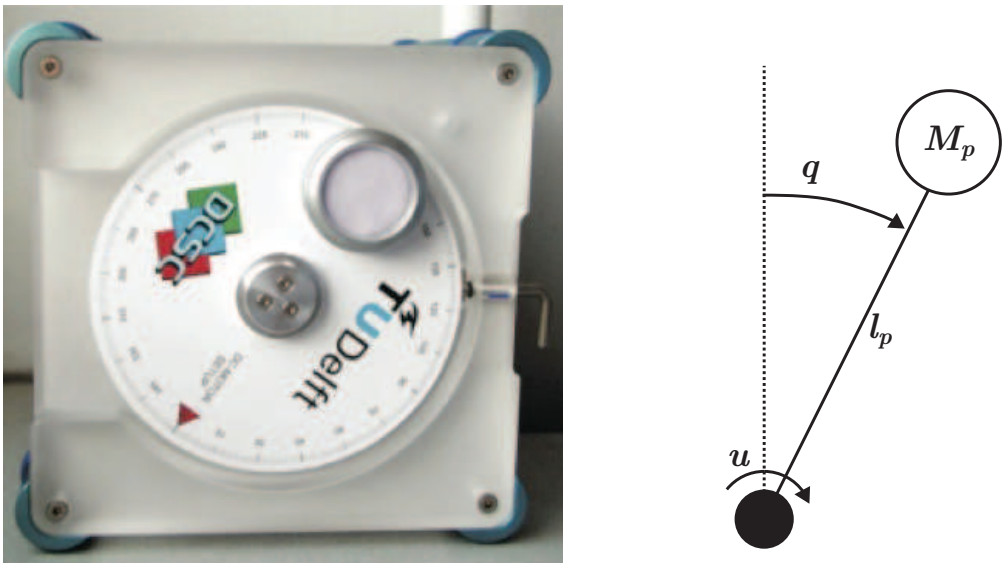


Figure 4-1: Inverted pendulum setup.

in PBC [2]. The equations of motion admit (4-41) and read:

$$\Sigma_{\mathrm{p}}: \begin{cases} \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -\bar{R}(\dot{q}) \end{bmatrix} \begin{bmatrix} \nabla_q H(q,p) \\ \nabla_p H(q,p) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_{\mathrm{p}}}{R_{\mathrm{p}}} \end{bmatrix} u \\ y = \begin{bmatrix} 0 & \frac{K_{\mathrm{p}}}{R_{\mathrm{p}}} \end{bmatrix} \begin{bmatrix} \nabla_q H(q,p) \\ \nabla_p H(q,p) \end{bmatrix} \end{cases} \tag{4-47}$$

with q the angle of the pendulum and p the angular momentum, thus we denote the full measurable state $x = [q, p]^T$. The damping term is:

$$\bar{R}(\dot{q}) = b_{\mathrm{p}} + \frac{K_{\mathrm{p}}^2}{R_{\mathrm{p}}} + \frac{\sigma_{\mathrm{p}}}{|\dot{q}|} \tag{4-48}$$

for which it holds that $\bar{R}(\dot{q}) > 0,\ \forall \dot{q}$. Furthermore, we denote the Hamiltonian:

$$H(q,p) = \frac{p^2}{2J_{\mathrm{p}}} + P(q) \tag{4-49}$$

with:

$$P(q) = M_{\mathrm{p}} g_{\mathrm{p}} l_{\mathrm{p}} (1 + \cos q) \tag{4-50}$$

The model parameters are given in Table 4-1.

Applying (4-28) yields:

$$\begin{bmatrix} \nabla_q H_{\mathrm{d}}(x) \\ \nabla_p H_{\mathrm{d}}(x) \end{bmatrix} - \begin{bmatrix} \nabla_q H(x) \\ \nabla_p H(x) \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} a \tag{4-51}$$

Table 4-1: Inverted pendulum model parameters

Model parameters	Symbol	Value	Units
Pendulum inertia	J_p	$1.90 \cdot 10^{-4}$	kgm^2
Pendulum mass	M_p	$5.2 \cdot 10^{-2}$	kg
Gravity	g_p	9.81	$\mathrm{m/s}^2$
Pendulum length	l_p	$4.20 \cdot 10^{-2}$	m
Dynamic friction	b_p	$2.48 \cdot 10^{-6}$	Nms
Static friction	σ_p	$1.0 \cdot 10^{-3}$	N
Torque constant	K_p	$5.60 \cdot 10^{-2}$	Nm/A
Rotor resistance	R_p	9.92	Ω

Hence, only the potential energy can be shaped such that the desired Hamiltonian (4-30) reads:

$$\begin{aligned}\hat{H}_\mathrm{d}(x,\xi) &= \bar{H}_\mathrm{d}(p) + \xi^T \phi_H(q) \\ &= \frac{p^2}{2J_\mathrm{p}} + \hat{P}_\mathrm{d}(q,\xi)\end{aligned} \tag{4-52}$$

with:

$$\hat{P}_\mathrm{d}(q,\xi) = \xi^T \phi_H(q) \tag{4-53}$$

the desired potential energy. Furthermore, as there is only one input, $\hat{K}(x,\Psi)$ becomes a scalar:

$$\hat{K}(x,\psi) = \psi^T \phi_K(x) \tag{4-54}$$

Thus, control law (4-33) becomes:

$$\begin{aligned}u(x,\xi,\psi) = g^\dagger(x)F(x) &\begin{bmatrix} \left(\xi^T \frac{\partial \phi_H(q)}{\partial q}\right)^T - \nabla_q H(x) \\ 0 \end{bmatrix} \\ - \psi^T \phi_K(x) g^T(x)) &\begin{bmatrix} \left(\xi^T \frac{\partial \phi_H(q)}{\partial q}\right)^T \\ \nabla_z H(x) \end{bmatrix}\end{aligned} \tag{4-55}$$

or:

$$\begin{aligned}u(x,\xi,\psi) = \left(\frac{K_\mathrm{p}}{R_\mathrm{p}}\right)^{-1} &\left(- \xi^T \frac{\partial \phi_H(q)}{\partial q} - \left(- \frac{\partial P(q)}{\partial q} \right.\right. \\ &\left.\left. - \bar{R}(\dot{q})\dot{q} \right)\right) - \psi^T \phi_K(x) \frac{K_\mathrm{p}}{R_\mathrm{p}} \dot{q}\end{aligned} \tag{4-56}$$

which we define as the policy $\hat{\pi}(x,\xi,\psi)$. Hence, we have two actor updates:

$$\xi_{k+1} = \xi_k + \alpha_{\mathrm{a},\xi} \delta_{k+1} \Delta \bar{u}_k \nabla_\xi \varsigma \left(\hat{\pi}(x_k,\xi_k,\psi_k)\right) \tag{4-57}$$

$$\psi_{k+1} = \psi_k + \alpha_{\mathrm{a},\psi} \delta_{k+1} \Delta \bar{u}_k \nabla_\psi \varsigma \left(\hat{\pi}(x_k,\xi_k,\psi_k)\right) \tag{4-58}$$

for the desired potential energy $\hat{P}_\mathrm{d}(q,\xi)$ and the desired damping $\hat{K}(x,\psi)$, respectively.

4-6-1 Function Approximation

To approximate the critic and the two actors, function approximators are necessary. In this paper we use the Fourier basis [26] because of its ease of use, the possibility to incorporate information about the symmetry in the system and the ability to ascertain properties useful for stability analysis of this specific problem. We define a multivariate Nth-order[3] Fourier basis for n dimensions as:

$$\phi(\bar{x}) = \sum_{i=1}^{(N+1)^n} \cos(\pi c_i^T \bar{x}) \tag{4-59}$$

with $c_i \in \mathbb{Z}^n$, which means that all possible $N+1$ integer values, or frequencies, are combined in a vector in $\mathbb{Z}^n$ to create a matrix $c \in \mathbb{Z}^{n\times(N+1)^n}$ containing all possible frequency combinations. For example,

$$c_1 = [0\ 0]^T,\ \ c_2 = [1\ 0]^T,\ \ \dots\ ,\ c_{(3+1)^2} = [4\ 4]^T \tag{4-60}$$

for a 3$^{\text{rd}}$-order Fourier basis in 2 dimensions. The state x is scaled according to:

$$\bar{x}_i = \frac{x_i - x_{i,\min}}{x_{i,\max} - x_{i,\min}} (\bar{x}_{i,\max} - \bar{x}_{i,\min}) + \bar{x}_{i,\min} \tag{4-61}$$

for $i = 1, \dots, n$ with $(\bar{x}_{i,\min}, \bar{x}_{i,\max}) = (-1, 1)$. Projecting the state variables onto this symmetrical range has several advantages. First, this means that the policy will be periodic with period $T = 2$, such that it wraps around (i.e., modulo 2π) and prevents discontinuities at the boundary values of the angle ($x = [\pi \pm \epsilon, p]$, ϵ very small). Second, learning will be faster because updating the value function and policy for some x also applies to the sign-opposite value of x. Third, $\dot{\hat{P}}_{\text{d}}(0, \xi) = 0$ by the choice of parameterization, which is beneficial for stability analysis. Although the momentum will now also be periodic in the value function and policy, this is not a problem because the value function and policy approximation are restricted to a domain and the momentum itself is also restricted to the same domain using saturation.

We adopt the adjusted learning rate from [26] such that:

$$\alpha_{\text{a}_i,\xi} = \frac{\alpha_{\text{a}_\text{b},\xi}}{\|c_i\|_2}, \qquad \alpha_{\text{a}_i,\psi} = \frac{\alpha_{\text{a}_\text{b},\psi}}{\|c_i\|_2} \tag{4-62}$$

for $i = 1, \dots, (N+1)^n$ with $\alpha_{\text{a}_\text{b},\xi}$, $\alpha_{\text{a}_\text{b},\psi}$ the base learning rate for the two actors (Table 4-2) and $\alpha_{\text{a}_1,\xi} = \alpha_{\text{a}_\text{b},\xi}$, $\alpha_{\text{a}_1,\psi} = \alpha_{\text{a}_\text{b},\psi}$ to avoid division by zero for $c_1 = [0\ 0]^T$. Equation (4-62) implies that parameters corresponding to basis functions with higher (lower) frequencies are learned slower (faster).

4-6-2 Simulation

The task is to learn to swing up and stabilize the pendulum from the initial position pointing down $x_0 = [\pi, 0]^T$ to the desired equilibrium position at the top $x^* = [0, 0]^T$.

[3] 'Order' refers to the order of approximation; 'dimensions' to the number of states in the system.

Since the control action is saturated, the system is not able to swing up the pendulum directly, but rather it must swing back and forth to build up momentum to eventually reach the equilibrium.

The reward function ρ is defined such that it has its maximum in the desired unstable equilibrium and penalizes other states via:

$$\rho(x, u) = Q_{\mathrm{r}} \left(\cos(q) - 1\right) - R_{\mathrm{r}} p^2 \tag{4-63}$$

with:

$$Q_{\mathrm{r}} = 25 \ , \ R_{\mathrm{r}} = \frac{0.1}{J_p^2} \tag{4-64}$$

This reward function is consistent with the mapping $S^1 \to \mathbb{R}$ for the angle and proved to improve performance over a purely quadratic reward, such as the one used in e.g. [20]. For the critic, we define the basis function approximation as:

$$\hat{V}(x, \theta) = \theta^T \phi_{\mathrm{c}}(x) \tag{4-65}$$

with $\phi_{\mathrm{c}}(x)$ a 3^{rd}-order Fourier basis resulting in 16 learnable parameters θ in the domain $[q_{min}, q_{max}] \times [p_{min}, p_{max}] = [-\pi, \pi] \times [-8\pi J_p, 8\pi J_p]$.

Actor 1 ($\hat{P}_{\mathrm{d}}(q, \xi)$) is parameterized using a 3^{rd}-order Fourier basis in the range $[-\pi, \pi]$ resulting in 4 learnable parameters. Actor 2 ($\hat{K}(x, \psi)$) is also parameterized using a 3^{rd}-order Fourier basis for the full state space, in the same domain as the critic. Exploration is done at every time step by randomly perturbing the action with a normally distributed zero-mean white noise with standard deviation $\sigma = 1$, i.e.:

$$\Delta u \sim \mathcal{N}(0, 1) \tag{4-66}$$

We incorporate saturation by defining the saturation function (4-14) as:

$$\varsigma(u_k) = \begin{cases} u_k & \text{if } |u_k| \leq u_{\max} \\ \mathrm{sgn}(u_k) u_{\max} & \text{otherwise} \end{cases} \tag{4-67}$$

Recall that the saturation must be taken into account in the policy gradients by applying (4-37)-(4-38). The parameters were all initialized with zero vectors of appropriate dimensions, i.e. $(\theta_0, \ \xi_0, \ \psi_0) = 0$.

The algorithm was first run with the system simulated in Matlab for 200 trials of three seconds each (with a near-optimal policy, the pendulum needs approximately one second to swing up). Each trial begins in the initial position x_0. This simulation was repeated 50 times to get an estimate of the average, minimum, maximum and confidence regions for the learning curve. The simulation parameters are given in Table 4-2.

Fig. 4-2 shows the average learning curve obtained after 50 simulations. The algorithm shows good convergence and on average needs about 2 minutes (40 trials) to reach a near-optimal policy. The initial drop in performance is caused by the zero-initialization of the value function (critic), which is too optimistic compared to the true value function. Therefore, the controller explores a large part of the state space

Table 4-2: Simulation parameters

Simulation parameters	Symbol	Value	Units
Number of trials	$-$	200	-
Trial duration	T_t	3	s
Sample time	T_s	0.03	s
Decay rate	γ	0.97	-
Eligibility trace decay	λ	0.65	-
Exploration variance	σ^2	1	-
Max control input	u_max	3	V
Learning rate of critic	α_c	0.05	-
Learning rate of $\hat{P}_\mathrm{d}(q,\xi)$	$\alpha_{\mathrm{a_b},\xi}$	1×10^{-10}	-
Learning rate of $\hat{K}(x,\psi)$	$\alpha_{\mathrm{a_b},\psi}$	0.2	-

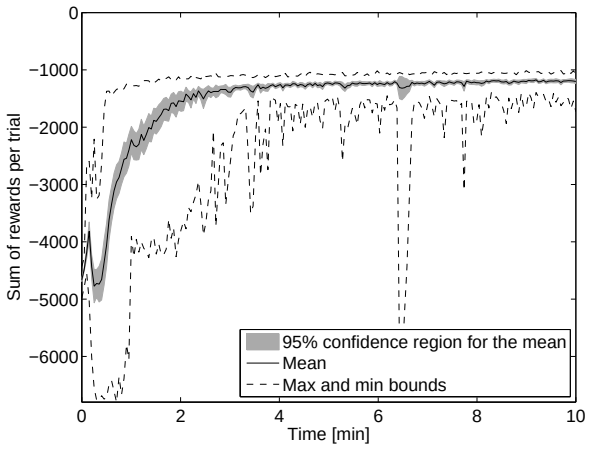
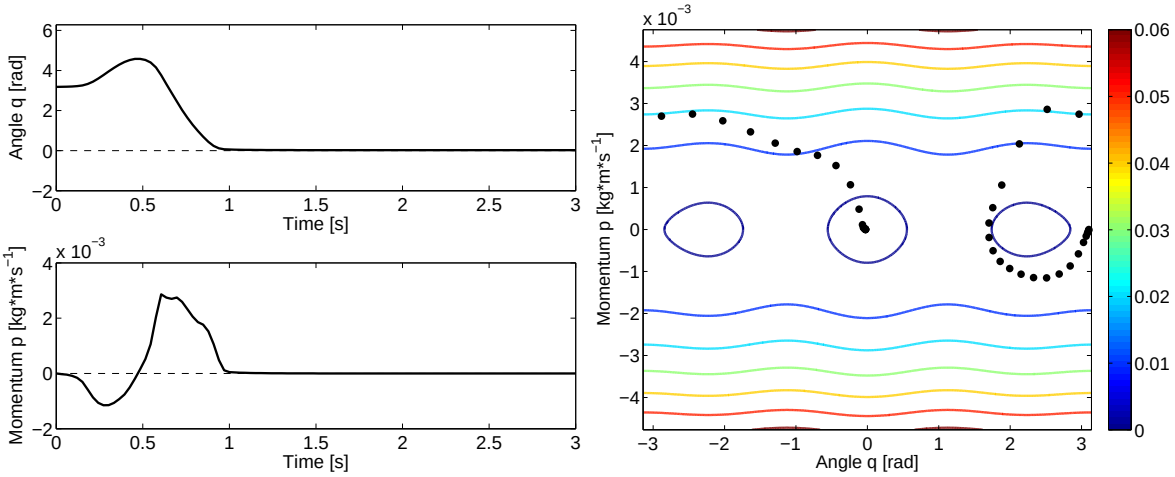


Figure 4-2: Results for the EBAC method for 50 learning simulations.

and receives a lot of negative rewards before it learns the true value of the states. A simulation using the policy learned in a typical experiment is given in Fig. 4-3a. As can be seen, the pendulum swings back once to build up momentum to eventually get to the equilibrium. The desired Hamiltonian $\hat{H}_\mathrm{d}(x,\xi)$ (4-52), acquired through learning, is given in Fig. 4-3b. There are three minima, of which one corresponds to the desired equilibrium. The other two equilibria are undesirable wells that come from the shaped potential energy $\hat{P}_\mathrm{d}(q,\xi)$ (Fig. 4-4a). These minima are the result of the algorithm trying to swing up the pendulum in a single swing, which is not possible due to the saturation. Hence, a swing-up strategy is necessary to avoid staying in these wells. The number of these undesirable wells is a function of the control saturation and it is independent of the number of basis functions chosen to approximate $\hat{P}_\mathrm{d}(q,\xi)$. The learned damping $\hat{K}(x,\psi)$ (Fig. 4-4b) is positive (white) towards the equilibrium thus extracting energy from the system, while it is negative (gray) in the region of the initial state. The latter corresponds to pumping energy into the system, which is necessary to build up momentum for the swing-up and to escape the undesirable wells of $\hat{P}_\mathrm{d}(q,\xi)$.



(a) Simulated response **(b)** Desired Hamiltonian

Figure 4-3: Simulation results for the angle q (a, top), momentum p (a, bottom) and the desired closed-loop Hamiltonian $H_{\mathrm{d}}(x, \xi, \psi)$ (b) including the simulated trajectory (black dots) using the policy learned.

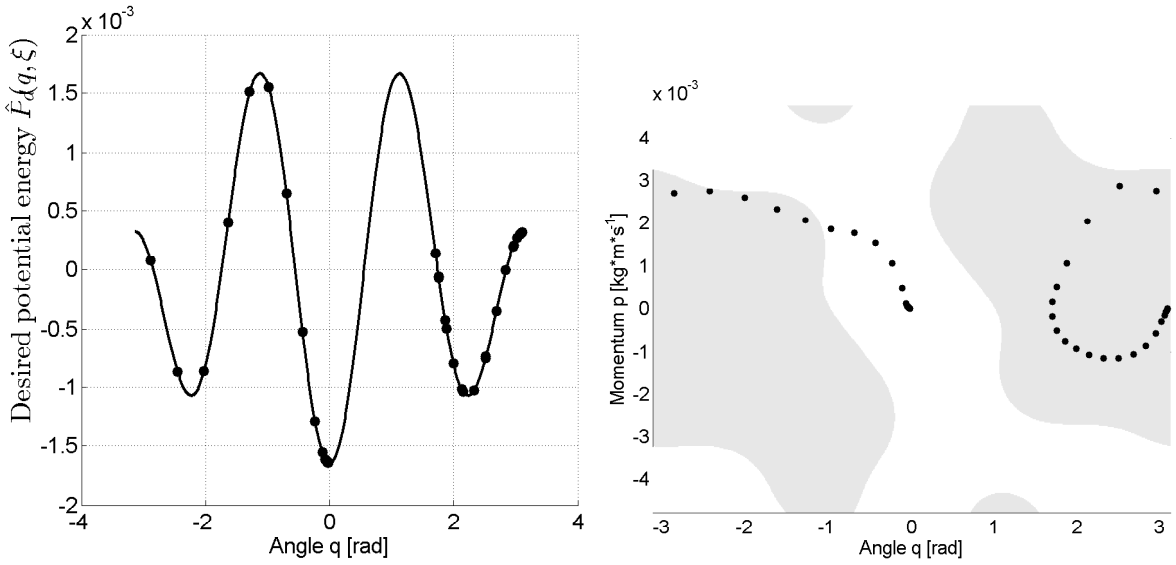
A disadvantage is that control law (4-56), with the suggested basis functions, is always zero for the set $\Omega = \{x \mid x = (0 + j\pi, 0),\ j = 1, 2, \dots\}$ which implies that it is zero not only at the desired equilibrium, but also at the initial state x_0. During learning this is not a problem because there is constant exploration, but after learning the system should not be initialized in exactly x_0 otherwise it will stay in this set. It can be overcome by initializing with a small perturbation ϵ around x_0. In real-life systems it will also be less a problem because there is generally noise present on the sensors.

4-6-3 Stability of the Learned Controller

Since control saturation is present, the target dynamics satisfy (4-16). Hence, to conclude local stability of x^* based on (4-6), we calculate $\dot{\hat{H}}_{\mathrm{d}}(x, \xi)$ for the unsaturated case (Fig. 4-5a)[4] and the saturated case ($\dot{\hat{H}}_{\mathrm{d,sat}}(x, \xi)$) and compute the sign of the difference (Fig. 4-5b). By looking at Fig. 4-5b, it appears that $\exists \delta \subset \mathbb{R}^n : |x - x^*| < \delta$ such that $\dot{\hat{H}}_{\mathrm{d,sat}}(x, \xi) = \dot{\hat{H}}_{\mathrm{d}}(x, \xi)$. It can be seen from Fig. 4-5b that such a δ exists, i.e., a small gray region around the equilibrium x^* exists. Hence, we can use $\dot{\hat{H}}_{\mathrm{d}}(x, \xi)$ around x^* and assess stability using (4-6).

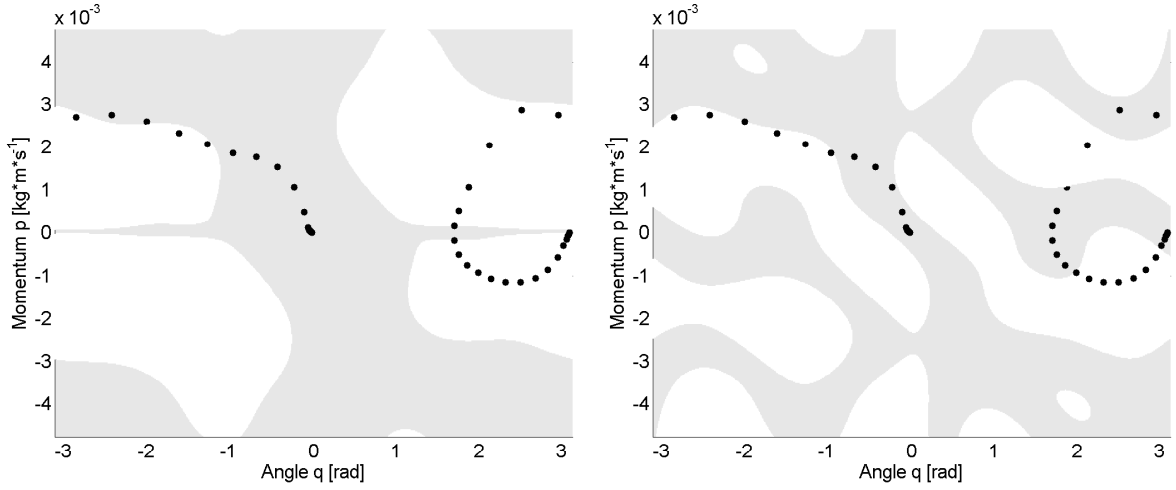
From Fig. 4-3b it follows that $\hat{H}_{\mathrm{d}}(x, \xi) > 0$ for all states in Fig. 4-3b. From Fig. 4-4a

[4]Fig. 4-5a is sign-opposite to Fig. 4-4b, which is logical, because the negative (positive) regions of $\hat{K}(x, \psi)$ correspond to negative (positive) damping which corresponds to a positive (negative) value of $\dot{\hat{H}}_{\mathrm{d}}(x, \xi)$ based on (4-6).



(a) $\hat{P}_{\mathrm{d}}(q,\xi)$ **(b)** $\mathrm{sgn}\left(\hat{K}(x,\psi)\right)$

Figure 4-4: Desired potential energy (a) and desired damping (b) (gray: negative; white: positive) for a typical learning experiment. The black dots indicate the value of the respective quantity for the simulation of Fig. 4-3a.



(a) $\mathrm{sgn}\left(\dot{\hat{H}}_{\mathrm{d}}(x,\psi)\right)$ **(b)** $\dot{\hat{H}}_{\mathrm{d,diff}}(x,\psi)$

Figure 4-5: Signum of $\dot{\hat{H}}_{\mathrm{d}}(x,\psi)$ (a) indicating positive (white) and negative (gray) regions and (b) $\dot{\hat{H}}_{\mathrm{d,diff}}(x,\psi) = \mathrm{sgn}\left(\dot{\hat{H}}_{\mathrm{d}}(x,\psi) - \dot{\hat{H}}_{\mathrm{d,sat}}(x,\psi)\right)$ indicating regions where $\dot{\hat{H}}_{\mathrm{d}}(x,\psi) = \dot{\hat{H}}_{\mathrm{d,sat}}(x,\psi)$ (gray) and $\dot{\hat{H}}_{\mathrm{d}}(x,\psi) \neq \dot{\hat{H}}_{\mathrm{d,sat}}(x,\psi)$ (white). Black dots indicate the simulated trajectory.

we infer that locally,

$$\arg\min \hat{P}_{\mathrm{d}}(q,\xi) = x^* \tag{4-68}$$

$$\dot{\hat{P}}_{\mathrm{d}}(x^*,\xi) = 0 \tag{4-69}$$

$$\ddot{\hat{P}}_{\mathrm{d}}(x^*,\xi) > 0 \tag{4-70}$$

the latter two of which naturally result from the basis function definition. Furthermore, from Fig. 4-4b it can be seen that around x^*, $\hat{K}(x,\psi) > 0$. Hence, in a region δ around x^*,

$$\hat{H}_\mathrm{d}(x,\xi) > 0 \tag{4-71}$$

$$\dot{\hat{H}}_\mathrm{d}(x,\xi) \leq 0 \tag{4-72}$$

$$\dot{\hat{H}}_\mathrm{d}(x^*,\xi) = 0 \tag{4-73}$$

which implies local asymptotic stability of x^*. Extensive simulations show that similar behaviour is always achieved.

4-6-4 Real-time Experiments

Using the physical setup shown in Fig. 4-1, 20 learning experiments were run using identical settings as in the simulations. The result is given in Fig. 4-6. The algorithm shows slightly slower convergence - about 3 minutes of learning (60 trials) to reach a near-optimal policy instead of 40 - and a less consistent average when compared to Fig. 4-2. This can be attributed to a combination of model mismatch and the symmetrical basis functions (through which it is not possible to incorporate non-symmetrical friction that is present in the real system). Overall though, the performance can be considered good when compared to the simulation results. Also, the same performance dip is present which can again be attributed to the optimistic value function initialization.

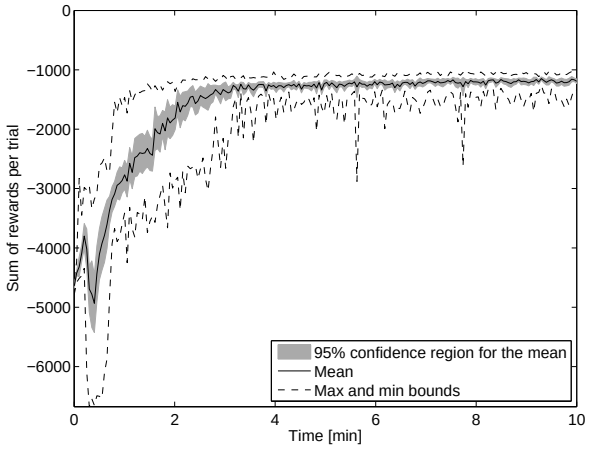


Figure 4-6: Results for the EBAC method for 20 learning experiments with the real physical system.

4-7 Final Remarks and Future Directions

In [41], the equivalence between EB-PBC and *Control-by-Interconnection* (CbI) using a state-modulated source is established. This means that EB-PBC can be viewed as an

interconnection of the PH system (4-1) with a controller Hamiltonian ($H_{\mathrm{C}}(\zeta)$) system of the form:

$$\Sigma_{\mathrm{C}} : \begin{cases} \dot{\zeta} = u_{\mathrm{C}} \\ y_{\mathrm{C}} = \dfrac{\partial H_{\mathrm{C}}(\zeta)}{\partial \zeta} \end{cases} \tag{4-74}$$

and the state-modulated (SM) interconnection:

$$\Sigma_I^{SM} : \left\{ \begin{bmatrix} u \\ u_{\mathrm{C}} \end{bmatrix} = \begin{bmatrix} 0 & -a(x) \\ a^T(x) & 0 \end{bmatrix} \begin{bmatrix} y \\ y_{\mathrm{C}} \end{bmatrix} \right. \tag{4-75}$$

with $a(x)$ equal to (4-9) subject to (4-10). Damping $K(x)$ is then injected by feeding back the passive output y. In our method, $a(x) = u(x, \xi, \psi)$ and $K(x) = \hat{K}(x, \psi)$. Hence, our method can also be interpreted in terms of CbI of PH systems. In [2], a family of smooth controllers for swinging up a pendulum is proposed. However, the method by Åstrom et al. does not facilitate learning and is not generally applicable to systems of the form (4-1). The iterative learning control scheme in [14] does facilitate learning for systems of the form (4-1), but it does not incorporate control saturation. Also, the learning itself is based on a desired output, whereas our learning method is based on a numerical reward which can be received from the environment. Escobar et al. [13] use energy shaping and damping injection to formulate PBC type controllers subject to input saturation. However, their method does not facilitate learning and our method uses the PH framework, which allows for a broader class of dynamical systems to which the method can be applied. Finally, EB-PBC is possibly too limited, for example in underactuated systems, hence our future research will focus on extending the approach proposed here to IDA-PBC [36] to include changing the interconnection matrix $J(x)$ of (4-1).

4-8 Conclusions

In this paper, we have presented a method to systematically parameterize EB-PBC control laws subject to control saturation such that the PDE's arising can be split into an unassignable part verifying a matching condition and an assignable, parameterized part. The parameters are then found by making use of actor-critic reinforcement learning. In this way, we are able to learn a closed-loop energy landscape for a PH system of the form (4-1). The advantages are that no PDE has to be solved, optimal controllers can be generated using energy-based control techniques and the solutions acquired by means of reinforcement learning can be interpreted in terms of energy shaping and damping injection, which makes it possible to numerically assess stability using passivity theory. By making use of the model knowledge the actor-critic method is able to quickly learn near-optimal policies. A drawback is that for multiple input systems, generating many actor updates for the desired damping matrix can be computationally expensive.

Chapter 5

Energy-Balancing Actor-Critic

In addition to Chapter 4, this chapter comprises additional results of the Energy-Balancing Actor-Critic (EBAC) method. First, the results for the single Degree Of Freedom (DOF) pendulum from Chapter 4 are further investigated. After that, the original equations of motion for the pendulum are used in simulations to study the effect of changing important functions and parameters present in the EBAC method.

5-1 Inverted Pendulum - Simulation

In addition to the results presented in Chapter 4, this section provides background information and further investigation on the pendulum to get a better understanding of the EBAC method. Tests were done using different reward and saturation functions, model parameters and initial values. First, explanatory results on Chapter 4 are given.

5-1-1 Paper

In this section, additional information on the results presented in Chapter 4 is given.

Policy and Value Function

A typical[1] policy and value function for the simulation under investigation in Chapter 4 is given in Figure 5-1. The policy is as expected and in line with results from e.g. [20]. The value function however, shows peaks at undesired positions at the top and bottom. This can be attributed to the pumping-damping term $\hat{K}(x, \psi)$. If this term learns too fast, the system assigns high values to states close to the equilibrium but with high velocity. In other words, the system is swinging up, keeps rotating and assigns large

[1] For references to the data and MATLAB® files used for each figure the reader is referred to Appendix A.

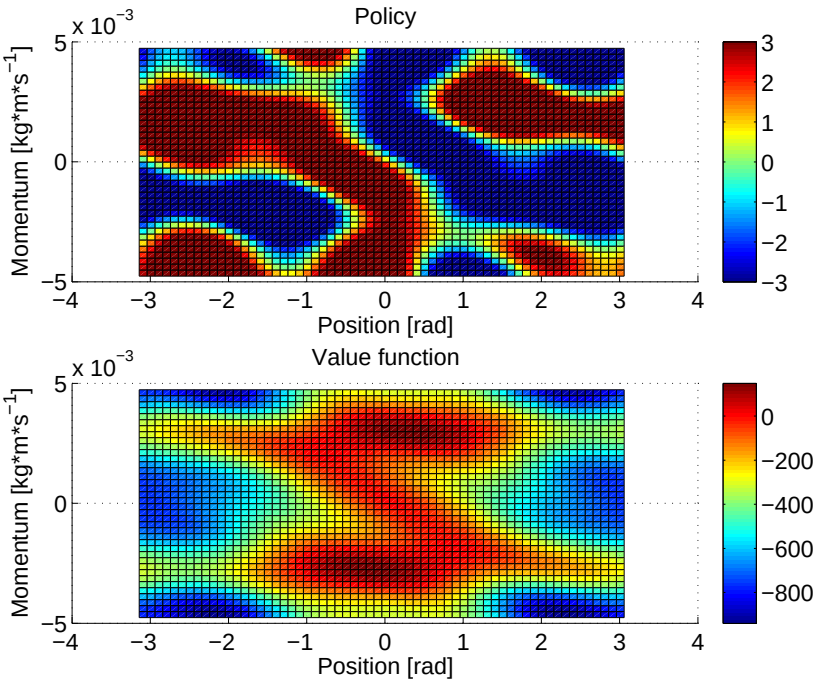


Figure 5-1: A typical policy and value function for the simulation results of Chapter 4. As can be seen, there are some states that have been assigned a high value while this should not happen.

values to the states close to the equilibrium it then visits. It is possible to eliminate these peaks by decreasing the learning rates of the critic α_c and of the pumping-damping term $\alpha_{a,\psi}$ such that a better value function can be learned. However, this would decrease learning speed and since the policy is very good it is not necessary to do so.

Learning Phases

In Figure 5-2, which is a zoomed-in version of Figure 4-2, different learning phases can be identified.

Phase 1 In the first phase (light red) the algorithm tries to swing up the pendulum.

Phase 2 In the second phase (light green) the algorithm has achieved a successful swing-up but after that, it increases the pumping of energy (i.e. decrease $\hat{K}(x,\psi)$), causing the pendulum to spin around very fast which causes the algorithm to receive a lot of negative rewards.

Phase 3 In the third phase (light brown) the algorithm is injecting damping, thus increasing the pumping-damping term $\hat{K}(x,\psi)$ in the relevant area around the equilibrium.

Phase 4 In the final phase (light blue) the algorithm has converged to a near optimal solution and swings up and stabilizes the pendulum.

These phases will be referenced to in the subsequent part of this chapter.

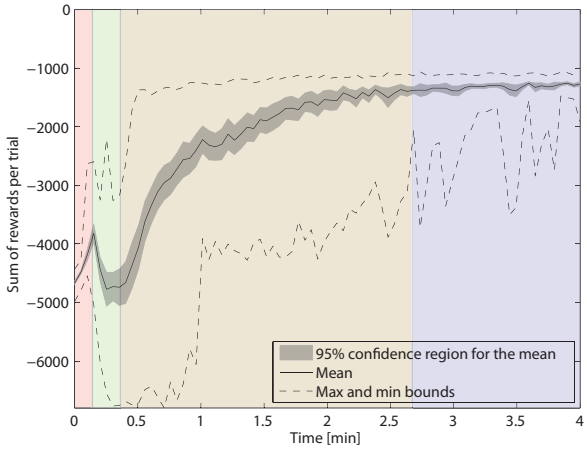


Figure 5-2: The learning phases the algorithm passes through while learning: trying to swing up (light red), a swing-up and rotating (light green), injecting damping (light brown), convergence to optimal solution (light blue).

Control Action

In Chapter 4, a simulation with the learned control policy was performed in order to show that the learned policy indeed swings up and stabilizes the pendulum after learning. For this simulation, the control action is displayed in Figure 5-3.

- The top graph shows the position q, from which it can be seen that the pendulum is stabilized at the top in approximately one second.
- The middle graph displays the separate contributions of the learned potential energy term u_{es} and the learned damping term u_{di} of the control input (recall (4-13)). At the beginning ($T = 0$s until $T \approx 0.2$s) both terms calculate a control action that pulls the pendulum to a side. Then, it can be seen that the energy shaping term calculates a control action to keep it there, which is in line with Figure 4-4a. Hence, it 'falls' into an undesirable well of $\hat{P}_{\text{d}}(x, \xi)$. However, the damping term u_{di} now applies a larger control action in the opposite direction, corresponding to the pumping of energy into the system. This is line with Figure 4-4b, which means that the system is now in a gray region of Figure 4-4b. After about 0.7s the damping term has pumped enough energy into the system and the undesirable wells of $\hat{P}_{\text{d}}(x, \xi)$ have been left. The energy shaping term now 'takes over', and stabilizes the pendulum at its local minimum, which is the desired equilibrium x^*. The latter is in line with Figure 4-4a, which means that the pendulum 'falls' to its desired equilibrium point.
- The total control action, which is the sum of the energy shaping and damping injection part, is displayed in the bottom graph of Figure 5-3, which can be seen as the graphical representation of the aforementioned explanation. In this graph, two events need further explanation:
  - There is a sudden 'dip' in control action after approximately 0.6s. This is

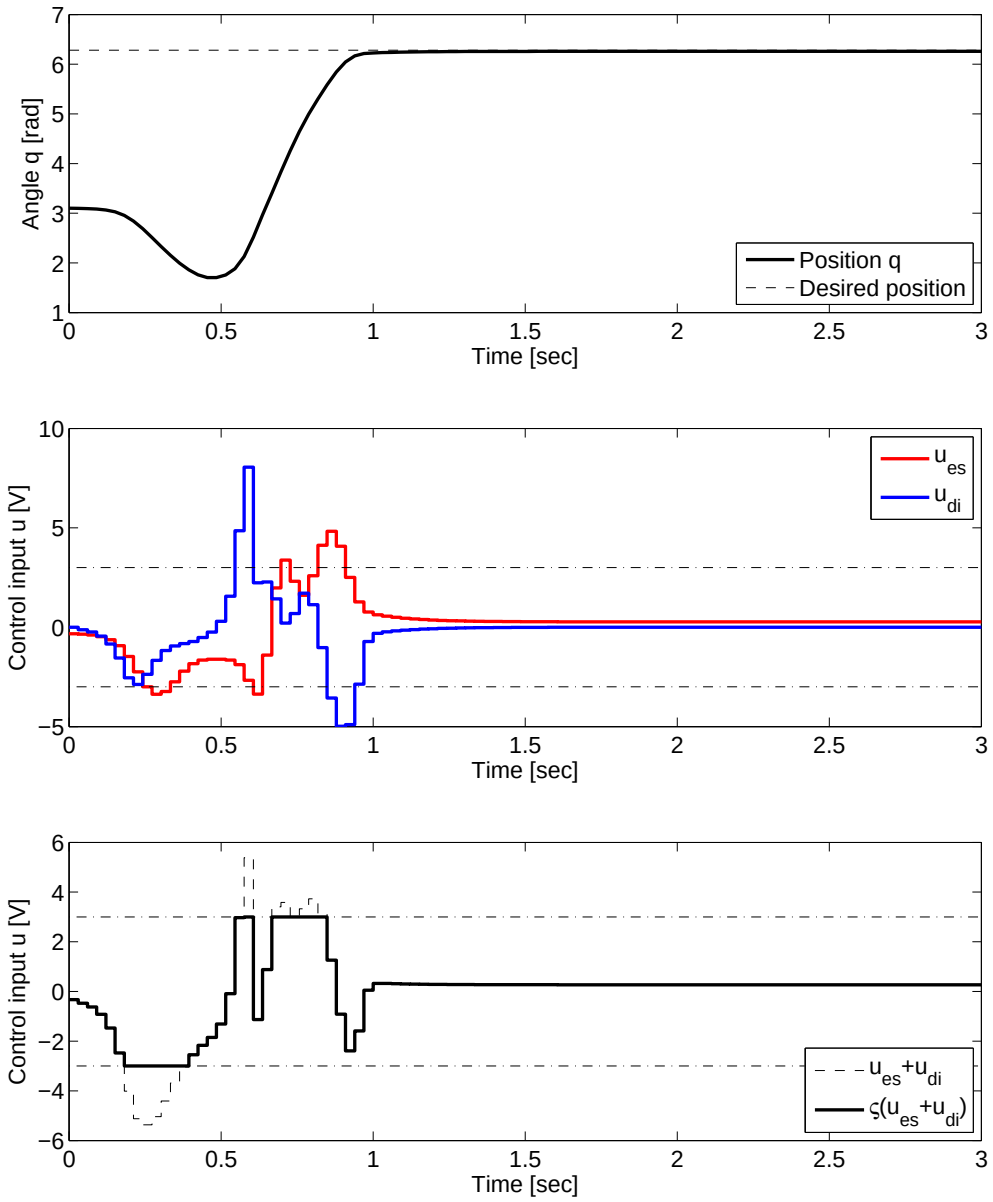


Figure 5-3: Control action for the simulation results of Chapter 4 using a learned policy. As can be seen, the pendulum is stabilized at the desired equilibrium position x^* (top graph). In the middle graph, the evolution of the two control actions is shown: the learned energy shaping term u_{es} (red solid) and the damping injection term u_{di} (blue solid) along with the saturation bounds of the control action $u_{\min}$, $u_{\max}$ (black dashed-dotted). The bottom graph shows the sum of u_{di} and u_{es} (black dashed) and the saturated control action (black solid) that is finally sent to the pendulum system.

caused by the fact that both the control terms do not work together perfectly. This means that after 0.6s, the damping term has injected enough energy into the system to facilitate the swing-up, thus that control part already declines. However, the energy shaping term is still 'pulling' the pendulum into an undesirable well of $\hat{P}_{\mathrm{d}}(x,\xi)$, hence causing the sudden decline in overall control action. Since the velocity is high enough, this is not a problem. After this, the velocity is high enough to overcome the opposite undesirable well of $\hat{P}_{\mathrm{d}}(x,\xi)$, and finally the pendulum 'falls' into the desired well after approximately 0.8s where it is attracted to by the energy term u_{es}.

– The control action never exactly becomes zero, which is caused by the energy shaping term u_{es} (middle graph). It is unclear why this happens, because there seems to be no steady-state error by looking at the position (top graph of Figure 5-3) and by definition, the position feedback at the top is zero. Also, in the experiments using the physical set-up, this phenomenon does not occur.

The nice result of Figure 5-3 is that the cooperation of the energy shaping and damping term can be illustrated, which further enlarges the understanding of what happens in the EBAC method and provides an intuitive explanation of Figures 4-4a–4-4b.

Passivity

In Chapter 4, local asymptotic stability was demonstrated by analyzing (4-6) locally. In this section, it will be mathematically shown why it is not possible to conclude passivity for the entire state space using (4-6). Therefore, recall (4-3), which for the desired Hamiltonian reads:

$$\dot{\hat{H}}_{\mathrm{d}}(x,\xi) = \frac{\partial^T \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x}\dot{x} \tag{5-1}$$

$$= \frac{\partial^T \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x}\left(F\frac{\partial H(x)}{\partial x} + gu(x,\xi,\psi)\right) \tag{5-2}$$

Now, the unsaturated control law (4-56) can be inserted in (5-2), by using:

$$\varsigma(u(x,\xi,\psi)) \leq u(x,\xi,\psi) \quad \forall x \tag{5-3}$$

from which it can be inferred that if passivity can be proven for the unsaturated control action it also means that the system is passive with respect to the saturated control action (because the saturated control action is always smaller, hence inserting less energy into the system). Thus, inserting (4-56):

$$\dot{\hat{H}}_{\mathrm{d}}(x,\xi) = \frac{\partial^T \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x}\left(F\frac{\partial H(x)}{\partial x} + g\left(g^{\dagger}\left(F\left(\frac{\partial \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x} - \frac{\partial H(x)}{\partial x}\right)\right)\right.\right.$$
$$\left.\left. -\hat{K}(x,\psi)g^T\frac{\partial \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x}\right)\right) \tag{5-4}$$

$$= \frac{\partial^T \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x}\left(F\frac{\partial \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x} - g\hat{K}(x,\psi)g^T\frac{\partial \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x}\right) \tag{5-5}$$

Now, using the fact that $J(x) = -J^T(x)$,

$$\dot{\hat{H}}_{\mathrm{d}}(x,\xi) = -\frac{\partial^T \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x}\left(\overbrace{\underbrace{R(x)}_{\geq 0} + g\hat{K}(x,\psi)g^T}^{R_{\mathrm{d}}}\right)\frac{\partial \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x} \tag{5-6}$$

$$\leq -\frac{\partial^T \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x} g\hat{K}(x,\psi)g^T \frac{\partial \hat{H}_{\mathrm{d}}(x,\xi)}{\partial x} \tag{5-7}$$

However, since $\hat{K}(x,\psi) \ngeq 0 \ \forall x$, no conclusion can be made about passivity using a straight-forward analysis of the desired Hamiltonian. This is logical, since the pumping of energy, corresponding to a negative damping $\hat{K}_{\mathrm{d}}(x,\psi)$, is needed to overcome the undesirable wells of $\hat{P}_{\mathrm{d}}(x,\xi)$. For future research, it might be interesting to study the change of $\hat{H}_{\mathrm{d}}(x,\xi)$ along a set of trajectories by initializing at different x_0, or investigating the (possible) invariance of the sets of gray points in Figure 4-4b, where $\hat{K}(x,\psi) \leq 0$, such that - if possible - it can be proven that every point in such a set will converge in finite time to the set of white points, where $\hat{K}(x,\psi) \geq 0$. In that case, global (asymptotic) stability for the system could be mathematically proven. Finally, it is worth investigating the passivity *during learning*. This means that it would be possible to guarantee dissipation during the learning progress, such that a form of safety can be guaranteed.

5-1-2 Equations of Motion Original Model

For most of the simulation results, the following equations of motion of the inverted pendulum were used as opposed to the system (4-47):

$$\Sigma_{\mathrm{pend}} : \begin{cases} \begin{bmatrix}\dot{q}\\ \dot{p}\end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -\left(b_{\mathrm{p}} + \frac{K_{\mathrm{p}}^2}{R_{\mathrm{p}}}\right)\end{bmatrix}\begin{bmatrix}\nabla_q H(q,p)\\ \nabla_p H(q,p)\end{bmatrix} + \begin{bmatrix}0\\ \frac{K_{\mathrm{p}}}{R_{\mathrm{p}}}\end{bmatrix} u \\ y = \begin{bmatrix}0 & \frac{K_{\mathrm{p}}}{R_{\mathrm{p}}}\end{bmatrix}\begin{bmatrix}\nabla_q H(q,p)\\ \nabla_p H(q,p)\end{bmatrix} \end{cases} \tag{5-8}$$

with the Hamiltonian:

$$H(q,p) = \frac{p^2}{2J_{\mathrm{p}}} + \underbrace{M_{\mathrm{p}} g_{\mathrm{p}} l_{\mathrm{p}}(1+\cos q)}_{P(q)} \tag{5-9}$$

with q the position of the pendulum and p the momentum, thus the full measurable state can be denoted as $x = [q,p]$. The model parameters are given in Table 5-1. The difference between (4-47) and (5-8) is the absence of dry friction in the latter, which was later added (See Section 5-2-2) to obtain a better model and was hence used in the paper. Also, there is a slight difference in parameter values that is the result of an identification of parameter values for system (4-47).

Table 5-1: Inverted pendulum model parameters (original).

Model parameters	Symbol	Value	Units
Pendulum inertia	J_p	$1.91 \cdot 10^{-4}$	kgm^2
Pendulum mass	M_p	$5.50 \cdot 10^{-2}$	kg
Gravity	g_p	9.81	$\mathrm{m/s}^2$
Pendulum length	l_p	$4.20 \cdot 10^{-2}$	m
Dynamic friction	b_p	$3 \cdot 10^{-6}$	Nms
Torque constant	K_p	$5.36 \cdot 10^{-2}$	Nm/A
Rotor resistance	R_p	9.50	Ω

5-1-3 Reward Function

In this section, the effect of changing the reward function is investigated. For the inverted pendulum, it is possible to define many reward functions that yield good results such as a quadratic reward [20], a cosine reward, Gaussian rewards, etc. Because the quadratic reward has been well studied [20] for the inverted pendulum while the cosine reward provides an intuitive way of penalizing the angle q, the following reward functions are considered:

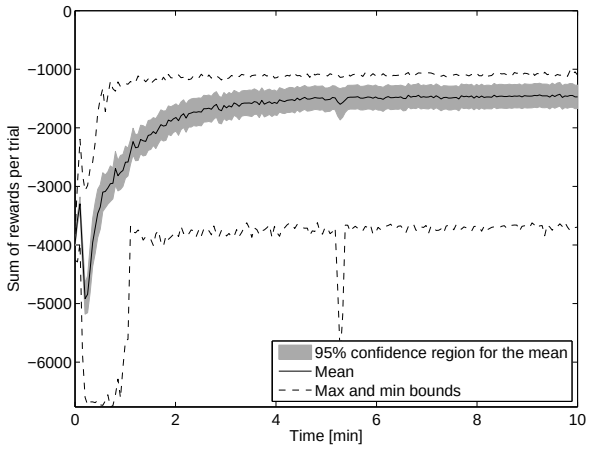
$$r_\mathrm{A}: \; r_{k+1}(x_k, u_k) = -x_k^T Q_\mathrm{r} x_k - u_k^T P_\mathrm{r} r u_k \tag{5-10}$$
$$r_\mathrm{B}: \; r_{k+1}(x_k, u_k) = -x_k^T Q_\mathrm{r} x_k \tag{5-11}$$
$$r_\mathrm{C}: \; r_{k+1}(x_k, u_k) = W_\mathrm{r} \left(\cos(q_k) - 1\right) \tag{5-12}$$
$$r_\mathrm{D}: \; r_{k+1}(x_k, u_k) = W_\mathrm{r} \left(\cos(q_k) - 1\right) - p_k^2 Q_{\mathrm{r}(2,2)} \tag{5-13}$$
$$r_\mathrm{E}: \; r_{k+1}(x_k, u_k) = W_\mathrm{r} \left(\cos(q_k) - 1\right) - p_k^2 Q_{\mathrm{r}(2,2)} - u_k^T P_\mathrm{r} u_k \tag{5-14}$$

where:

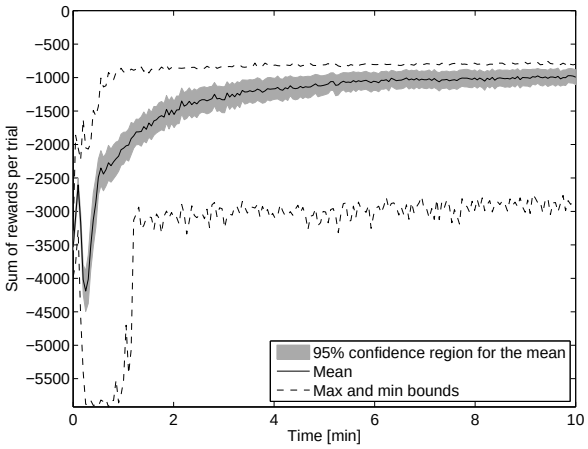
$$Q_\mathrm{r} = \begin{bmatrix} 5 & 0 \\ 0 & \frac{0.1}{J_\mathrm{p}^2} \end{bmatrix} \; , \; P_\mathrm{r} = 1 \; , \; W_\mathrm{r} = 25 \tag{5-15}$$

The scaling (5-15) was applied to be able to use the same learning rates in all reward types, which is because the reward contributes in an important way to the (size of the) temporal difference (4-18). The simulation parameters were the same as in Chapter 4 and are given in Table 4-2. The simulation results are given in Figure 5-4. The following observations can be made:

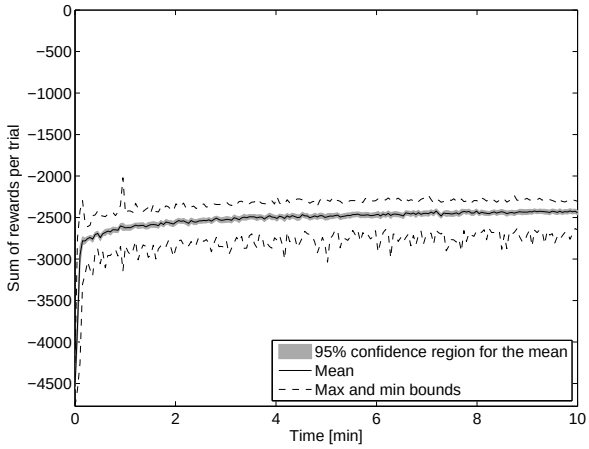
- It can be noticed from Figure 5-4c that simulations using reward r_C, (5-12), do not converge to the optimal policy, corresponding to an average cumulative reward of approximately -1200. Instead, convergence to a policy corresponding to an average cumulative reward of ≈ -2500 can be seen, which corresponds to the motion of swinging up the pendulum and then keep rotating about (hence it stays in Phase 3 but never learns to inject enough damping). This is logical, because there is no penalty on the momentum in r_C and if the system keeps rotating fast enough it receives the maximum reward each time it passes the desired equilibrium



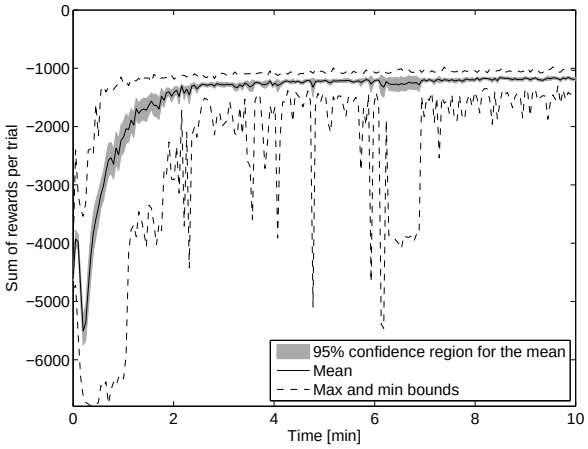
(a) r_{A}



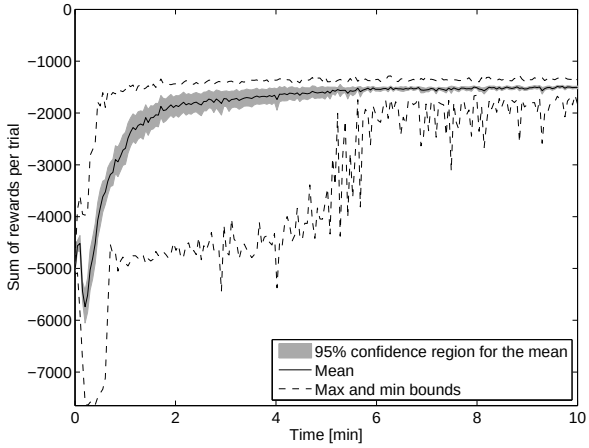
(b) r_{B}



(c) r_{C}



(d) r_{D}



(e) r_{E}

Figure 5-4: Results for the EBAC method for 50 learning simulations for 5 different reward types. As can be seen, r_{D} shows the best performance in terms of learning speed and consistency.

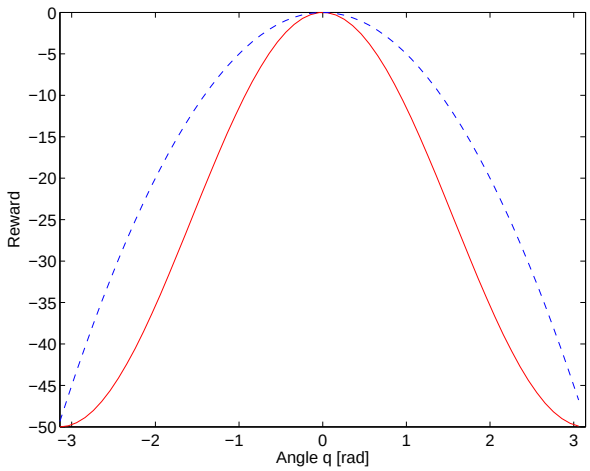


Figure 5-5: Difference between instantaneous position reward from r_{B} (blue dashed) and r_{D} (red solid) with $p = 0$. The larger steepness of the cosine (red solid) explains the better performance of r_{D}.

position at the top. Hence, it is very hard using this reward type for the algorithm to learn that it is better to swing up and stay at the desired equilibrium.

- Using r_{A} and r_{B} (Figures 5-4a–5-4b) the system sometimes does not converge to the optimal policy. This is caused by the relatively fast learning rates, and setting these values lower would probably eliminate those 'non-convergent' trials. However, since learning would then be slower and other reward types do not suffer from this problem, this is deemed not necessary.

- Reward r_{D} and r_{E} show very fast and good convergence. The addition of penalizing the control action in r_{E} has a negative effect on learning in Phase 3, because it can be seen that the increase in reward in Phase 3 in Figure 5-4e is less fast than with r_{D} (Figure 5-4d), which shows a faster increase in reward in that phase. This can be explained by the fact that penalizing the control actions causes the algorithm to learn not to increase the control action too much (of course), which has the effect that learning becomes slower.

- The difference between r_{B} and r_{D} is only in either quadratically penalizing the position (r_{B}) or by using a cosine (r_{D}). The instantaneous position reward for all q for both reward types is given in Figure 5-5. When compared for a single position, the reward is always smaller (more negative) for r_{D} and r_{D} generally increases faster (i.e. the red solid graph is steeper upwards for most q). This explains why r_{D} shows slightly better performance than r_{B}, because if the graph is steeper, each update in the direction of a reward closer to the equilibrium is larger than if it is less steep, as in r_{B}.

- The simulation parameters were identical for all reward types, but optimal for reward r_{D}. However, the only way to eliminate the non-converging simulations in r_{A} and r_{B} is to decrease the learning rates; hence the average learning speed will become even less than it is now. Also, because there are only a few (i.e. 5 for r_{A}

and 2 for r_{B}) non-convergent simulations, the effect of these simulations on the average learning speed is not substantial. Therefore, eliminating these simulations still would not yield a better learning curve in terms of speed for those types than for r_{D} or r_{E}.

- Reward r_{D} performs the best in terms of consistency in the sense that there is the least deviation from the average, which means that most simulations show behaviour close to the average.

Based on this simulation experiment it can be concluded that reward r_{D} is the best choice, hence this simulation and Figure 5-4d will be used as a reference in subsequent sections. It can also be concluded that the algorithm performs quite well in terms of speed of learning with all the reward types, although some simulations do not converge to the optimal solution, which can be solved by choosing lower learning rates.

5-1-4 Control Saturation

An important novelty of the EBAC method is to incorporate control saturation in PBC for PH systems, which was previously done only for Euler-Lagrange systems in e.g. [13, 30], in which there was no form of adaptation or learning present. In this section, different saturation functions are used in simulations as well as a different bound on the input, in order to investigate the influence of control saturation in the EBAC method.

Saturation Function

In Chapter 4, the saturation function (4-67) was used:

$$\varsigma(u_k) = \begin{cases} u_k & \text{if } |u_k| \leq u_{\max} \\ \operatorname{sgn}(u_k) u_{\max} & \text{otherwise} \end{cases} \tag{5-16}$$

In this section the effect of changing the saturation is studied, which is relevant because the derivative of the saturation function is incorporated in the policy gradient and hence has an important effect on learning (recall (4-37)). Because the saturation function (5-16) has derivative:

$$\frac{\partial \varsigma(u_k)}{\partial u_k} = \begin{cases} 1 & \text{if } \varsigma(u_k) = u_k \\ 0 & \text{otherwise} \end{cases} \tag{5-17}$$

which indicates a non-smooth transition from saturated to non-saturated inputs, it is interesting to study the effect of saturation functions that have a smooth derivative to see whether this has a positive effect on learning. Specifically, *arctangent* and *hyperbolic tangent* saturation were used. Arctangent (using subscript 'a') saturation can be given by:

$$\varsigma_{\mathrm{a}}(u_k) = \frac{2u_{\max}}{\pi} s_{\mathrm{a}} \arctan\left(s_{\mathrm{a}} u_k\right) \tag{5-18}$$

$$\frac{\partial \varsigma_{\mathrm{a}}(u_k)}{\partial u_k} = \frac{2u_{\max}}{\pi} \frac{s_{\mathrm{a}}}{s_{\mathrm{a}}^2 u_k^2 + 1} \tag{5-19}$$

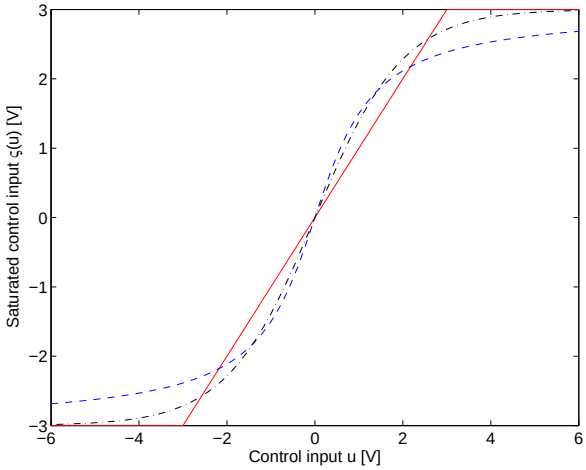


Figure 5-6: Different saturation types considered: plain (red solid), arctangent (blue dashed), hyperbolic tangent (black dashed-dotted). Every input u is saturated between the bounds $(-3, 3)$. Arctangent and hyperbolic tangent show smooth behaviour in the transition between u and u_{sat}.

with s_{a} the skewness of the arctangent function. The hyperbolic tangent (using subscript 't') saturation reads:

$$\varsigma_{\text{t}}(u_k) = u_{\max} s_{\text{t}} \tanh\left(s_{\text{t}} u_k\right) \tag{5-20}$$

$$\frac{\partial \varsigma_{\text{t}}(u_k)}{\partial u_k} = u_{\max} s_{\text{t}} - u_{\max} s_{\text{t}} \tanh^2\left(s_{\text{t}} u_k\right) \tag{5-21}$$

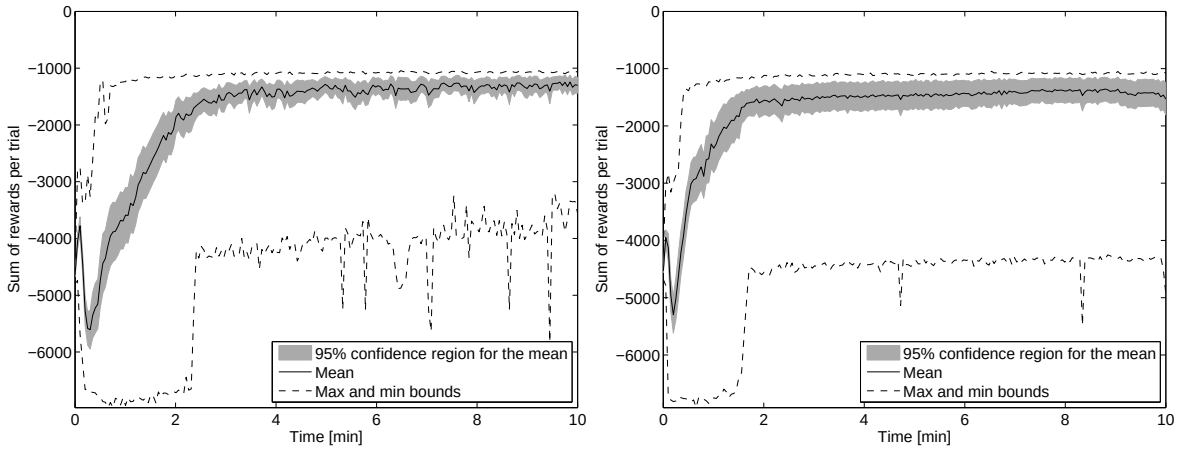
with s_{t} the skewness of the hyperbolic tangent function. For simulation, the following values were chosen:

$$s_{\text{a}} = 1 \tag{5-22}$$

$$s_{\text{t}} = 0.5 \tag{5-23}$$

and $u_{\max}$ and all other simulation parameters as in Table 4-2. The reward type was r_{D}. The saturation functions are given in Figure 5-5 from which the smooth behaviour of the arctangent and hyperbolic tangent saturation can be seen. The results for 50 simulations are given in Figures 5-7a–5-7b.

- Arctangent saturation shows slower convergence speed than using plain saturation (compare with Figure 5-4d) which is logical because the saturation is 'harder', which means that it can be seen from Figure 5-6 that the full control action of 3V is not reached in the displayed interval and the increase is small for larger u. This is reflected in the learned policies for arctangent saturation; due to the fact that a policy gradient is in practice always available ((5-19) $= 0$ only if $u_k \to \infty$) but the control u_k should be very large for (5-18) to return the full actuation $u_{\max}$, the algorithm learns a bang-bang policy for this type of saturation.
- The hyperbolic tangent shows much better results and converges to a good policy faster than using arctangent saturation, as can be seen in Figure 5-7b. The



(a) arctan saturation simulation results.

(b) tanh saturation simulation results.

Figure 5-7: Results for the EBAC method for 50 learning simulations for the arctangent (a) and hyperbolic tangent (b) saturation.

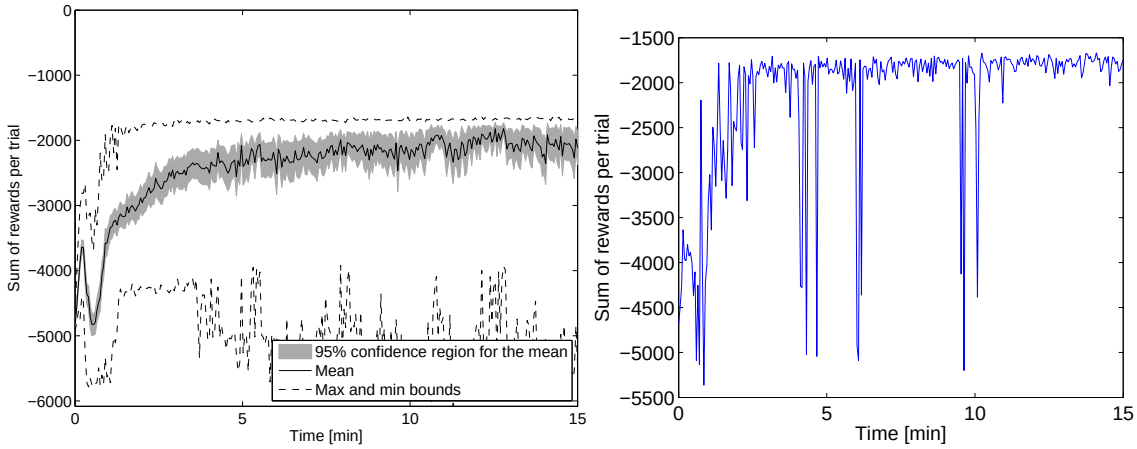
behaviour is more similar to that of Figure 5-4d and perhaps using this saturation learning is even faster.

- Both types of saturation show that there are some non-convergent trials, which can only be prevented by decreasing learning rates, thus the performance will be worse if it is required that all simulations converge to the optimal policy.

It can be concluded that none of the smooth saturation functions yield significantly better performance than when using the saturation function (5-16). For future work, it is interesting to study another saturation function, called smoothing minimum/maximum [3], because in this type of saturation, linear behaviour in the area of normal operation is achieved but, unlike the non-smooth behaviour of (5-16), a smooth transition is obtained between the area of normal operation and the area of saturation.

Input Bounds

Apart from changing the saturation function, it is also worth investigating the effect of changing the input bounds ($u_{\min}$, $u_{\max}$), because in practice, it is possible that an actuator has a narrower bound on the input than expected. If this is not known, e.g. the algorithm applies a certain saturated voltage but this is internally saturated again to a stricter bound due to component limitations or unknown hardware limitations, the algorithm might not be able to find a good policy. Therefore, 50 simulations were done using an input bound of ($u_{\min}$,$u_{\max}$) of $(-2, 2)$V using the simulation settings from Table 4-2 except for the number of trials, which was increased to 300. This is because the optimal policy will now consist of two swings back and forth, which takes more time during the trial and hence it is expected that it will also take more time to learn this policy. The results are given in Figure 5-8a. The (near) optimal policy is now obtained



(a) Results for the EBAC method for 50 learning simulation using a narrower input saturation bound.

(b) Results for the EBAC method for 1 learning simulation using a narrower input saturation bound.

Figure 5-8: Results for the EBAC method for 50 learning simulations using a narrower input saturation bound. Although the speed of learning seems to have declined significantly (a), this in mainly caused by the large deflections from the near optimal policy that sometimes occur, as can be seen from (b).

for an average cumulative reward of $r = -1700$. Although it appears from Figure 5-8a that most of the trials do not converge to the near optimal policy, this is actually not happening. This can be seen by looking at the progress of a single, representative trial in Figure 5-8b, from which it becomes clear that learning speeds are comparable to the simulations with the previously used input bounds, but after convergence there are some trials in which there is a sudden drop in performance. It turns out that all simulations experience this behaviour, such that the overall average of the simulations is much lower while each separate trial converges to the near optimal solution quickly. Hence, it can be concluded that using a narrower input bound, learning is still fast in the sense that still only 3 minutes of learning (60 trials) are necessary to reach a near optimal policy. After convergence, there are a number of trials in which there is a sudden drop in performance. These deflections from the near optimal policy are probably the effect of too high learning rates for this simulation, but these rates were intentionally held constant to study only the influence of the input bounds on learning speed and convergence.

5-1-5 Sensitivity

Since the EBAC method relies on incorporating a model in the learning algorithm, it is worth investigating the sensitivity to uncertain model parameters. Therefore, simulations were done using perturbed parameters in the policy (4-56) and the original parameters from Table 5-1 for simulating the pendulum dynamics. For simulations, the parameters from Table 4-2 were used once again with reward type r_{D}. The results are given in Figures 5-9a–5-9c where the average is given for 50 learning simulations of

each perturbed parameter. The simulations are compared with the average cumulative reward of Figure 5-4d.

- Mass: the mass M_p was changed with $\pm 20\%$, the simulation results of which are given in Figure 5-9a. Decreasing the mass (red solid) by 20% has an interesting effect: the algorithm is then able to swing up the mass immediately to the equilibrium position and hence it converges to that solution with a corresponding higher average cumulative reward. Decreasing by 20% (blue dashed) shows a significant impact on performance compared to the reference average cumulative reward. The algorithm needs to pump more energy into the system so as to swing up the increased mass. Because the initialization of the actors and critic remain the same as well as the learning rates, it takes more time to 'build up' the pumping-damping term $\hat{K}(x, \psi)$. Also, because the maximum allowed voltage remains constant, more maximum voltage actions will be necessary. These actions are difficult to learn, because (4-37) forces that there is no update of policy parameters if the policy exceeds the maximum allowed voltage - which will happen more frequently if it is increasingly necessary to approach that bound.
- Inertia: The inertia J_p was also changed with $\pm 20\%$, the simulation results of which are given in Figure 5-9b. Decreasing J_p by 20% (red solid) has no significant impact on performance. Increasing J_p by 20% (blue dashed) shows convergence to a policy corresponding to a lower average cumulative reward. This can be attributed to the fact that the reward r_D is dependent on J_p, as well as the bounds on the state space. However, it is not clear why this effect does not happen in the same manner when decreasing the inertia.
- Damping: The damping b_p was also changed with $\pm 20\%$, the simulation results of which are given in Figure 5-9c. Both increasing (blue dashed) and decreasing (red solid) the damping shows no significant impact on performance.

5-1-6 Initialization of Value Function

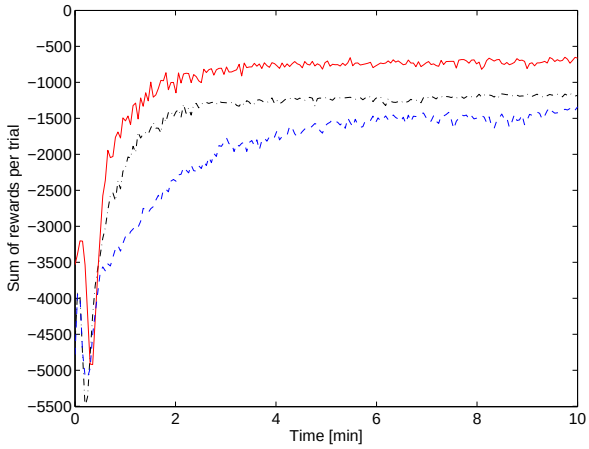
To prevent the initial drop in performance of Phase 2 (Figure 5-2) the value function $\hat{V}(x, \theta)$ can be more realistically initialized. Two examples of this were tested in simulation, the first with the worst possible initialization,

$$V_0(x) = \frac{1}{1-\gamma} \min_{x,u} r(x, u) \tag{5-24}$$

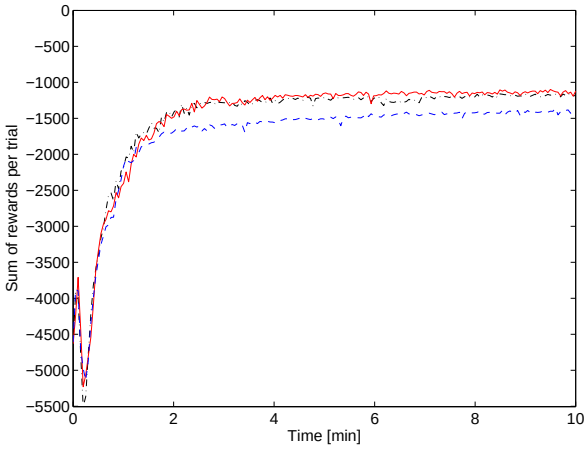
in which $r(x, u)$ is (5-13). The worst possible reward is then calculated by inserting $x = \begin{bmatrix} \pi & 8\pi \cdot J_\mathrm{p} \end{bmatrix}$ (the bounds on the state space for the function approximator) such that the worst (using subscript 'w') possible value function initialization yields:

$$V_{0,\mathrm{w}}(x) = -3.772 \cdot 10^3 \qquad \forall x \tag{5-25}$$

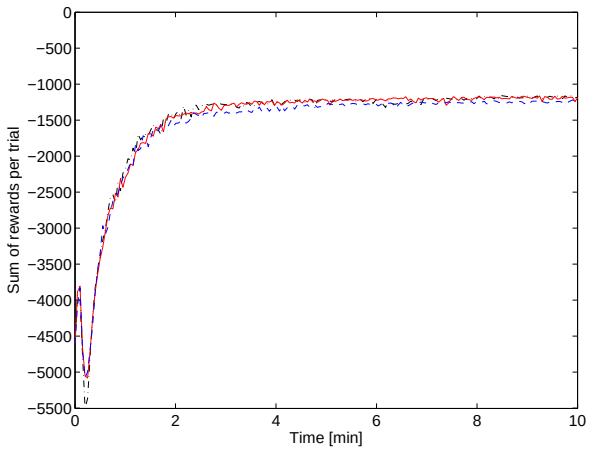
The second initialization was done using the minimum of the value function derived by inspecting the minima of several learned value functions (as in Figure 5-1). For the



(a) Mass M_p increased with 20% (blue dashed), decreased with 20% (red solid) and reference (black dashed-dotted).



(b) Inertia J_p increased with 20% (blue dashed), decreased with 20% (red solid) and reference (black dashed-dotted).



(c) Damping b_p increased with 20% (blue dashed), decreased with 20% (red solid) and reference (black dashed-dotted).

Figure 5-9: Results for the EBAC method for 50 learning simulations with perturbed parameters. Only increasing or decreasing the mass shows a significant impact on performance.

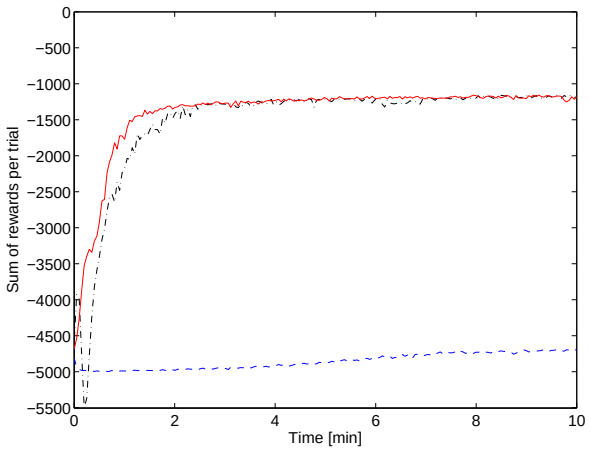


Figure 5-10: Value function initialization realistically (red solid, $V_{0,\mathrm{r}} = -1000$), worst possible (blue dashed, $V_{0,\mathrm{w}} = -3.772 \cdot 10^3$) and reference (black dashed-dotted, $V_0 = 0$). Realistic initialization eliminates the initial drop in performance and speeds up the algorithm.

simulation done with reward r_{D}, this minimum on average is approximately -1000. Hence, in the second initialization the value function is realistically (using subscript 'r') initialized with:

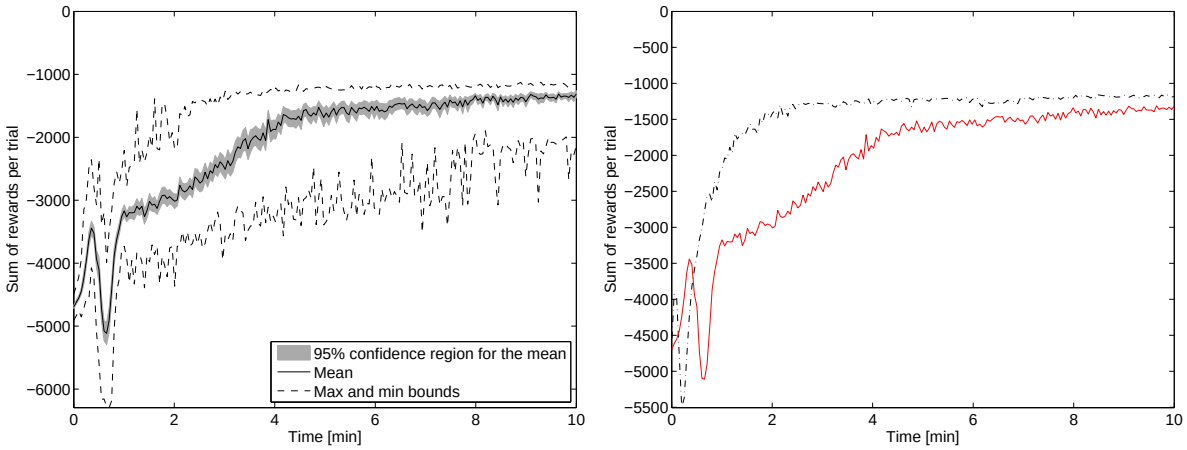
$$V_{0,\mathrm{r}}(x) = -1000 \qquad \forall x \tag{5-26}$$

The simulation parameters are as in Table 4-2 with reward r_{D}. The result for an average of 50 simulations with $V_{0,\mathrm{w}}$ and the average of 50 simulations with $V_{0,\mathrm{r}}$ is given in Figure 5-10 along with the reference average of Figure 5-4d. As can be seen, initializing the algorithm with the worst possible reward makes it impossible to learn in the given time frame. Initializing realistically with $V_{0,\mathrm{r}}$ indeed eliminates the initial drop in performance and hence even further speeds up the algorithm. The difference between when convergence is reached when compared to the reference is approximately 5 trials, corresponding to 30 seconds of learning. Hence, initializing the value function with a proper value improves performance, but this can only be done by trial and error which makes it hard to estimate a good value to initialize with.

5-1-7 Cost Function Gradient

For the update of the two actors, a heuristic estimate of the cost function gradient $\nabla_\vartheta J(x_k)$ is used from the S-AC algorithm (See (3-30)). In [20], a model-based cost function gradient estimate is used when a model is available[2]. Because in EBAC there is a model available, it is possible that using this cost function gradient estimate yields better performance than using the heuristic estimate. Therefore, a simulation was done

[2]In the Model Learning Actor-Critic (MLAC) method [20].



(a) Model cost function gradient simulation results.

(b) Comparison of average of (a) with average of heuristic cost function gradient estimate.

Figure 5-11: Results for the EBAC method for 50 learning simulations with model-based cost function gradient estimate (a) and comparison with reference average of Figure 5-4d using the heuristic cost function gradient estimate. As can be seen, the heuristic cost function gradient estimate performs better.

with the following cost function gradient estimates:

$$\begin{aligned}\nabla_\xi J(x_k) &\approx \nabla_x V_\theta(x_k)^T \nabla_u f(x_k,u_k) \nabla_{\hat{\pi}} \varsigma(\hat{\pi}(x,\xi,\psi)) \nabla_\xi \hat{\pi}(x_k,\xi,\psi) \\ &= \theta^T \begin{bmatrix} \frac{\partial \phi_c(x)}{\partial q} & \frac{\partial \phi_c(x)}{\partial q} \end{bmatrix}^T g \nabla_{\hat{\pi}} \varsigma(\hat{\pi}(x,\xi,\psi)) \nabla_\xi \hat{\pi}(x_k,\xi,\psi) \qquad (5\text{-}27) \\ \nabla_\psi J(x_k) &\approx \nabla_x V_\theta(x_k)^T \nabla_u f(x_k,u_k) \nabla_\psi \hat{\pi}(x_k,\xi,\psi) \\ &= \theta^T \begin{bmatrix} \frac{\partial \phi_c(x)}{\partial q} & \frac{\partial \phi_c(x)}{\partial q} \end{bmatrix}^T g \nabla_{\hat{\pi}} \varsigma(\hat{\pi}(x,\xi,\psi)) \nabla_\psi \hat{\pi}(x_k,\xi,\psi) \qquad (5\text{-}28)\end{aligned}$$

with $\nabla_u f(x_k,u_k)$ the partial derivative to u of the PH system (4-47), yielding the input matrix g, and $\nabla_{\hat{\pi}} \varsigma(\hat{\pi}(x,\xi,\psi))$ the partial derivative of the saturation function (5-17). An advantage of (5-27)-(5-28) is that no exploration is necessary. However, since the actors are initialized with zeros, exploration is at least necessary at the first update of the parameters because otherwise the parameters will not change. Also, without exploration it is possible that not enough parts of the state space are visited, as Grondman et al. point out in [20]. The simulation parameters are as in Table 4-2 with reward r_D. The results are given in Figure 5-11a. Clearly the algorithm shows slower learning behaviour using the model based cost function gradient estimate (red solid) when compared to the reference (black dashed-dotted) of Figure 5-11b. To investigate the reason for this, the size of the cost function gradient is investigated for one simulation using the model cost function gradient estimate (5-27)-(5-28) and using the heuristic cost function gradient estimate (4-23) as a reference . Therefore, the 2-norm of the cost function gradient estimate for each visited state for a single trial is calculated via:

$$b_{es} = \|\nabla_\xi J(x(j))(i)\|_2 \qquad (5\text{-}29)$$
$$b_{di} = \|\nabla_\psi J(x(j))(k)\|_2 \qquad (5\text{-}30)$$

where $i = 1, \ldots, 4$ (see Section 4-6-2) for the energy shaping parameter vector, $k = 1, \ldots, 16$ for the damping injection parameter vector and $j = 1, \ldots, 100$ for each visited state per trial such that $b_{\text{es}} \in \mathbb{R}^{c \times 4}$ and $b_{\text{di}} \in \mathbb{R}^{c \times 16}$, $c = 200$. Then, if we calculate b for each trial and sum the norms, it is possible to study the evolution of the size of the cost function gradient for each trial,

$$B_{\text{h,es}} = \sum_{i=1}^{4} [b_{\text{h,es}}]_{c,i} \tag{5-31}$$

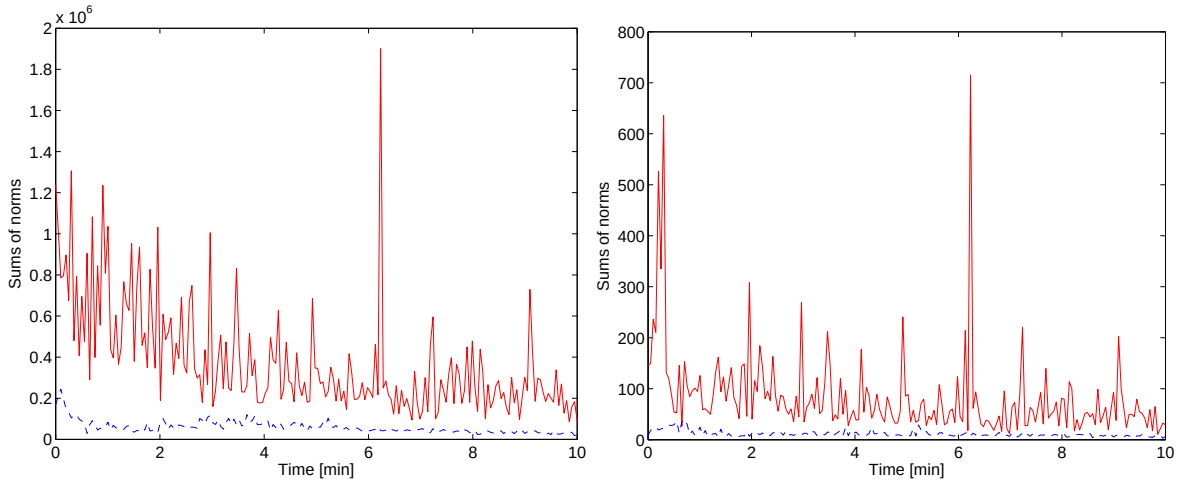
$$B_{\text{h,di}} = \sum_{i=1}^{16} [b_{\text{h,di}}]_{c,i} \tag{5-32}$$

$$B_{\text{m,es}} = \sum_{i=1}^{4} [b_{\text{m,es}}]_{c,i} \tag{5-33}$$

$$B_{\text{m,di}} = \sum_{i=1}^{16} [b_{\text{m,di}}]_{c,i} \tag{5-34}$$

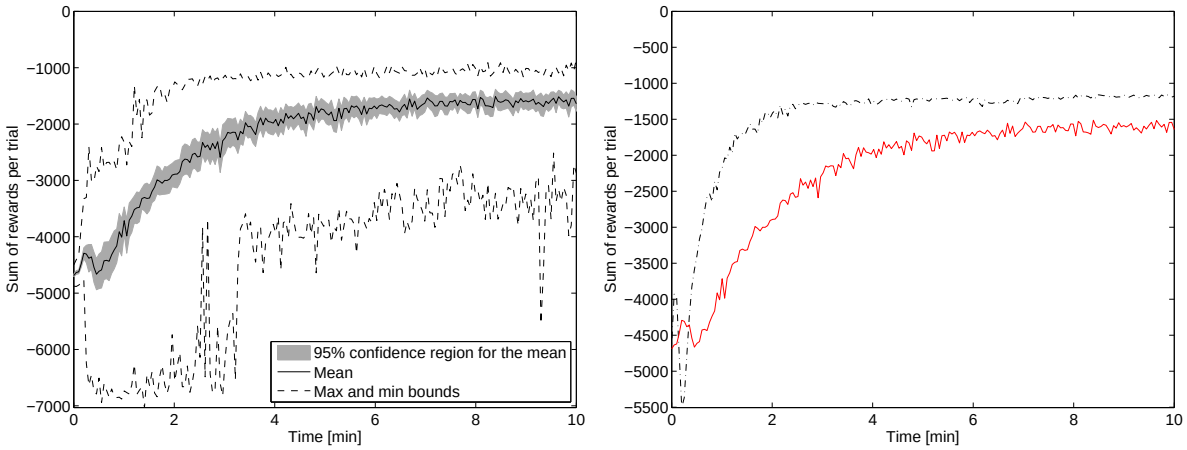
where B denotes the vector of sums of norms of cost function gradient estimates and the subscript h, m denotes the heuristic cost function gradient estimate and model cost function gradient estimate, respectively.

Figure 5-12 displays the result of (5-31)-(5-34). In both Figure 5-12a and in Figure 5-12b it can be seen that the size of the heuristic cost function gradient estimate vector B is much larger than that of the model cost function gradient estimate. This can explain the faster learning behaviour using the heuristic cost function estimate: each update is larger, which means that it can be expected that the algorithm learns faster. On the other hand, updating with large gradients could also result in converging to non-optimal solutions or no convergence at all. The reason that this does not happen is that it turns out that the heuristic cost function gradient estimate is often zero caused by saturation (recall (4-37) and (5-17)). However, when the gradient is not zero, the updates are very large and - apparently - in the correct direction. The model policy cost function gradient estimate however is smaller and thus does not update very fast to a policy that gets saturated. Hence, there are more updates but these updates are also smaller. Also, note that both gradients show convergence: they both decrease in size over time. It is beyond the scope of this thesis to examine the policy gradients any further, but in future research it is recommended to investigate the evolution and direction of a cost function gradient estimate, since this greatly determines learning speed. In this simulation example, it turns out that the heuristic cost function gradient estimate is better due to its nature of high values that are less used as a result of saturation, compared to a model cost function gradient estimate where the values are more frequently used but are of less size. Also, this simulation of course depends on the initialization of the actors and critic, which also determines the size of the cost function gradient estimate. Finally, it is recommended that in future research other types of cost function gradients are investigated, such as the Natural gradient [47].



(a) $B_{\mathrm{h,es}}, B_{\mathrm{m,es}}$ **(b)** $B_{\mathrm{h,di}}, B_{\mathrm{m,di}}$

Figure 5-12: Results for the EBAC method for 1 learning simulation consisting of 200 trials using the heuristic cost function gradient estimate and the model cost function gradient estimate. As can be seen, the size of the heuristic cost function gradient estimate (red solid, both graphs) is larger than the size of the model cost function gradient estimate (blue dashed, both graphs) both for the energy shaping actor gradient (a) as for the damping injection actor gradient (b).



(a) Non-symmetrical Fourier Basis simulation results.

(b) Average of (a) compared to reference average.

Figure 5-13: Results for the EBAC method for 50 learning simulations using a non-symmetrical Fourier Basis (a) and comparison (b) with reference average of Figure 5-4d. Performance has degraded significantly.

5-1-8 Basis Functions

In Section 4-6-1, the state x was scaled according to (4-61). In this section, the effect of projecting the variables onto a non-symmetrical range,

$$(\bar{x}_{i,\min}, \bar{x}_{i,\max}) = (0, 1) \tag{5-35}$$

in (4-61), is investigated. Then, it is possible to draw conclusions about the effect of incorporating model knowledge by using the 'symmetrical' range as in Section 4-6-1. Thus, 50 simulations were done using the non-symmetrical Fourier Basis with simulation settings all equal to those in Table 4-2 except for the learning rate $\alpha_{\mathrm{a_b},\psi}$, which was chosen:

$$\alpha_{\mathrm{a_b},\psi} = 0.1 \tag{5-36}$$

Furthermore, more functions are now necessary to represent the actor and critic. Thus, actor 1 ($\hat{P}_d(q,\xi)$) is parameterized using a 4^{rd}-order Fourier basis in the range $[-\pi\ \pi]$ resulting in 5 learnable parameters, and actor 2 ($\hat{K}(x,\psi)$) is parameterized using a 4^{rd}-order Fourier basis for the full state space in the same domain as for the critic resulting in 25 learnable parameters. The critic was identically parameterized as actor 2. The results for 50 learning simulations are given in Figure 5-13. As can be seen, learning is considerably slower than in the reference of Figure 5-4d, of which the average is displayed in Figure 5-13b. Hence, it appears that the symmetrical projection of the state x is the main cause of fast learning in the EBAC method - or at least the combination of the symmetrical projection of the Fourier Basis with the parameterized EB-PBC control law. Thus, it can be concluded that it is important to incorporate model knowledge in the basis functions - if possible. Also, tests were done using Radial Basis Functions (RBF)[3] but that did not deliver good results. This can be attributed to the fact that it is important to select basis functions that have a well-behaving derivative, considering parameterization (4-29). For example, in the Fourier Basis, the derivative of the cosine function basis is a sine, which is naturally a good choice for the gradient of the desired potential energy in the inverted pendulum set-up. Radial Basis Functions do not show this nice behaviour (i.e. the gradient is much smaller which causes a need to initialize the actors at a non-zero value to facilitate fast learning) which makes them less suitable. For future research, it is recommended to try other types of basis functions, where it can be expected that a Polynomial Basis [26], which has a well-behaving derivative for a gradient of potential energy, will show good results.

5-1-9 Comparison with Standard Actor-Critic

In this section, a comparison with the Standard Actor-Critic method from [20], given in Algorithm 3.1, is made such that it is possible to put the performance of EBAC in line with currently available and well-known methods. To compare the EBAC method with the S-AC method, we define a 6^{th}-order Fourier Basis for both the actor (3-22) and the critic (3-23) resulting in 49 learnable parameters for the policy and value function, respectively. The state-variables are projected onto:

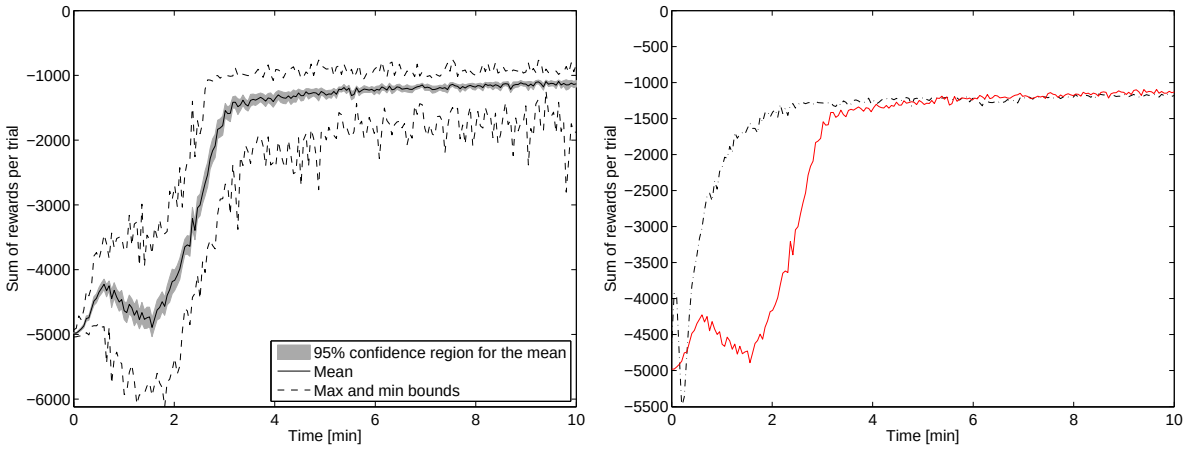
$$(\bar{x}_{i,\min}, \bar{x}_{i,\max}) = (0, 1) \tag{5-37}$$

using (4-61), such that an arbitrary nonlinear function can be approximated by the Fourier Basis and no model knowledge is incorporated. The simulation parameters are given in Table 5-2 and the reward r_{D} (5-13) is used. Saturation is incorporated using

[3] For an explanation, see e.g. [18].

Table 5-2: Simulation parameters for Standard Actor-Critic

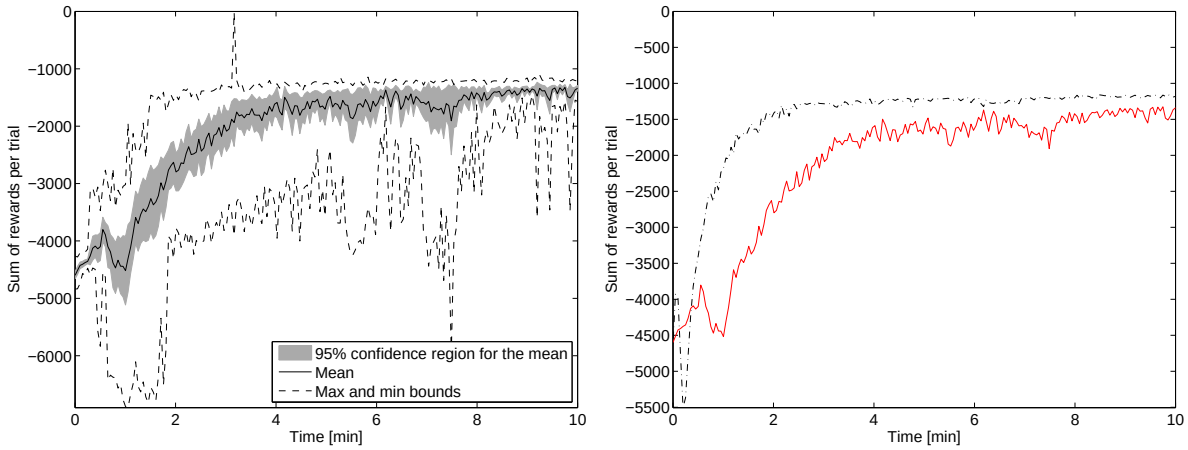
Simulation parameters	Symbol	Value	Units
Trials	–	200	-
Time per trial	T_{t}	3	s
Sample time	T_{s}	0.03	s
Decay rate	γ	0.97	-
Eligibility trace	λ	0.65	-
Exploration variance	σ^2	1	-
Max torque	$u_{\max}$	3	V
Learning rate critic	α_{c}	0.005	-
Learning rate actor	α_{a}	0.0001	-



(a) S-AC 50 simulations.

(b) Average of (a) compared to reference EBAC average.

Figure 5-14: Results for the S-AC method for 50 learning simulations using the normal Fourier Basis (a) and comparison (b) of average (red solid) to reference average (black dashed-dotted) using the symmetrical Fourier Basis of Figure 5-4d. As can be seen, the EBAC method outperforms S-AC by converging to a good policy faster, i.e. 50 trials instead of 100 trials.



(a) Results for the EBAC method for 20 learning experiments.

(b) Average of (a) compared to reference simulation average.

Figure 5-15: Results for 20 learning experiments on the DCSC set-up (a) and comparison (b) of average (red solid) to reference average (black dashed-dotted) of Figure 5-4d. Learning is considerably slower than in simulation, on average at least 80 trials (4 minutes of learning) are necessary to obtain a good policy.

(4-67) and all parameter vectors were initialized with zeros of appropriate size. Each simulation that consists of 200 trials of 3 seconds of learning was repeated 50 times. The results are given in Figure 5-14. It can be seen that the S-AC method learns considerably slower, compared to the reference using the EBAC method of Figure 5-4d of which the average is compared to S-AC in Figure 5-14b. However, it must be noted that tests showed that it is possible to get initial learning speeds equivalent to those obtained when using the EBAC method, but this resulted in slow overall convergence and policies that were of less quality. Finally, it can be seen that the performance of S-AC is in line with the results from [18] where Local Linear Regression (LLR) is used as a basis, although the Fourier Basis used here only needs 49 parameters both for the actor and critic.

5-2 Inverted Pendulum - Experiments

To verify the developed method on a real set-up, the DCSC pendulum was used, of which a picture is depicted in Figure 4-1. The equations admit (5-8) and the model parameters are as in Table 4-1. Of course, these model parameters are now only used for calculating the control action via policy (4-56).

5-2-1 Comparison with Simulation

To compare the real-life experiments with the data obtained from simulation, the parameters from Table 4-2 were used to perform 20 learning experiments using the EBAC

method on the DCSC set-up. The results are given in Figure 5-15a. It can be seen that learning is considerably slower than in the reference simulation of Figure 5-4d, of which the average is compared to the average obtained in experiments in Figure 5-15b. This has the following reasons:

- Model mismatch. In the physical set-up, dry friction is present and this is not present in the model. This decreases learning speed, which can be seen by looking at the update for the desired damping $\hat{K}(x,\psi)$, (4-58):

$$\begin{aligned}\psi_{k+1} &= \psi_k + \alpha_{a,\psi}\delta_{k+1}\Delta u_k \nabla_\psi \hat{\pi}(x_k,\xi,\psi) && (5\text{-}38)\\ &= \psi_k + \alpha_{a,\psi}\delta_{k+1}\Delta u_k \nabla_{\hat{\pi}}\varsigma(\hat{\pi}(x,\xi,\psi))\nabla_\psi \hat{\pi}(x,\xi,\psi) && (5\text{-}39)\\ &= \psi_k + \alpha_{a,\psi}\delta_{k+1}\Delta u_k \nabla_{\hat{\pi}}\varsigma(\hat{\pi}(x,\xi,\psi))\nabla_x \phi_K(x) g \nabla_x \hat{H}_\mathrm{d}(x,\xi) && (5\text{-}40)\\ &= \psi_k + \alpha_{a,\psi}\delta_{k+1}\Delta u_k \nabla_{\hat{\pi}}\varsigma(\hat{\pi}(x,\xi,\psi))\nabla_x \phi_K(x)\dot{q} && (5\text{-}41)\end{aligned}$$

  Hence, the update of ψ depends on the velocity $\dot{q}$. In the beginning of learning, the system will have to 'build up' velocity as fast as possible in order to facilitate fast learning of this parameter (given the zero initialization of the parameters). If dry friction is present, there is much less overshoot in the beginning (this will be seen later on in a step response). Hence, the update is smaller and the system has trouble learning the pumping-damping term $\hat{K}(x,\psi)$ and, with too high learning rate $\alpha_{\mathrm{a},\xi}$, will converge to the undesirable wells of $\hat{P}_\mathrm{d}(q,\xi)$. It is possible to increase the learning rate $\alpha_{\mathrm{a},\psi}$ and decrease the learning rate $\alpha_{\mathrm{a},\xi}$, but that is not without consequences. First of all, further increasing $\alpha_{\mathrm{a},\psi}$ will result in fast learning of pumping energy (i.e. fast decrease in $\hat{K}(x,\psi)$), but slow (or none at all) learning of injecting damping (increase of $\hat{K}(x,\psi)$). In other words, the system will remain in Phase 3 of learning. Decreasing $\alpha_{\mathrm{a},\xi}$ creates more time for the pumping-damping term to learn the swing-up, but then the risk is that the position control, which is based on ξ, becomes very bad (i.e. too low gain). This means that at the top, perturbing the equilibrium slightly will result in an offset and none or extremely slow position feedback. Hence, there is a careful balance between learning the swing-up fast enough to escape the undesirable wells of $\hat{P}_\mathrm{d}(q,\xi)$ and having a good position feedback at the top.

- Parameter mismatch. The parameters used in the control law are probably not exactly equal to the true parameters of the system. However, earlier simulations showed the robustness of the method to parameter variations. Hence, unless the parameters deviate from those in Table 4-1 by an extremely large amount (e.g. $\gg 20\%$), slower learning performance is not likely the result of parameter mismatch.

- Momentum estimation. In the experiment, the momentum is calculated using a simple Euler backward difference method:

$$p_k \approx \frac{(q_k - q_{k-1})J_p}{T_\mathrm{s}} \qquad (5\text{-}42)$$

  which can be replaced by a better method, for example an observer.

These three issues will be addressed in the next sections.

5-2-2 Identified Model

As said in the previous section, dry friction is probably one of the reasons causing a decrease in learning speed. Thus, it is interesting to study the effects of dry friction by incorporating it in the PH model (4-47). To do so, a simple Coulomb friction term is considered:

$$F_f = \sigma_{\mathrm{p}} \operatorname{sgn}(\dot{q}) \tag{5-43}$$

with σ_{p} the coefficient of friction. Equation (5-43) indicates positive (negative) dry friction for positive (negative) velocities. It is possible to rewrite (5-43) as:

$$F_f = \sigma_{\mathrm{p}} \frac{\dot{q}}{|\dot{q}|} \tag{5-44}$$

where in simulation, the pseudo-inverse is applied to the denominator of (5-44) such that division by zero is avoided for $\dot{q} = 0$. This equation can then be directly inserted in the PH system (4-47) by adding it to the damping term, such that the new PH model for the pendulum reads:

$$\Sigma_{\text{pend}} : \begin{cases} \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -\left(b_{\mathrm{p}} + \frac{K_{\mathrm{p}}^2}{R_{\mathrm{p}}}\right) + \frac{\sigma_{\mathrm{p}}}{\left|\frac{p}{J_{\mathrm{p}}}\right|} \end{bmatrix} \begin{bmatrix} \nabla_q H(q,p) \\ \nabla_p H(q,p) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_{\mathrm{p}}}{R_{\mathrm{p}}} \end{bmatrix} u \\ y = \begin{bmatrix} 0 & \frac{K_{\mathrm{p}}}{R_{\mathrm{p}}} \end{bmatrix} \begin{bmatrix} \nabla_q H(q,p) \\ \nabla_p H(q,p) \end{bmatrix} \end{cases} \tag{5-45}$$

with Hamiltonian still (4-49). Compare (5-8)-(5-45) and note how the damping term has changed with the addition of the dry friction term. Since the size of σ_{p} is unknown, an identification can be done using the physical set-up. By studying the model (4-47), all the relevant parameters can be replaced by a vector of parameters:

$$\mu = \begin{bmatrix} M_{\mathrm{p}} l_{\mathrm{p}} g_{\mathrm{p}} & J_{\mathrm{p}} & b_{\mathrm{p}} & K_{\mathrm{p}} & R_{\mathrm{p}} & \sigma_{\mathrm{p}} \end{bmatrix}^T \tag{5-46}$$

$$= \begin{bmatrix} \mu_1 & \mu_2 & \mu_3 & \mu_4 & \mu_5 & \mu_6 \end{bmatrix}^T \tag{5-47}$$

such that the PH system of the pendulum reads:

$$\Sigma_{\text{pend}} : \begin{cases} \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -\mu_3 + \frac{\mu_4^2}{\mu_5} + \frac{\mu_6}{\left|\frac{p}{\mu_2}\right|} \end{bmatrix} \begin{bmatrix} \nabla_q H(q,p) \\ \nabla_p H(q,p) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\mu_4}{\mu_5} \end{bmatrix} u \\ y = \begin{bmatrix} 0 & \frac{\mu_4}{\mu_5} \end{bmatrix} \begin{bmatrix} \nabla_q H(q,p) \\ \nabla_p H(q,p) \end{bmatrix} \end{cases} \tag{5-48}$$

with Hamiltonian:

$$H(q,p) = \frac{p^2}{2\mu_2} + \underbrace{\mu_1(1+\cos q)}_{P(q)} \tag{5-49}$$

An identification is then performed by applying a multisinusoidal input[4] using 10 sinusoids between the bounds $(-3, 3)V$ to both the model and the physical set-up. The first 5 input values (from $T = 0$ until $T = 0.15$s) are set to zero to prevent errors caused by initializing the physical set-up. Then, the parameters μ of the model can be fitted to match the multisine response data of the physical set-up by performing a minimization of the scalar quadratic error e_μ:

$$e_\mu = (\boldsymbol{q} - \boldsymbol{q}_{\text{ref}})^T(\boldsymbol{q} - \boldsymbol{q}_{\text{ref}}) \tag{5-50}$$

where $\boldsymbol{q}$ denotes the angle response trajectory data for the model and $\boldsymbol{q}_{\text{ref}}$ denotes the angle reference response trajectory data obtained from the physical system when the same input is applied. The momentum is intentionally left out of the error (5-50). This is because identification tests show that the velocity estimation using the Euler difference does not a produce a good model, while if the velocity estimation is left out of the identification, the identified model approximates the physical set-up better both in terms of position and momentum. The minimization of (5-50) is performed using the Nelder-Mead algorithm [29] with the following initial conditions[5]:

$$\mu_0 = \begin{bmatrix} M_{\text{p}} l_{\text{p}} g_{\text{p}} & J_{\text{p}} & b_{\text{p}} & K_{\text{p}} & R_{\text{p}} & 0 \end{bmatrix}^T \tag{5-51}$$

with the value of the known parameters as in Table 4-1. After minimization, the following estimate of the model parameters is obtained:

$$\hat{\mu} = \begin{bmatrix} 0.0223 & 1.9379 \cdot 10^{-4} & 3.0882 \cdot 10^{-6} & 0.0546 & 9.3762 & 6.2884 \cdot 10^{-4} \end{bmatrix}^T \tag{5-52}$$

A summary of the identification results is given in Table 5-3. A graph of the multisine identification and a validation using a step-response for a simulation with the initial values μ_0, for the response of the physical system and for a simulation with the identified parameters $\hat{\mu}$ is given in Figure 5-16[6]. It can be seen from Figure 5-16 that the identified model (blue dashed) is a better representation of the physical set-up (red solid) than the original model (black dashed-dotted), although there is still a small steady-state offset present. What immediately becomes apparent from Table 5-3 is that the original model parameters do not deviate from the identified parameters significantly (in the range of 5%), apart from the dynamic friction term b_p, which is probably now lower at the cost of the dry friction term. Hence, it can be concluded that the dominating difference between the simulated model response and physical setup response is indeed caused by the dry friction. Thus, incorporating this dry friction results in a better model, as is illustrated in Figure 5-16. Therefore, this model was used for Chapter 4[7]. For future research, it would be interesting to incorporate friction by building a friction

[4] A figure of this input is given in Appendix B.

[5] The assumption made here is that the identified (optimal) parameters will be in the range or close to the original parameters (apart from the dry friction term σ_{p}).

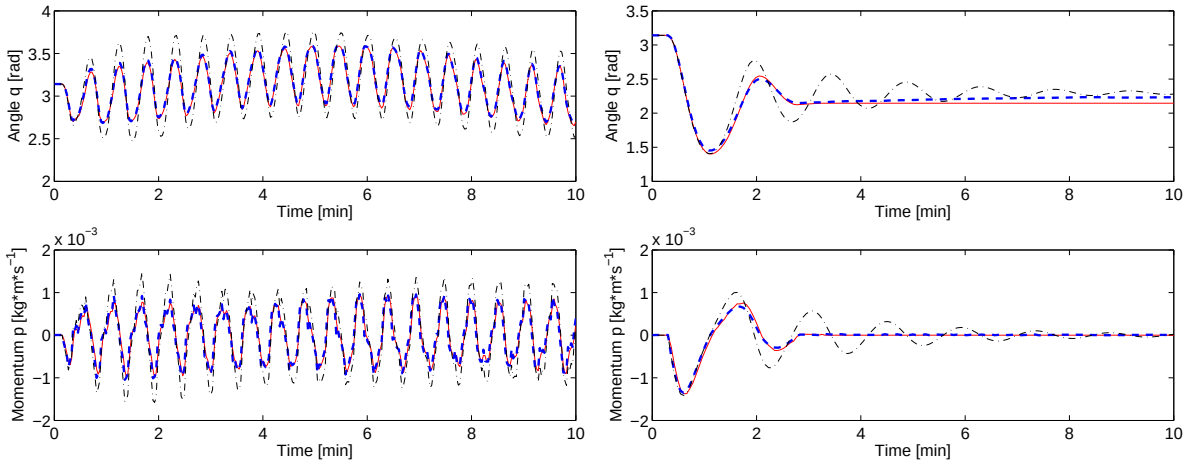
[6] It should be noted that for the error minimization, the trajectory data is wrapped to 2π to avoid wrapping problems, hence the initial position is π instead of zero in Figure 5-16.

[7] It should be noted that, since the parameters M_{p}, l_{p} and g_{p} cannot be identified separately, the identified parameter $\hat{\mu}_1$ is used to recalculate the identified mass where the assumption is made that the length of the pendulum and gravity remain constant. This explains the difference in value of M_{p} in Table 4-1 compared to the value in Table 5-1, while l_{p} and g_{p} remain constant in both these tables.

Table 5-3: Pendulum parameter identification results

Parameter	Model	Identified	% deviation
μ_1	0.0227	0.0214	-5.7%
μ_2	$1.9100 \cdot 10^{-4}$	$1.8970 \cdot 10^{-4}$	0.68%
μ_3	$3 \cdot 10^{-6}$	$2.4795 \cdot 10^{-6}$	-17.4%
μ_4	0.0536	0.0560	4.5%
μ_5	9.5	9.9233	-4.46%
μ_6	–	0.0010	–

observer and incorporate that in the PH framework, as has been done in [48] (dynamic friction compensation in the IDA-PBC framework), [8] (friction compensation for underactuated mechanical systems admitting (4-41)) and [27] (incorporating the LuGre friction model in the PH framework).



(a) Multisine identification **(b)** Step response validation

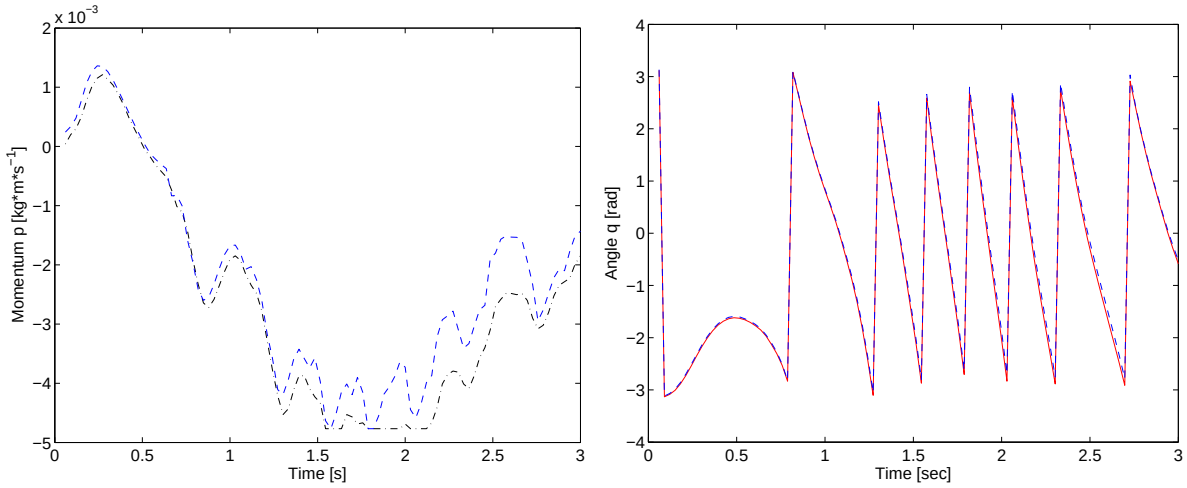
Figure 5-16: Identification results of pendulum parameters μ. As can be seen, the identified model (blue dashed) fits the step-response trajectory from the physical system (red solid) better than the initial model parameters (black dashed-dotted). There is a slight steady-state offset in the step response still present (b, top).

5-2-3 Model Momentum Estimation

Instead of using the Euler difference momentum estimation of (5-42), the identified model can be used to estimate the momentum. Hence, the update for the state in the physical set-up becomes:

$$x_{k+1} = \begin{bmatrix} q_{k+1} \\ \hat{p}_{k+1} \end{bmatrix} \tag{5-53}$$

where q_{k+1} denotes the measured new state after applying input u_k to state x_k and $\hat{p}_{k+1}$ denotes the simulated momentum obtained by simulating the model (5-45) with u_k and x_k such that the predicted $\hat{x}_{k+1}$ is obtained. Of course, there is a steady-state error introduced by the identified model, as can be seen in Figure 5-16b. To avoid this error, for small differences between state x_{k+1} and x_k the Euler-method is used to approximate the momentum. The difference in momentum estimation for a single trial for the method using the model and the Euler method is given in Figure 5-17. The trial is taken from the beginning of learning, where the algorithm has learned the swing-up but is rotating fast after that (Phase 2). In Figure 5-17a, the difference between the predicted momentum and the Euler momentum method is displayed. In this trial, it can be seen that the predicted model momentum (blue dashed) is more conservative (i.e. the momenta are generally lower) than when using the Euler method (black dashed-dotted). As can be seen in Figure 5-17b, the predicted next state $\hat{x}_{k+1}$ for the same trial is a very good estimate of the measured state. Therefore, this momentum estimate will be used for the remaining experiments. Of course, a well-tuned nonlinear observer will probably show even better results, which is a recommendation for future research, e.g. Venkatraman et al. [55] provide a mathematical way of building a nonlinear observer in PH form for systems satisfying (4-41).



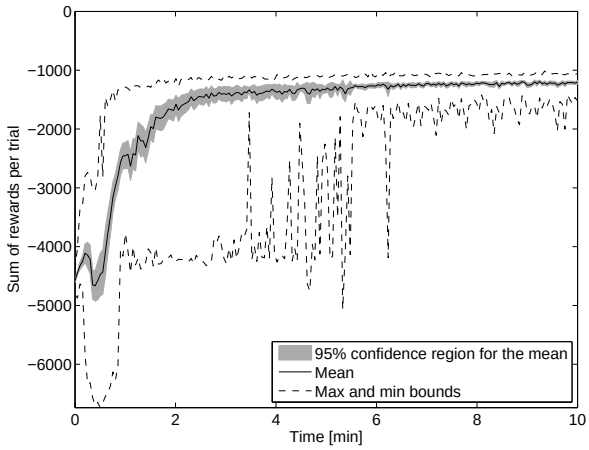
(a) Momentum p: Euler and predicted.

(b) Position q: measured and predicted.

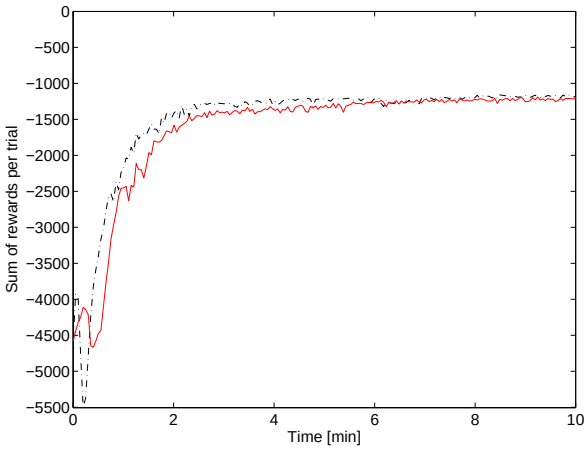
Figure 5-17: Comparison of Euler velocity estimation and model momentum estimation. In (a), the momentum calculated using the Euler method (black dashed-dotted) versus the model momentum estimation (blue dashed) is displayed. In (b), the predicted simulated output $\hat{x}_{k+1}$ (blue dashed) is compared to the measured output x_{k+1} (red solid) for the same trial, to illustrate accurateness of the identified model.

5-2-4 Simulation with Identified Model

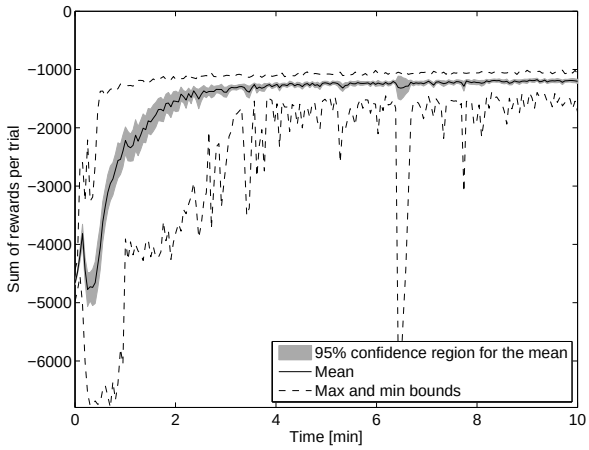
To study the effects of incorporating dry friction in the model, the identified parameters and changing the momentum estimation, two simulations (50 learning simulations) were done using the parameters from Table 4-2 with reward r_{D}, such that the simulations



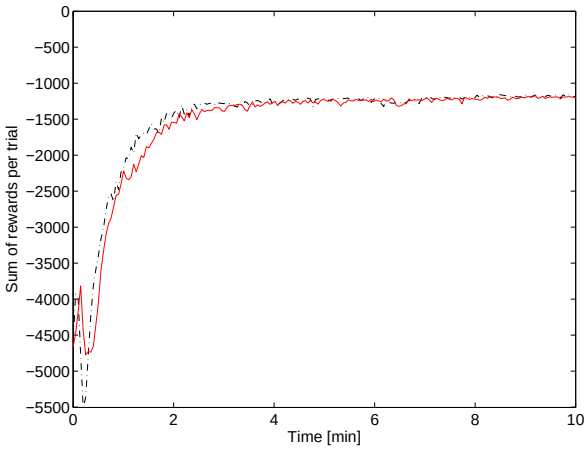
(a) Results for the EBAC method using model parameters for learning and identified parameters for simulation.



(b) Average of (a) compared to reference simulation average.



(c) Results for the EBAC method using identified parameters for learning and for simulation.



(d) Average of (c) compared to reference simulation average.

Figure 5-18: Results for the EBAC method for 50 learning simulations using identified parameters only for learning (a), using identified parameters both for learning and simulation (c) and the average of (a) and (c) compared in (b), (d) to the reference average of Figure 5-4d.

can be once again compared to the reference simulations and the effect of the identified model can be investigated.

- The first simulation uses the original model in the policy (4-56), while it uses the identified model for simulating the pendulum dynamics. The results are given in Figure 5-18a. As can be seen, the algorithm shows slower convergence when comparing the average cumulative reward (red solid) to the reference average (black dashed-dotted) which is displayed in Figure 5-18b. Also, it converges rapidly to a policy corresponding to a lower average cumulative reward, and needs more time to find the optimal policy. This can probably be attributed to the mismatch

between using the identified model for simulation and the original model in the policy (4-56).

- The second simulation[8] uses the identified model both in the policy (4-56) and for simulation. The results are given in Figure 5-18c. As can be seen in Figure 5-18d, the algorithm shows almost identical behaviour and convergence speed when compared to the reference average of Figure 5-4d, which can be seen in Figure 5-18d. Because the simulated pendulum dynamics now incorporate dry friction, learning is slightly slower in Phase 3 due to the decreased speed that enters (4-58) caused by dry friction. However, generally the performance is still quite good, with only 3 minutes of learning (60 trials) necessary to converge to an optimal policy. Also, this simulation shows slightly faster convergence than the simulation in Figure 5-18a. This can probably be attributed to the model mismatch that is present in the latter simulation, which causes some trials to converge very slowly, thus decreasing the overall average learning rate in comparison to the average learning rate of Figure 5-18c.

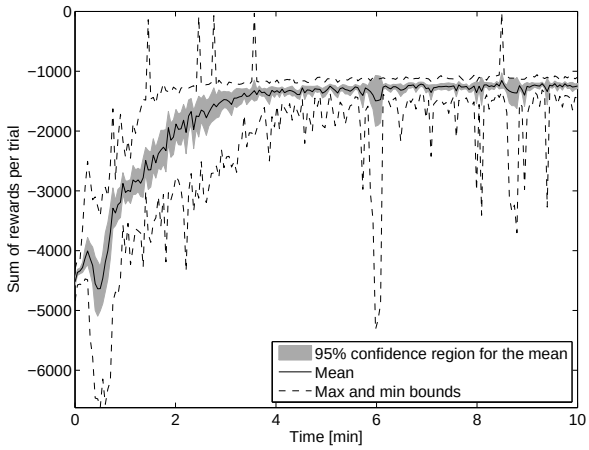
5-2-5 Experiments with Identified Model

Apart from simulations with the identified model, two experiments (20 learning experiments) were done using the parameters from Table 4-2 with reward r_{D}.

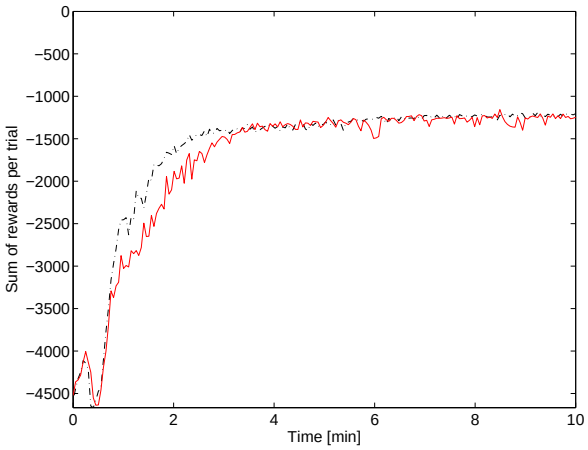
- The first experiment uses the original model in the policy (4-56), but instead of the Euler difference for the momentum estimation, the simulated model momentum from Section 5-2-3 is used on the physical set-up. Thus, the only difference between this experiment and the experiment of Figure 5-15a is the momentum estimation, which is done using the identified model. Hence, this experiment completely resembles the simulation of Figure 5-18a. The results are given in Figure 5-19a. As can be seen, the algorithm converges quickly and on average after approximately 60 trials a near-optimal policy is obtained. A few conclusions can be made. First, the momentum estimation using the Euler difference method results in poor performance when compared to using the model momentum estimation, given the difference between Figure 5-15a and Figure 5-19a. Clearly, learning speed has increased significantly as well as average performance. Also, the algorithm converges to a better policy, given the higher average cumulative reward. Second, comparing the average of Figure 5-19a with the average of the simulation of Figure 5-18a, it can be seen that the experimental results now more closely resemble the simulation results. Hence, it can be concluded that using the model for the velocity estimation delivers a better momentum estimator than the Euler difference method.

- The second experiment[9] uses the identified model in the policy (4-56) and the model momentum estimator on the physical set-up. Hence, this experiment com-

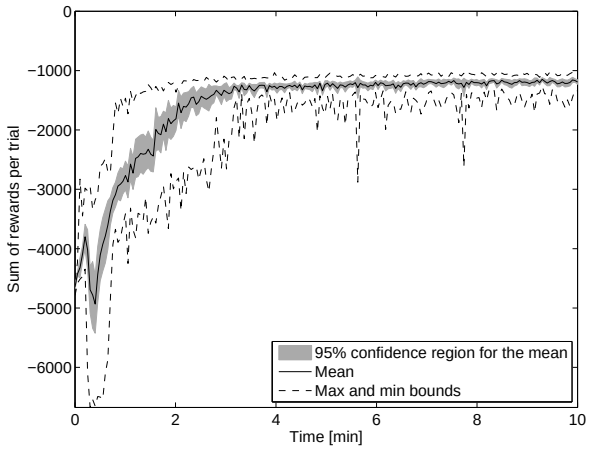
[8] Note that this is the simulation displayed in Chapter 4, hence Figure 5-18c is equivalent to Figure 4-2.
[9] Note that this is the experiment displayed in Chapter 4, hence Figure 5-19c is equivalent to Figure 4-6.



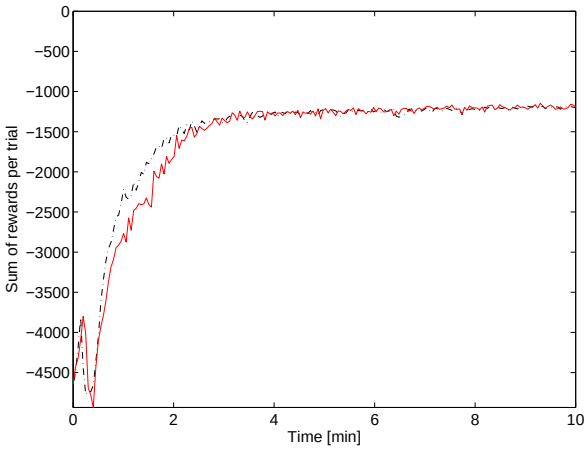
(a) Results for the EBAC method for 20 experiments using model parameters for learning.



(b) Average of (a) compared to reference simulation average.



(c) Results for the EBAC method for 20 experiments using identified parameters for learning.



(d) Average of (c) compared to reference simulation average.

Figure 5-19: Results for the EBAC method for 20 learning experiments using model parameters for learning (a) and using identified parameters for learning (c). The average of (a) and (c) is compared in (b) and (d) to their respective reference averages of Figure 5-18a and Figure 5-18c.

pletely resembles the simulation of Figure 5-18c. The results are given in Figure 5-19c. Learning speed is good, on average approximately 60 trials (3 minutes of learning) are necessary to reach a near-optimal policy. Comparing the simulations and experiments, Figure 5-19d, the experimental results indicate slower convergence (approximately 60 trials compared to 50 trials to reach near-optimal policy). The difference between experiment and simulation, also present in Figure 5-19b, can be attributed to errors in the model momentum estimator. Probably, a well-tuned observer would yield better results, which, as said before, is a recommendation for future research.

5-2-6 Initialization of Value Function with Identified Model

In the same way as for the simulation results, the initial drop in performance of the experiments can be eliminated by setting the value function V_0 to the minima of -1000 once again, i.e.

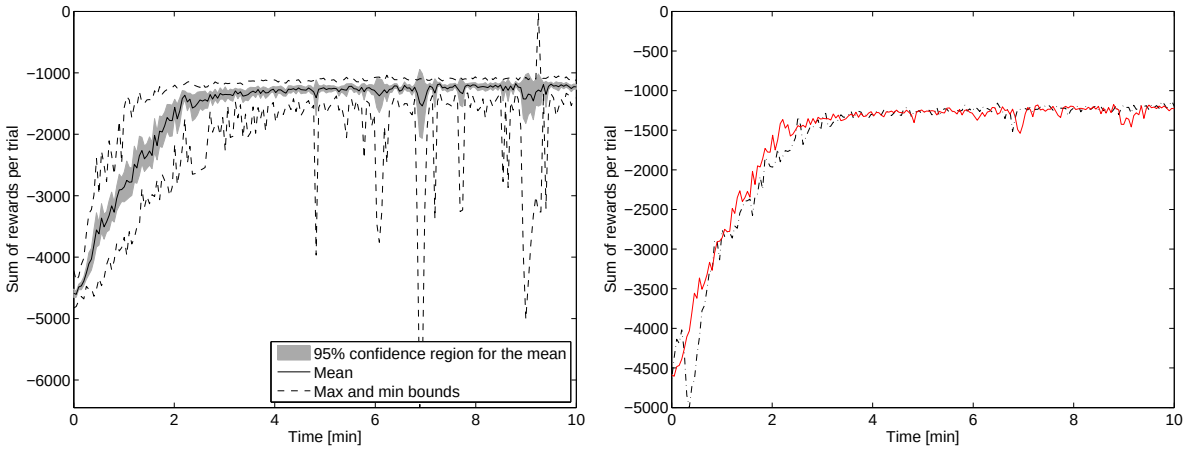
$$V_{0,\mathrm{r}}(x) = -1000 \qquad \forall x \tag{5-54}$$

Thus, 20 experiments were done on the physical set-up using both the model momentum estimation and the identified model for learning with the realistic value function initialization. The results are given in Figure 5-20a and a comparison with the average cumulative reward obtained without using value function initialization, Figure 5-19c, is given in Figure 5-20b. In simulations (see Section 5-1-6) the realistic initialization showed a significant performance increase in terms of speed of learning. However, this is not present in the experiments. Only the performance dip is now removed, as can be seen in Figure 5-20b. The following reasons can be given for the result. First, the performance dip was already very small, i.e. only one or two trials delivered the very negative reward, as can be seen from the black dashed-dotted line in Figure 5-20b. Second, as was seen in the previous simulations, the dominating factor for speed of learning is mainly the momentum with which the control policy is being calculated and the momentum that is obtained from the experiment/simulation. Since in this experiment, the momentum is left unchanged, there is no significant change in performance. Hence, these two reasons combined are the main cause that initializing the value function realistically when applying the EBAC method on the physical set-up does not yield much better performance than without this initialization. For future research, it is recommended to also try different initializations of the actors, because the initialization has effects on the cost function gradient estimate and thus on the speed of learning.

5-3 Conclusion

In this chapter, extra background and insights in the EBAC method have been given in order to further illustrate the method. First, the results from Chapter 4 were further discussed which leads to the following conclusions:

- A typical policy and value function have been presented for the simulation used in Chapter 4 which are in line with literature [20, 18].
- The control action during simulation with a learned policy was investigated, which showed the effects of the energy shaping and damping injection term, thus giving a better insight in the EBAC method.
- Passivity cannot be proven due to the nature of the damping term $\hat{K}_{\mathrm{d}}(x, \xi)$. For future research the energy change along trajectories can be investigated as well as the invariance of the sets of points where $\hat{K}_{\mathrm{d}}(x, \xi) \leq 0$. Also, it is interesting to study the passivity during learning, such that dissipation can be guaranteed during learning, thus increasing safety.



(a) Results for the EBAC method for 20 learning experiments with value function initialization (realistic).

(b) Average of (a) compared to reference simulation average.

Figure 5-20: Value function initialization realistically (red solid, $V_{0,\mathrm{r}} = -1000$) and reference of Figure 5-19c (black dashed-dotted, $V_0 = 0$). Realistic initialization eliminates the initial drop in performance.

Furthermore, numerous simulations were done to study the effects of changing important functions and parameters present in the EBAC method using the original model of the inverted pendulum. The following conclusions can be made:

- The cosine plus quadratic speed reward function (5-13) has been shown to be the best reward function of the five reward functions that were under consideration in the inverted pendulum simulation. This provides an interesting insight, since often [20, 18] reward type A (5-10) is used. Further study on the reward functions is recommended, because although these five reward functions are all closely related, it was shown for these five examples that learning performance is heavily affected when choosing for a particular type.

- The EBAC method can handle saturation functions that have a smooth and non-smooth derivative, where the standard saturation function with a non-smooth derivative (4-67) showed the best performance.

- The EBAC method is able to handle parameter uncertainty well in the sense that it is still possible to learn optimal policies in the presence of parameter uncertainty although the learning speed sometimes decreases, e.g. when the mass is increased by 20%. However, this can probably be overcome by choosing higher learning rates such that the damping term $\hat{K}(x, \psi)$ builds up quicker, such that more energy is pumped into the system at a faster rate.

- To speed up learning, it is convenient to initialize the value function realistically, which will prevent the performance dip commonly present in all the simulations and experiments. However, this has to be done on a trial-and-error basis which can

be difficult. The effect of better performance by value function initialization was not present in an experiment on the physical set-up, which can be attributed to the momentum estimation as the dominant factor for improving speed of learning. For future research, it might be interesting to see whether the actor can also be initialized based on EB-PBC perspectives, for example in the pendulum system, by initializing the desired potential energy with e.g. the negative of the system potential energy.

- The cost function gradient estimate used in the EBAC method outperforms a model-based method from [20], which is mainly caused by the difference in size of the gradient. However, this is only one type of cost function gradient and a lot of research has been done on these gradients [19] such that it is recommended to try to incorporate different gradient types in the EBAC method. Also, initialization of the actors can affect the result of this as well, which further emphasizes the importance of future study on initialization of the actors.

- The main cause of the increased learning speed in the EBAC method is the symmetrical basis function definition of Section 4-6-1. Therefore, in future research, effects of the function approximator should be further investigated, such as in [18], to get a better understanding of which type of function approximator is best suited for a specific control problem to which the EBAC method is applied and to what extent model knowledge can be incorporated in the function approximator.

- The Standard Actor-Critic (S-AC) method is outperformed by the EBAC method in terms of speed of learning, which is logical since the latter is a model-based method.

- Experiments on the real set-up confirm the results obtained from simulations, although the original model had to be adjusted to match the physical set-up and the momentum approximation had to be replaced. On average, a maximum of 60 trials, corresponding to 3 minutes of learning, is required in experiments to converge to a near-optimal policy. Simulations with similar settings show a better learning performance, on average a maximum of 50 trials is necessary to reach a near-optimal policy. The difference between simulations and experiments are caused by the presence of a speed estimator in the latter that introduces errors. This can be overcome by using a nonlinear observer, which is recommended for future work.

Chapter 6

Conclusions and Recommendations

In this thesis, Energy-Balancing Passivity-Based Control (EB-PBC) for port-Hamiltonian (PH) systems was combined with Actor-Critic (AC) Reinforcement Learning (RL). This chapter summarizes the findings of this thesis and lists the most important recommendations for future research.

6-1 Summary and Conclusions

The general goal of this thesis was to design a methodology that combines the major advantages of RL and PBC for PH systems. Therefore, Energy-Balancing Passivity-Based Control (EB-PBC) (Chapter 2) was combined with the Standard Actor-Critic (S-AC) algorithm (Chapter 3) to yield the Energy-Balancing Actor-Critic (EBAC) (Chapter 4). In the EBAC method, the Partial Differential Equation (PDE) arising from EB-PBC is parameterized such that it is split in two parts: a part satisfying a matching condition and a part that can be arbitrarily assigned. The assignable part is then parameterized such that it is suitable for AC RL. Finally, damping is injected by learning parameters of the desired damping matrix. The results for the EBAC method were verified both in simulations and experiments on the inverted pendulum set-up, which showed its effectiveness. The most important conclusions from simulations and experiments on the inverted-pendulum set-up with the EBAC method are given below.

- The EBAC method can learn near-optimal controllers that can be interpreted in terms of energy shaping and damping injection techniques by endowing the closed-loop with a prescribed PH structure. The advantage is that the learned controller has a physical interpretation in terms of energy exchange. Local stability can be numerically demonstrated by using passivity theory. Unfortunately, the author was not able to provide a conclusive proof of passivity for the closed-loop system, only local stability could be numerically assessed by studying the simulation and experimental results.

- There is no need in the EBAC method to explicitly solve a generally complex Partial Differential Equation (PDE). This last issue is commonly encountered in PBC for PH systems, which was demonstrated in Chapter 2.
- Control saturation can be incorporated in the EBAC method, however in the inverted pendulum example, control saturation requires a pumping-damping regime for swinging up the pendulum, which means that passivity for the entire state space of the system using the learned desired Hamiltonian cannot be proven in the classical way of Definition 2-1.1.
- The most important contribution to the enhanced speed of learning of EBAC over the model-free S-AC method is incorporating model knowledge by symmetrically defining the basis functions (Section 5-1-8). This is an important aspect however, since this definition ensures that the policy is well-behaved from a PBC perspective. This means that by parameterizing in a certain manner, properties of functions can be guaranteed, at least locally around an equilibrium point. For the inverted pendulum, it was shown that the desired potential energy naturally locally satisfies certain Lyapunov-stability properties by using this parameterization. This is beneficial in (numerical) stability analysis.
- The EBAC method is robust in the sense that it can handle model uncertainty well, as extensive simulations with varying parameters have shown. Also, performance is barely affected by changing the type of saturation.

6-2 Open Issues and Recommendations

- The Fourier Basis is naturally a suitable basis for the inverted pendulum problem, such that few parameters can be used to learn good policies. It is recommended to study the effect of the approximator on the performance of the EBAC method, since it is required that the basis functions satisfy a number of properties. First, it has to have a continuous, preferably smooth derivative, because this derivative will be used to approximate the gradient of the desired Hamiltonian. Second, the gradient of the basis should be able to represent the gradient of an energy function, hence e.g. a Polynomial Basis would be a good choice.
- The effect of initialization should be further studied, as simulation showed that the learning speed in the EBAC method could be further increased by initializing the value function with a realistic value. Experiments however did not confirm this behaviour, although the performance drop in the beginning of learning was removed. For future research, it is interesting to study the effect of incorporating model knowledge in the initialization of the actors and critic, so as to speed up learning even more. This can be done by initializing the actors based on PBC perspectives.
- EB-PBC is possibly too limited, for example, it is less applicable in underactuated systems. Hence, in future research a more general PBC method has to be

chosen in order to be able to deal with e.g. systems with underactuation. The IDA-PBC method is a well-studied control type that would be a good choice for this. For example, algebraic IDA-PBC could be exploited, in which the desired Hamiltonian is fixed and $R_{\mathrm{d}}(x), J_{\mathrm{d}}(x)$ and $g^{\perp}(x)$ can be computed. A proposal for parameterization can be given as:

$$\hat{H}_{\mathrm{d}}(x, \vartheta) = \nu^T \phi(x) \tag{6-1}$$

with $\phi(x)$ basis functions and ν a parameter vector such that the desired Hamiltonian is completely parameterized. Then, to satisfy (2-68), the interconnection matrix $F_{\mathrm{d}}(x)$ can be computed using the approximated $\hat{H}_{\mathrm{d}}(x, \nu)$. When computing $F_{\mathrm{d}}(x)$ the only property that has to be satisfied, apart from the PDE (2-68), is (2-53). There are some key advantages and disadvantages of this approach:

- As mentioned in Section 2-3-4, IDA-PBC is applicable to a larger set of PH plants than EB-PBC, because the interconnection and damping matrix is now also free for assignment. Hence, the freedom of choosing $F_{\mathrm{d}}(x)$ based on the parameterized desired Hamiltonian enlarges the set of stabilizable plants to which the method can be applied.
- The desired Hamiltonian can be completely parameterized and if it is parameterized as in (6-1), only one actor is necessary to update the gradient, which means that the desired damping is also included in this actor update.
- The energy-balancing property will be generally lost, which means that the solutions found can no longer be intuitively interpreted as in the EBAC method, which is based on EB-PBC. This includes the loss of the intuitive CbI methods, because as Ortega et al. [41] point out, there is no CbI version of IDA-PBC yet.

Also, in [28] a proposal for parameterization of IDA-PBC is done such that the parameterized IDA-PBC control law renders the closed-loop system (asymptotically) stable and that it behaves like a linear sample system. For future research, a similar parameterization as in [28] could be investigated in which the parameters are learned via RL. Finally, a lot of research has been done on IDA-PBC for mechanical systems [40, 17, 1]. In this special case, $F_{\mathrm{d}}(x)$ can be suitably parameterized such that the structure of the Hamiltonian of the mechanical system is preserved and solutions can be interpreted in terms of energy-balancing strategies. This is also left as a recommendation for further study, to combine these methods with a form of RL.

- In this thesis, the EBAC method was only applied to the inverted pendulum set-up. It is recommended for future research to apply the method to a more complex real-life system such as, for example, a robotic manipulator, in which the PH framework can be beneficial when modeling the interaction between various dynamical (sub)systems. In [12, 49] various methods using a port-Hamiltonian approach to the control of interaction are proposed. Using these methods in combination with a complex real-life set-up would be an interesting application of the EBAC method.

6-3 Final Words

As the title suggests, in this thesis machine learning has been embedded into passivity theory using the port-Hamiltonian framework. Specifically, an Actor-Critic Reinforcement Learning method has been combined with Energy-Balancing Passivity-Based Control to yield the Energy-Balancing Actor-Critic method. The EBAC method indeed incorporates advantages of the two methods, while relaxing their disadvantages, which was the general goal of this thesis. Simulations and experiments showed the effectiveness of the proposed method. However, no conclusive stability theorem could be derived based on passivity, which is left open for future research. Also, the application of the EBAC method to more complex systems, such as robotic manipulators, will further provide insights in the combination of energy-based control and RL. To conclude, in this thesis another step has been taken to bring two fields in control - PBC and RL - closer together, with the hope of someday being able to safely implement learning controllers in interactive environments.

Appendix A

List of Simulations and Experiments

In this appendix an overview is given of which MATLAB[1] files have been used to generate each figure in this thesis, along with the path on the supplied DVD of the relevant m-file, such that each figure in this thesis can be reproduced if necessary. For all the simulations and experiments, 2 types of filenames were used:

- eval_{}: File used to evaluate the simulation or experiment.
- compare_{}: File used to compare the simulation or experiment with a relevant reference simulation or experiment.

Each path in Tables A-1–A-2 is relative to the home directory

Matlab/

of the DVD supplied with this thesis. For more information, please look in the directory to see which data files were used by each m-file.

A-1 Simulations

(see next page)

[1] MATLAB is a registered trademark of The Mathworks Inc., Natick, Massachusetts. In this thesis, version 7.13.0.564 (R2011b), 32bit, was used.

Table A-1: Simulation figure data and associated Matlab files.

Figure no.	**Directory**	m**-file**
4-2, 4-3a, 4-3b, 4-4a, 4-4b, 4-5a, 4-5b, 4-6	paper/	eval_paper
5-1, 5-2, 5-3	simulations/ebac/invpend/identification/identifiedboth	eval_identifiedboth
5-4a	simulations/ebac/invpend/reward/R_A/	eval_R_A
5-4b	simulations/ebac/invpend/reward/R_B/	eval_R_B
5-4c	simulations/ebac/invpend/reward/R_C/	eval_R_C
5-4d	simulations/ebac/invpend/reward/R_D/	eval_R_D
5-4e	simulations/ebac/invpend/reward/R_E/	eval_R_E
5-5	simulations/ebac/invpend/reward/	compare_reward
5-6	simulations/ebac/invpend/saturation/	compare_saturation
5-7a	simulations/ebac/invpend/saturation/atan/	eval_atan
5-7b	simulations/ebac/invpend/saturation/tanh/	eval_tanh
5-8a	simulations/ebac/invpend/sensitivity/lessvoltage/	eval_lessvoltage
5-8b	simulations/ebac/invpend/sensitivity/lessvoltage/lessvoltage_1/	eval_lessvoltage
5-9a	simulations/ebac/invpend/sensitivity/	eval_mass
5-9b	simulations/ebac/invpend/sensitivity/	eval_inertia
5-9c	simulations/ebac/invpend/sensitivity/	eval_damping
5-10	simulations/ebac/invpend/initialization/	compare_initialization
5-11a	simulations/ebac/invpend/polgrad/modelpolgrad/	eval_modelpolgrad
5-11b, 5-12a, 5-12b	simulations/ebac/invpend/polgrad/	compare_polgrad
5-13a	simulations/ebac/invpend/basis/four_normal/	eval_ebac_four_normal
5-13b	simulations/ebac/invpend/basis/	compare_basis
5-14a	simulations/sac/invpend/basis/four_normal/	eval_sac_four_normal
5-14b	simulations/sac/invpend/basis/	compare_sac_ebac
5-18a	simulations/ebac/invpend/identification/identified/	eval_identified
5-18b, 5-18d	simulations/ebac/invpend/identification/identified/	compare_identified
5-18c	simulations/ebac/invpend/identification/identifiedboth/	eval_identifiedboth

A-2 Experiments

Table A-2: Experiment figure data and associated Matlab files.

Figure no.	**Directory**	**m-file**
5-15a	experiments/ebac/exp1_normal/	eval_exp1
5-15b	experiments/ebac/exp1_normal/	compare_sim_exp
5-16a, 5-16b, B-1	experiments/identification/	eval_identification
5-17a, 5-17b	experiments/ebac/exp2_ModelMomentum	compare_momentum
5-19a	experiments/ebac/exp2_ModelMomentum	eval_exp2
5-19b	experiments/ebac/exp2_ModelMomentum	compare_identified
5-19c	experiments/ebac/exp3_MMandIdentified	eval_exp3
5-19d	experiments/ebac/exp3_MMandIdentified	compare_identifiedboth
5-20a	experiments/ebac/exp4_MMIandV0	eval_exp4
5-20b	experiments/ebac/exp4_MMIandV0	compare_V0andMMI

Appendix B

Identification Input

The multisinusoidal input used for identification of the inverted pendulum set-up is given in Figure B-1.

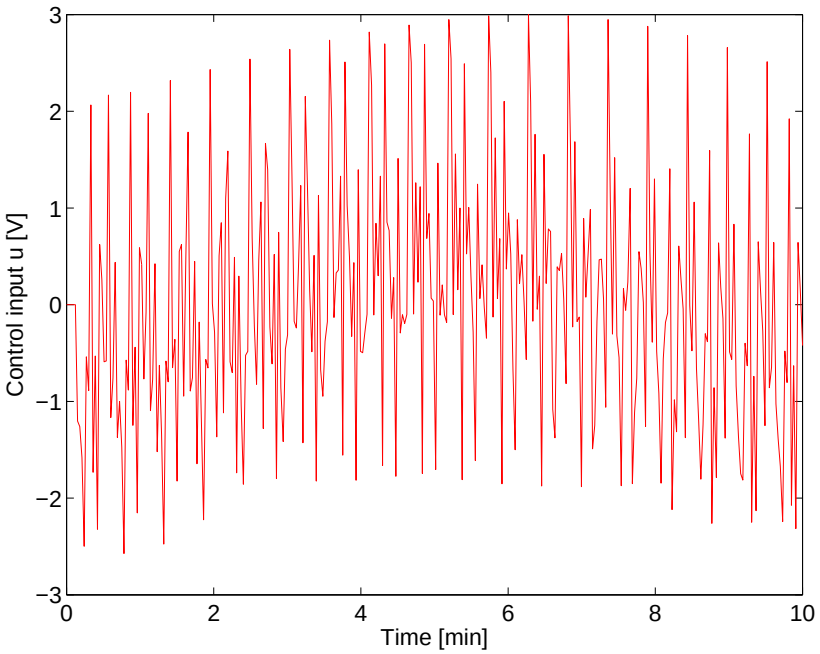


Figure B-1: Multisine input used for identification.

Bibliography

[1] J.A. Acosta, R. Ortega, A. Astolfi, and A.D. Mahindrakar. Interconnection and damping assignment passivity-based control of mechanical systems with underactuation degree one. *IEEE Transactions on Automatic Control*, 50(12):1936–1955, 2005.

[2] K.J. Åstrom, J. Aracil, and F. Gordillo. A family of smooth controllers for swinging up a pendulum. *Automatica*, 44:1841–1848, 2008.

[3] R. Babuška. *Fuzzy Modeling for Control*. Kluwer Academic Publishers, 1998.

[4] R. Babuška. *Knowledge-Based Control Systems*, chapter 9, pages 149–178. Delft University of Technology, 2010.

[5] A. Barto, R. Sutton, and C. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846, 1983.

[6] S. Bhasin, M. Johnson, and W. E. Dixon. A model-free robust policy iteration algorithm for optimal control of nonlinear systems. In *Proc. 49th IEEE Conf. Decision and Control (CDC)*, pages 3060–3065, 2010.

[7] B. Chu, D. Hong, J. Park, and J. Chung. Passive dynamic walker controller design employing an RLS-based natural actor-critic learning algorithm. *Engineering Applications of Artificial Intelligence*, 21:1027–1034, 2008.

[8] C. Cornejo and L. Alvarez-Icaza. A nonlinear friction model for the passivity-based control of underactuated mechanical systems. In *Proc. 46th IEEE Conf. Decision and Control*, pages 3859–3864, 2007.

[9] D.A. Dirksz, J.M.A. Scherpen, and R. Ortega. Interconnection and damping assignment passivity-based control for port-Hamiltonian mechanical systems with only position measurements. In *Proc. 47th IEEE Conf. Decision and Control CDC 2008*, pages 4957–4962, 2008.

[10] K. Doya. Reinforcement learning in continuous time and space. *Neural Computation*, 12:219–245, 2000.

[11] V. Duindam, A. Macchelli, S. Stramigioli, and H. Bruyninckx. *Modeling and Control of Complex Physical Systems*. Springer-Verlag, 2009.

[12] V. Duindam and S. Stramigioli. *Modeling and Control for Efficient Bipedal Walking Robots: A Port-Based Approach*. Springer, 2009.

[13] G. Escobar, R. Ortega, and H. Sira-Ramirez. Output-feedback global stabilization of a nonlinear benchmark system using a saturated passivity-based controller. *IEEE Transactions on Control Systems Technology*, 7(2):289–293, 1999.

[14] K. Fujimoto and T. Sugie. Iterative learning control of Hamiltonian systems: I/O based optimal control approach. *IEEE Transactions on Automatic Control*, 48(10):1756–1761, 2003.

[15] E. García-Canseco, D. Jeltsema, R. Ortega, and J. Scherpen. Power-based control of physical systems. *Automatica*, 46:127–132, 2010.

[16] E. García-Canseco, R. Ortega, J.M.A. Scherpen, and D. Jeltsema. Power shaping control of nonlinear systems: A benchmark example. In *Lagrangian and Hamiltonian Methods for Nonlinear Control*, volume 366 of *Lecture Notes in Control and Information Sciences*, pages 135–146. Springer Berlin / Heidelberg, 2007.

[17] F. Gomez-Estern and A.J. van der Schaft. Physical damping in IDA-PBC controlled underactuated mechanical systems. *European Journal of Control*, 10:451–468, 2004.

[18] I. Grondman, L. Buşoniu, and R. Babuška. Model learning actor-critic algorithms: Performance evaluation in a motion control task. In *2012 IEEE 51st Annual Conference on Decision and Control*, Under review.

[19] I. Grondman, L. Buşoniu, G.A.D Lopes, and R. Babuška. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications & Reviews*, Under review.

[20] I. Grondman, M. Vaandrager, L. Buşoniu, R. Babuška, and E. Schuitema. Efficient model learning methods for actor-critic control. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, In press.

[21] V. Heidrich-Meisner, M. Lauer, C. Igel, and M. Riedmiller. Reinforcement learning in a nutshell. In *15th European Symposium on Artificial Neural Networks (ESANN 2007)*, pages 277–288, 2007.

[22] S. Kakade. A natural policy gradient. *Advances in Neural Information Processing Systems*, 14:1531–1538, 2002.

[23] H.K. Khalil. *Nonlinear Systems Third Edition*. Prentice Hall, 2002.

[24] B. Kim, J. Park, S. Park, and S. Kang. Impedance learning for robotic contact tasks using natural actor-critic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 40(2):433–443, 2010.

[25] V.R. Konda and J.N. Tsitsiklis. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

[26] G. Konidaris and S. Osentoski. Value function approximation in reinforcement learning using the Fourier basis. Technical Report 101, Autonomous Learning Laboratory, Computer Science Department, University of Massachusetts Amherst, 2008.

[27] J. Koopman, D. Jeltsema, and M. Verhaegen. Port-hamiltonian formulation and analysis of the lugre friction model. In *Proc. 47th IEEE Conf. Decision and Control CDC 2008*, pages 3181–3186, 2008.

[28] P. Kotyczka and B. Lohmann. A constructive approach for the parametrization of interconnection and damping assignment passivity based control. In *Proc. 16th Mediterranean Conf. Control and Automation*, pages 964–969, 2008.

[29] J.C. Lagarias, J.A. Reeds, M.H. Wright, and P.E. Wright. Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9:112–147, 1998.

[30] A. Loria, R. Kelly, R. Ortega, and V. Santibanez. On global output feedback regulation of euler-lagrange systems with bounded inputs. *IEEE Transactions on Automatic Control*, 42(8):1138–1143, 1997.

[31] A.D. Mahindrakar, A. Astolfi, R. Ortega, and G. Viola. Further constructive results on interconnection and damping assignment control of mechanical systems: the acrobot example. In *Proc. American Control Conf*, pages 5584–5589, 2006.

[32] B. Maschke, R. Ortega, and A. van der Schaft. Energy-based Lyapunov functions for forced Hamiltonian systems with dissipation. *IEEE Transactions on Automatic Control*, 45(8):1498–1502, 2000.

[33] B. Maschke and A. van der Schaft. Port-controlled Hamiltonian systems: modelling origins and system theoretic properties. In *Proceedings of the third Conference on nonlinear control systems (NOLCOS)*, pages 282–288, 1992.

[34] V. Muralidharan, M.T. Ravichandran, and A.D. Mahindrakar. Extending interconnection and damping assignment passivity-based control (IDA-PBC) to underactuated mechanical systems with nonholonomic Pfaffian constraints: The mobile inverted pendulum robot. In *Proc. 48th IEEE Conf. held jointly with the 2009 28th Chinese Control Conf Decision and Control CDC/CCC 2009*, pages 6305–6310, 2009.

[35] Y. Nakamura, T. Mori, and S. Ishii. Natural policy gradient reinforcement learning for a CPG control of a biped robot. In *Parallel Problem Solving from Nature - PPSN VIII*. Springer Berlin / Heidelberg, 2004.

[36] R. Ortega and E. García-Canseco. Interconnection and damping assignment passivity-based control: A survey. *European Journal of Control*, 10:432–450, 2004.

[37] R. Ortega, D. Jeltsema, and J. M. A. Scherpen. Power shaping: a new paradigm for stabilization of nonlinear RLC circuits. *IEEE Transactions on Automatic Control*, 48(10):1762–1767, 2003.

[38] R. Ortega, A. Loria, and P.J. Nicklasson. *Passivity Based Control of Euler-Lagrange Systems: Mechanical, Electrical and Electromechanical Applications*. Springer-Verlag, 1998.

[39] R. Ortega and M. W. Spong. Adaptive motion control of rigid robots: a tutorial. *Automatica*, 25:877–888, 1989.

[40] R. Ortega, M. W. Spong, F. Gomez-Estern, and G. Blankenstein. Stabilization of a class of underactuated mechanical systems via interconnection and damping assignment. *IEEE Transactions on Automatic Control*, 47(8):1218–1233, 2002.

[41] R. Ortega, A. van der Schaft, F. Castanos, and A. Astolfi. Control by interconnection and standard passivity-based control of port-Hamiltonian systems. *IEEE Transactions on Automatic Control*, 53(11):2527–2542, 2008.

[42] R. Ortega, A. van der Schaft, I. Mareels, and B. Maschke. Putting energy back in control. *IEEE Control Systems Magazine*, 21(2):18–33, 2001.

[43] R. Ortega, A. van der Schaft, and B. Maschke. Stabilization of port-controlled Hamiltonian systems via energy balancing. In *Lecture Notes in Control and Information Sciences*, volume 246 of *Lecture Notes in Control and Information Sciences*, pages 239–260. Springer Berlin / Heidelberg, 1999.

[44] R. Ortega, A. van der Schaft, B. Maschke, and G. Escobar. Energy-shaping of port-controlled Hamiltonian systems by interconnection. In *Proc. 38th IEEE Conf. Decision and Control*, volume 2, pages 1646–1651, 1999.

[45] R. Ortega, A. van der Schaft, B. Maschke, and G. Escobar. Interconnection and damping assignment passivity-based control of port-controlled Hamiltonian systems. *Automatica*, 38:585–596, 2002.

[46] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71:1180–1190, 2008.

[47] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21:682–697, 2008.

[48] J. Sandoval, R. Kelly, and V. Santibánez. Interconnection and damping assignment passivity-based control of a class of underactuated mechanical systems with dynamic friction. *International Journal of Robust and Nonlinear Control*, 21:738–751, 2011.

[49] C. Secchi and S. Stramigioli. *Control of Interactive Robotic Interfaces*. Springer, 2007.

[50] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[51] R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12:1057–1063, 2000.

[52] K. Vamvoudakis and F. Lewis. Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, 46:878–888, 2010.

[53] A. van der Schaft. *L_2-Gain and Passivity Techniques in Nonlinear Control*. Springer, 2000.

[54] A. van der Schaft and B. Maschke. On the Hamiltonian formulation of nonholonomic mechanical systems. *Reports on Mathematical Physics*, 34:225–233, 1994.

[55] A. Venkatraman. *Control of Port-Hamiltonian systems: Observer Design and Alternate Passive Input-Output Pairs*. PhD thesis, University of Groningen, 2010.

[56] D. Vrabie and F. Lewis. Online adaptive optimal control based on reinforcement learning. *Optimization and Optimal Control*, 39:309–323, 2010.

[57] M. A. Wiering and H. van Hasselt. Two novel on-policy reinforcement learning algorithms based on TD(λ)-methods. In *Proc. IEEE Int. Symp. Approximate Dynamic Programming and Reinforcement Learning ADPRL 2007*, pages 280–287, 2007.

[58] M. A. Wiering and H. van Hasselt. The QV-family compared to other reinforcement learning algorithms. In *Proc. IEEE Symp. Adaptive Dynamic Programming and Reinforcement Learning ADPRL '09*, pages 101–108, 2009.

[59] J.C. Willems. Dissipative dynamical systems - part 1: General theory. *Archive for Rational Mechanics and Analysis*, 45:321–351, 1972.

[60] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 18:229–256, 1992.

Glossary

List of Acronyms

AC	Actor-Critic
B-IDA	Basic Interconnection and Damping Assignment Passivity-Based Control
CbDI	Control by Damping Injection
CbI	Control by Interconnection
DCSC	Delft Center for Systems and Control
DOF	Degree Of Freedom
DP	Dynamic Programming
EB-PBC	Energy-Balancing Passivity-Based Control
EBAC	Energy-Balancing Actor-Critic
IDA-PBC	Interconnection and Damping Assignment Passivity-Based Control
IFAC	International Federation of Automatic Control
LLR	Local Linear Regression
MDP	Markov Decision Process
MLAC	Model Learning Actor-Critic
MSc	Master of Science
NAC	Natural Actor-Critic
POMDP	Partially Observable Markov Decision Process
PBC	Passivity-Based Control

PDE	Partial Differential Equation
PH	port-Hamiltonian
PS-PBC	Power-Shaping Passivity-Based Control
RBF	Radial Basis Functions
RL	Reinforcement Learning
S-AC	Standard Actor-Critic
SPBC	Standard Passivity-Based Control
TD	Temporal Difference

List of Symbols

General

t	Continuous time instant
x	Continuous state of system Σ
y	Continuous output of system Σ
Σ	State space system
$\mathbb{R}$	Set of real numbers
$\mathbb{Z}$	Set of integer numbers
$\hat{a}$	Approximation or prediction of a variable or function a

Passivity-Based Control

d_{d}	Desired closed-loop damping
f	Continuous function describing system dynamics of Σ
g	Input matrix
h	Continuous function describing output of system Σ
s	Supply rate function
u	Continuous control input vector
v	Closed-loop input vector
z	Closed-loop output vector
C	Casimir function
D	Vector function
F	Interconnection and damping matrix
H	Hamiltonian
J	Interconnection matrix
K	Passive output damping matrix

L	Lyapunov function
R	Damping matrix
S	Storage function
U	Continuous set of control inputs for system Σ
Y	Continuous set of outputs for system Σ
$\mathcal{X}$	State space manifold
α	Static state dependent feedback
β	Continuous state feedback
δ	Arbitrary positive number
ϵ	Arbitrary positive number
κ	Level constant
ς	Control saturation function
ζ	Controller PH system continuous state variables
Ω	Set

Reinforcement Learning

e	Eligibility trace
k	Discrete time instant
r	Reward function
u_k	Discrete time control input
x_k	Discrete time state
E	Expectation
J	Cost function that has to be maximized in RL
V	State value function
P	State-action probability distribution
Q	State-action value function
$\mathcal{P}$	State-action transition probability distribution in an MDP
$\mathcal{R}$	Expected value of next reward
$\mathcal{U}$	Discrete time set of control actions
$\mathcal{X}$	Discrete time set of states
α	Learning rate
δ	Temporal difference
γ	Discount factor
ε	Greedy action selection probability
θ	Critic parameter vector
ϑ	Actor parameter vector
λ	Eligibility trace decay parameter
π	Deterministic policy
ϕ	Basis function

Δu	Exploratory action
Π	Probabilistic policy

Energy-Balancing Actor-Critic

a	Arbitrary vector
$b_{\{\text{es,di}\}}$	2-norm of cost function gradient
e_μ	Scalar quadratic error in the prediction of pendulum model parameters μ
i, j, l	Indices used in (multiple) element operations
q	Generalized position vector
$\boldsymbol{q}$	Position trajectory data
p	Generalized momentum vector
w	Vector containing states that correspond to non-zero elements of N
z	Vector containing states that correspond to zero elements of N
A	Matrix containing matching conditions
$B_{\{\text{es,di}\}}$	Sum of 2-norm of cost function gradient
F_f	Coulomb friction force
G	Input matrix in mechanical systems
I	Identity matrix
M	Inertia matrix
N	Null space of matrix A
P	Potential energy
Q_r	Reward state penalty matrix
R_r	Reward control action penalty
S	Saturation function subset of $\mathbb{R}^m$
$s_{\{\text{t,a}\}}$	Saturation function skewness
T_t	Time per trial
T_s	Sample time
μ	Pendulum model parameter vector
ν	IDA-PBC desired Hamiltonian parameter vector
ξ	Parameter vector for energy shaping
σ^2	Exploration variance
ψ	Parameter vector for damping injection
Ψ	Parameter matrix for damping injection
$\{Par\}_\text{p}$	Pendulum model parameter {Par}
$\mathcal{N}$	Normal distribution

Important Mathematical Operators

$\frac{\partial f}{\partial x}$ or $\nabla_x f$	Partial derivative of a function f to x
$\frac{df}{dt}$ or $\dot{f}$	Time derivative of a function f

D^T	Transposed of a matrix D
$D^\perp$	Left annihilator of a matrix D, i.e. $D^\perp D = 0$
$D^\dagger$	Pseudo-inverse of a matrix D, i.e. $D^\dagger D = I$
$\|\cdot\|_2$	Two-norm of an input $(\cdot)$