Machine Learning of Motor Skills for Robotics

Ph.D. Dissertation Proposal
submitted by

Jan Peters

May 2, 2005

Guidance Committee

Dr.	Stefan Schaal	(chair)
Dr.	Gaurav Sukhatme	
Dr.	Sven Koenig	
Dr.	Laurent Itti	
Dr.	Firdaus Udwadia	(external)

Acknowledgments

First of all, I have to thank the wonderful people at the University of Southern California. This thesis proposal would have never been possible without initiation, continuing encouragement, coordination, supervision and understanding help of *Stefan Schaal*. He is a great ‘sensei’ and has endured my emotional roller-coaster ride from my masters thesis to today — over the last five years. Another thanks goes to *Firdaus Udwadia* and *Sethu Vijayakumar* from whom both I learned a lot on analytical dynamics and machine learning, respectively. I am also very grateful to the committee members, i.e., *Stefan Schaal, Sven Koenig, Gaurav Sukhatme, Laurent Itti* and *Firdaus Udwadia* for reading my thesis proposal and participating in the exam. I am indebted to *Mitsuo Kawato* and *Gordon Cheng* who made two stays in 2000 and 2003 at the Advanced Telecommunication Research Center ATR in Kansai Science City, Kyoto, Japan, possible. Without the first visit in 2000, I would have never met Stefan Schaal and my life would have taken a very different turn. Finally, I have to thank all my friends and family for leading me to a career in science and continuing support. My girl-friend *Aysenur Altinguel* for her understanding and love. Another thanks goes to my fellow trolls in the HNB dungeon, i.e., *Aaron d'Souza, Aude Billard, Auke Ijspeert, Dimitris Pongas, Jun Nakanishi, Michael Mistry, Nerses Ohanyan, Peyman Mohajerian, Rick Cory, Sethu Viyahakumar, Srideep Musuvathy* and *Vidhya Navalpakkam* for all the advice, support, help, and . . . the good times and great parties!

Für meinen Opa, der mich durch sein Vorbild Disziplin und Gradlienigkeit lehrte,
für meinen Vater, der mich zum Denken und als Wissenschaftler aufzog,
für meine Mutter, die mir Kraft, Liebe und Kreativität auf den Weg gab,
für meine Freundin, die mich jeden Tag wieder glücklich macht,
für meine Schwester, die oft an mich denkt,
und für meinen viel zu netten wissenschaftlichen “grossen Bruder”
zum Dank für die letzten fünf Jahre.

Contents

List Of Tables

List Of Figures

Abstract

Autonomous robots that can assist humans in situations of daily life have been a long standing vision of robotics, artificial intelligence, and cognitive sciences. A first step towards this goal is to create robots that can accomplish a multitude of different tasks, triggered by environmental context or higher level instruction. Early approaches to this goal during the heydays of artificial intelligence research in the late 1980ies, however, made it clear that an approach purely based on reasoning and human insights would not be able to model all the perceptuomotor tasks that a robot should fulfill. Instead, new hope was put in the growing wake of machine learning that promised fully adaptive control algorithms which learn both by observation and trial-and-error. However, to date, learning techniques have yet to fulfill this promise as only few methods manage to scale into the high-dimensional domains of manipulator robotics, or even the new upcoming trend of humanoid robotics, and usually scaling was only achieved in well pre-structured domains. In this thesis proposal we investigate how a more general representation for motor skills, i.e., parameterized policies or motor primitives, can be used in combination with novel, modern machine learning algorithms in order to make one step closer towards human-like performance of learning control and motor skills. For doing so, we look at the machine learning of motor skills at three different levels, i.e., (i) the execution of motion, (ii) the generation of building blocks of motion and (iii) their sequencing and parallelization. At the *execution level*, i.e., lowest level, we study how we can control skills on a real robot with a particular focus on skills represented as dynamical systems $A_i(\dot{x}, x, t)\ddot{x} = b_i(\dot{x}, x, t)$. We present an optimal control framework based upon a generalization of Gauss' principle and show how various well-known robot control laws can be derived using this framework by just changing the metric of the cost function. We can show successful applications to inverse kinematics control for holomonic systems for several different metrics. This framework comes with a drawback as it requires an accurate model of the robot system which is rarely given in practice; we therefore discuss the perspective of integrating learning into this framework using techniques from internal model learning. The intermediary level is the one of the building blocks of motions, i.e., the *motor primitive level*. Following Ijspeert et al. (Ijspeert, Nakanishi, & Schaal, 2001, 2003), we focus on motor primitives represented by a special kind of parameterized nonlinear dynamical systems of the type $A_i\ddot{x} = b_i$. These motor primitives can be learned using a combination of supervised learning with trial-and-error, where the supervised learning is used for the initialization of the policy. We discuss different reinforcement learning techniques with respect to their applicability to

this problem and focus on parameterized policy search methods. It can be shown that a special kind of technique, i.e., the episodic natural actor-critic, is particularly well-suited for learning motor primitives – especially when compared to other gradient-based policy search techniques. We show first steps of how this technique can be extended towards probabilistic policy search. When such motor primitives are combined together, they can form very complex movements. Such complex movements require a representation which allows both the parallelization and sequencing of previously learned building blocks. We refer to this level as the *motor task level*. We review how such hierarchical learning problems have traditionally been tackled by the learning control community and show some preliminary experiments how this can be achieved in the context of the motor primitives above. In the light of the previously discussed levels of motor skills, we will discuss several important motor skills as applications of the techniques discusssed. These motor skills include *T-ball swings*, *biped locomotion*, and on *sequential movements*. In summary, this thesis proposal is built on the following pieces of accomplished work:

1. We have shown theoretically and in experiments that the generalized Gauss' control framework with a squared metric is suitable when accurate robot dynamics models exist and connects to previous control approaches.

2. The natural actor-critic methods have yielded a variety of theoretical insights into previous reinforcement learning problems and has been successfully applied to motor primitives for simple and complex motor tasks including the T-ball swing on an antropomorphic robot arm.

3. We have presented a hierarchical framework for the representation and learning of motor skills.

Until the completion of the thesis, we intend to tackle the following problems with descending level of importance:

1. Extend the Gauss' control framework to more complicated metrics and apply it to at least one nonholomonic system. Complete the preliminary work on how to aquire such controllers using model learning.

2. Move from the policy-gradient based method for learning motor primitives towards a probabilistic policy search method which is applied both on the motor primitive as well as motor task level.

3. Use the resulting architecture to learn locomotion for legged robots.

4. If time permits, show how the probabilistic policy search techniques can be extended to infer cost functions for motor policies.

Ideally, all of these points will be accomplished in the next year; however, if necessary, point 4 can be dropped as it is not the main point of this thesis proposal.

Chapter 1

Introduction

1.1 Motivation

Despite an increasing number of motor skills exhibited by manipulator and humanoid robots, the general approach to the generation of such motor behaviors has changed little over the last decades (De Wit, Siciliano, & Bastin, 1996). The roboticist attempts to model the task as accurately as possible and uses human understanding of the required motor skills in order to create the desired robot behavior as well as to eliminate all uncertainties of the environment. In most cases, such a process boils down to recording a desired trajectory in a pre-structured environment with precisely placed objects. If inaccuracies remain, the engineer creates exceptions using his understanding of the task. Subsequently, a robot designed or purchased with the main objective that a human operator can hand-tune a control law so that the robot can precisely track the positions and velocities of the generated behavior. For doing so, the robot has to be build as a heavy mechanical structure with non-backdrivable, uncompliant joints resulting both into high-payload to weight ratio and low energy-effiency (Hirzinger et al., 2002; Albu-Schaefer, 2002). Furthermore, due to the required large torques and the stiffness of the joints, the robot itself represents a danger to its environment (Hirzinger et al., 2002; Albu-Schaefer, 2002). While such highly engineered approaches are feasible in well-structured industrial or research environments, it is obvious that if robots should ever leave factory floors and research environments, we will need to reduce or eliminate the complete reliance on hand-crafted models of the environment and the robots exhibited to date. Instead, we need a general approach which allows us to use compliant robots designed for interaction with less structured and uncertain environments in order to reach domains outside industry. Such an approach can not solely rely upon human understanding of the task but instead has to be acquired and adapted from data generated both by human demonstrations of the skill as well as the robot's trials and errors.

The tremendous progress in machine learning over the last decades offers us the promise of less human-driven approaches to motor skill acquisition. However, despite offering the most general way of thinking about data-driven acquisition of motor skills, generic machine learning techniques which do not rely upon an understanding of motor systems often do not scale into the domain of manipulator or humanoid robotics due to the high domain dimensionality. Therefore, instead of attempting a brute force, unstructured machine learning approach to motor skill aquisition, we need to develop approaches suitable for this particular domain with the inherent problems addressed separately. Such a general architecture should employ a combination of imitation, reinforcement and model learning in order to cope with the complexities involved in motor skill learning. The advantage of such a concerted approach is that it allows the main problems of motor skill learning such as skill aquisition, refinement and execution to be addressed in seperate. Instead of either having a monolithic machine learning approach or creating hand-crafted approaches with pre-specified trajectories, we are capable of aquiring skills from demonstrations and refine them using trial and error. The creation and improvement of such skills can take place through a combination of imitation and reinforcement learning. The acquired skills are represented as policies and can include specifications ranging from positions, velocities and acceleration to applied forces, stiffnesses, etc. When using learned models of the robots dynamics and kinematics for control, we often can achieve accurate control without needing to model the complete system by hand. Furthermore, robots no longer needs to be build with the sole purpose of them being straightforward to model but can be chosen to fulfill the tasks requirements in terms of compliance with the environment, energy efficiency and other factors.

In my proposed work, I plan to take motor skill learning a significant step forward and develop a general architecture for representing, acquiring and refining motor skills through a combination of imitation, reinforcement and model learning. In order for doing so, we need to develop both novel learning algorithms and control architectures. Empirically, we will show how our framework can be applied to learning different skills including teeball, box-tumbling and legged locomotion.

1.2 Objective and Approach

The principal objective which I intend to accomplish throughout my thesis can be summed up in a single question

"How can we represent, learn and execute motor skills for robotics?"

As can be observed from this question, the goal of this thesis is threefold. First, we need to find an appropriate general motor skill representation which allows us to create both basic movements as well as their combination into complex movements. Second, we need to develope scalable learning algorithms which are efficient when used with the chosen motor skill representation. Finally, the acquired skills represented need to be executed without a precise model of the mechanics. Figure 1.1 illustrates these three problem areas and their relation.

Our general approach to *motor skill representation* relies on the insight that humans, while being capable to perform a large variety of complicated movements rely upon a

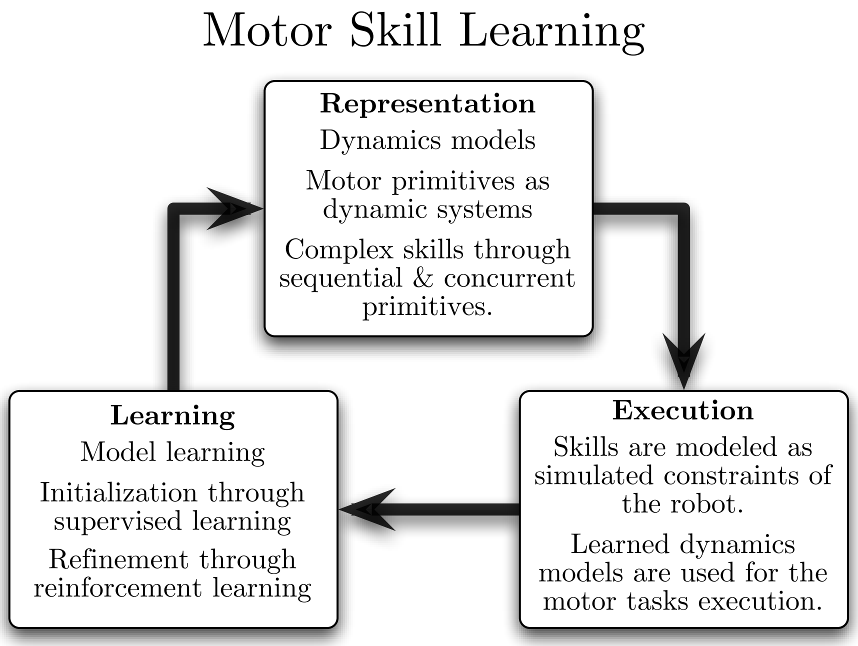

Motor Skill Learning
Representation
Dynamics models
Motor primitives as dynamic systems
Complex skills through sequential & concurrent primitives.
Execution
Skills are modeled as simulated constraints of the robot.
Learned dynamics models are used for the motor tasks execution.
Learning
Model learning
Initialization through supervised learning
Refinement through reinforcement learning

Figure 1.1: This figure illustrates our approach to motor skill learning by dividing it into a representation which is learned through reinforcement and supervised learning. Subsequently, it is executed which provides further data for reinforcement learning.

smaller amount of primitive motions (Schaal, Ijspeert, & Billard, 2004). As suggested by Ijspeert et al. (2002a, 2002b), such primitive movements can be represented by nonlinear dynamic systems. We can represent these in the form

$$\mathbf{A}_{\boldsymbol{\theta}_i}(\mathbf{x}_i, \dot{\mathbf{x}}_i, t)\ddot{\mathbf{x}} = \mathbf{b}_{\boldsymbol{\theta}_i}(\mathbf{x}_i, \dot{\mathbf{x}}_i, t), \tag{1.1}$$

where $i \in \mathbb{N}$ is the index of the motor primitive, $\boldsymbol{\theta}_i \in \mathbb{R}^L$ denote the parameters of the primitive i, t denotes time and $\mathbf{x}_i$,$\dot{\mathbf{x}}_i$,$\ddot{\mathbf{x}}_i \in \mathbb{R}^n$ denote positions, velocities and accelerations of the dynamic system, respectively. We intend to take the framework of Ijspeert et al. (2002a, 2002b) which only includes trajectory representations one step further by also incorporating force/stiffness representations into this framework. Clearly, complex behaviors can be generated by switching between motor primitives and by employing several dynamic motor primitives concurrently. Our desired motor skill representation is therefore hierachical with two layers, on the discrete top-level representation we attempt to model motor tasks using the continuous layer of dynamic motor primitives.

Learning motor skills consists out of both the sequencing and parallelization of the motor primitives as well as learning the motor primitives by adapting their parameters $\boldsymbol{\theta}_i$. The high dimensionality of our domain prohibits the exploration of the complete space of all admissible motor behaviors, rendering the application of machine learning techniques which rely upon exhaustive exploration impossible. Instead, we have to rely upon a combination of supervised and reinforcement learning in order to aquire motor skills where the supervised learning is used in order to obtain the initialization of the motor skill while reinforcement learning is used in order to improve it. Therefore, the aquisition of a novel motor task consists out of two phases. First, a human demonstration is parsed into seperate motor primitives which are compared to existing primitives in the motor primitive library; if they do not yet exist, they are added to the library. Second, the 'learning robot' attempts to reproduce the skill aquired through supervised learning and improve the skill from experience by trial-and-error, i.e., through reinforcement learning.

The *execution of motor skills* adds another level of complexity. It requires that a mechanical system

$$\mathbf{u} = \mathbf{M}(\mathbf{q}, \dot{\mathbf{q}}, t)\ddot{\mathbf{q}} + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}, t), \tag{1.2}$$

with a mapping $\mathbf{x}_i = \mathbf{f}_i(\mathbf{q}, \dot{\mathbf{q}}, t)$ can be forced to execute each motor primitive $\mathbf{A}_i\ddot{\mathbf{x}}_i = \mathbf{b}_i$ as required by the skill. A motor primitive can be viewed as a mechanical constraint acting upon the system. In this thesis proposal, we will present a novel approach for enforcing the robot to fulfill this constraint. However, in most cases it is very difficult to obtain accurate models of the mechanical system; therefore it is smarter to find a model learning approach which replaces the control law which incorporates the hand-crafted rigid body model.

In this thesis proposal, I will discuss each of these three different aspects and we will show both theoretical as well as empirical work. In Section 1.3, we will show what has been achieved and is planned until the completion of the thesis.

1.3 Major Contributions

In this thesis, we have made and intend to make progress in representing, learning and executing motor skills while demonstrating the application of this work to physical robots. This progress includes contributions to three different but related lines of research, i.e.,

- Machine Learning,
- Motor Skill Representation and Control,
- Robot Applications,

hence the name of this thesis is supposed to become "*Machine Learning for Motor Skills in Robotics*".

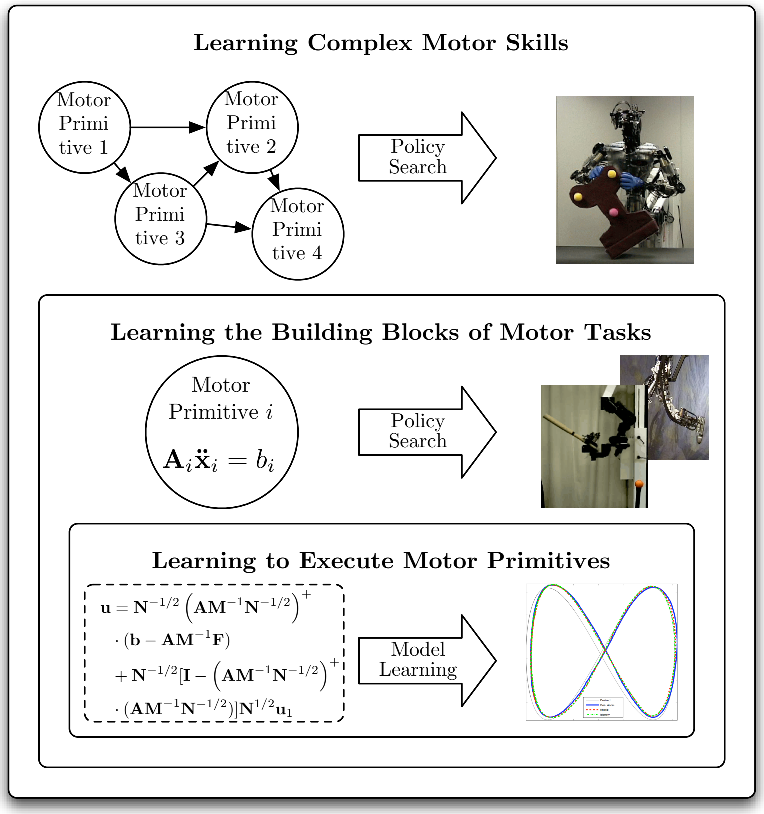

Learning Complex Motor Skills
Motor Primitive 1
Motor Primitive 2
Motor Primitive 3
Motor Primitive 4
Policy Search
Learning the Building Blocks of Motor Tasks
Motor Primitive i
$\mathbf{A}_i\ddot{\mathbf{x}}_i = b_i$
Policy Search
Learning to Execute Motor Primitives
$\mathbf{u} = \mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+} \cdot (\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{F}) + \mathbf{N}^{-1/2}[\mathbf{I} - \left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+} \cdot (\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2})]\mathbf{N}^{1/2}\mathbf{u}_1$
Model Learning

Figure 1.2: This figure illustrates the proposed three projects and their interrelation. The highest level project is the learning of complex motor tasks. These consist out of learned lower level motor primitives. The execution of motor primitives is build on learned system models.

While the general goal of bringing motor skills learning to robotics is not yet achieved, we have made significant progress including the following contributions. In the next paragraphs, we will outline three partially completed story lines which exemplify the greater goal of this thesis. Each of these sections has the basic contributions from all three disciplines and together they will result in a coherent and general way of learning motor skills for robotics.

Learning to Execute Motor Primitives. As we will show in detail in later parts of this proposal, based upon (Udwadia, 2003), we have shown in (Peters, Mistry, & Udwadia, 2005) that the generalized Gauss' control framework with a squared metric is applicable for robot control. We have discussed this theoretically and verified the general concepts by experiments. Furthermore, we have shown that it connects to various previous control approaches to trajectory control in joint and task space as well as force control.

In future work, we intend to extend the Gauss' control framework to more complicated metrics and apply it to unsolved control systems such as nonholomonic systems. Given that the Gauss' control framework is one of the most general ways how a system can be forced to fulfill multiple tasks given as dynamic systems at the same time, it can be particularly practical for the application to motor primitives as these are also represented as dynamic systems. However, as we will note in Chapter 4, the framework suffers from requiring a precise rigid body model. In practice, such a model is hardly ever given – particularly not for high-dimensional robots such as humanoid robots. We therefore need to apply machine learning techniques to this framework which replace the modeling by hand.

Learning the Building Blocks of Motor Tasks. Learning motor tasks consists of learning both the complete task (which we will discuss later) and the building blocks of motor tasks. Let us assume that we have isolated such a building block, e.g., learning a T-ball swing for baseball. If we attempted to learn such a task from scratch, we would need a huge amount of data given the high dimensionality of the robot. However, when using a combination of supervised and reinforcement learning, this problem becomes tractable. Supervised learning can be used in order to learn the motor primitives from a human demonstration while reinforcement learning is used to improve the performance by trial and error. During the last four years, we have researched policy search methods for reinforcement learning (Peters, Vijayakumar, & Schaal, 2003a, 2003b). We have obtained a variety of theoretical insights on policy search methods culminating into the policy search method category Natural Actor-Critic (Peters et al., 2003a, 2003b). Some of the Natural Actor-Critic methods have been successfully applied to learning motor primitives including the learning of the previously mentioned T-ball swing on an antropomorphic robot arm.

In future work, we intend to move away from the policy-gradient based formulation for learning motor primitives towards a probabilistic policy search method which hopefully will increase the speed of the learning while reducing the number of open learning parameters. Furthermore, a probabilistic policy search approach can be extended in order to infer cost functions of motor primitive and motor task policies.

A further topic in this area would be the learning of gaits for legged locomotion. As gaits can be learned as rythmic motor primitives, they are an appropriate application. Supervised learning on gaits for biped locomotion has already been applied (Schaal, Peters, Nakanishi, & Ijspeert, 2004; Nakanishi et al., 2004) and we intend to use our reinforcement learning techniques on improving the resulting gaits.

Learning Complex Motor Skills. Motor primitives become particularly interesting when used in combination with each other. Such combination require both the sequentialization as well as parallelization of motor primitives. To date, there is very little work in this area and we can only present a grande plan how we intend to achieve this. Similar to our previous approach to learning single motor primitives, we again intend to use a combination of supervised and reinforcement learning. The demonstrated motor skill is parsed into motor primitives in the supervised learning step; for this, the termination of the movement is determined based upon the velocity of the movement which then allows the determination of goal of the primitive. Subsequently, the seperated motor primitives are represented by dynamic systems and can be compared to existing movements. If novel, they will be added to the motor primitive library, otherwise existing primitives will be employed. Again, we will make use of reinforcement learning in order to improve the primitives as well as to reconnect these in a different sequence. However, we will not attempt to create new primitives through reinforcement learning as this problem is beyond the scope of this thesis. Two such problems are presented in Chapter 2.

Each of the projects described here uses the previous ones as basis. For learning motor skills on arbitrary robot platforms, we need the learning approach to motor skill execution. For learning complex motor skills, we need to learn the building blocks or motor primitives. This is also described by Figure 1.2.

1.4 Thesis Outline

In the remaining chapters of this thesis proposal, we attempt to give a glance at the final thesis. However, it is very likely that the different chapters will be split into smaller ones. The relation between the thesis structure and the different chapters of this thesis proposal is given in Figure 1.3.

In Chapter 2, *"Hierarchical Representation and Aquisition of Motor Skills"*, we will start by presenting comprehensible examples of complex motor skills. These examples include the usage of sequential and concurrent motor primitives and serve as our first planned motor skill learning applications. We review previous approaches to hierachical representations for learning and control in mechanical systems which includes work both in hybrid control and hierachical reinforcement learning. We use the examples then in order to present our approach to representing and learning compositions of motor primitives.

Learning techniques for motor primitives will be presented in Chapter 3, *"Policy Search for Parameterized Motor Primitives"*. We will start with a very general discussion of reinforcement learning and subsequently derive increasingly complex methods for policy search. We start with plain policy gradient methods with the gradients estimated from trial-and-error, and subsequently refine the gradient estimators in order to minimize the estimates variance and make the convergence more efficient. This results

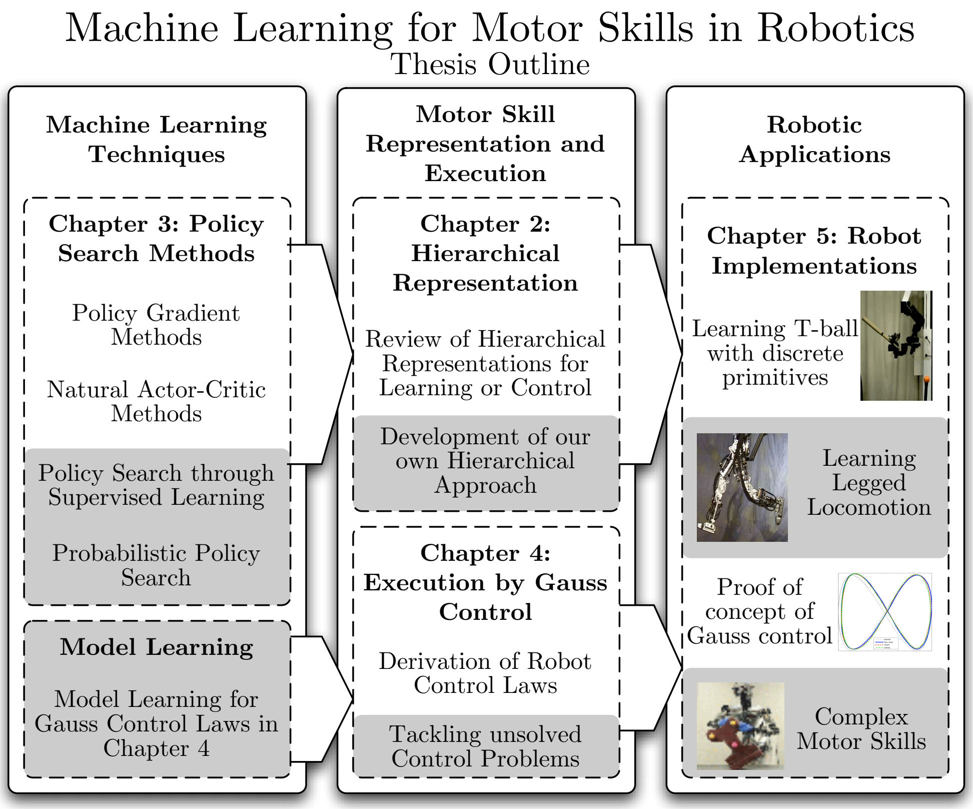

Machine Learning for Motor Skills in Robotics
Thesis Outline
Machine Learning Techniques
Chapter 3: Policy Search Methods
Policy Gradient Methods
Natural Actor-Critic Methods
Policy Search through Supervised Learning
Probabilistic Policy Search
Model Learning
Model Learning for Gauss Control Laws in Chapter 4
Motor Skill Representation and Execution
Chapter 2: Hierarchical Representation
Review of Hierarchical Representations for Learning or Control
Development of our own Hierarchical Approach
Chapter 4: Execution by Gauss Control
Derivation of Robot Control Laws
Tackling unsolved Control Problems
Robotic Applications
Chapter 5: Robot Implementations
Learning T-ball with discrete primitives
Learning Legged Locomotion
Proof of concept of Gauss control
Complex Motor Skills

Figure 1.3: This figure illustrates the proposed outline of the thesis. It illustrates the title and shows the three branches of research conducted: machine learning, motor skill representation and control as well as robotic applications. Shaded topics are still in their early stages.

into the Natural Actor-Critic algorithms which currently are the main focus of Chapter 3. However, we show preliminary work on how probabilistic methods for reinforcement learning can come into being, and discuss why these would be significant contribution to this thesis.

In Chapter 4, *"Executing Motor Skills through a Generalization of Gauss' Principle"*, we present a general framework for creating control laws for robotics based upon the Gauss' principles. We give a short perspective on future application to non-holomonic systems and to the usage of model learning for Gauss' control. In Chapter 5, *"Application to Robotics"*, we discuss the planned target applications and existing implementations which will allow my future work. Chapter 6, *"Summary and Time-Line"*, provides a time-line for my work and a summary of this thesis proposal.

Chapter 2

Hierarchical Representation and Aquisition of Motor Skills

Please note that the work presented in Chapter 2 is proposed future work despite its early appearance in the sequence of the proposal for didactical reasons. In this way, it differs significantly from Chapters 3 and 4.

2.1 Introduction

Abstraction in motor skills is familiar for everybody: whenever we perform a seemingly simple motor task, e.g., connect four points A,B,C,D without lifting the pen and in the shortest time or with the maximal smoothness, we can name the four steps of connecting these points, see Figure 2.1 (a); this problem is known as motorized traveling salesman problem (Buss, Stryk, & O., 2000). Similarly when tumbling a box without a ballistic phase (inspired by (Pollard, 2004; Pollard & Hodgins, 2002)), we can name the basic steps performed, such as e.g., pushing the box while holding against it in order to tip it, holding it with one hand while relocating the other and finally putting it down safely, just as shown in Figure 2.1 (b). While we can explain these basic steps in a discrete manner, we cannot dissect them into smaller units without nearly a complete loss of generality (Schaal, 1999; Schaal et al., 2004). Therefore, these building blocks or atoms of motor tasks form an entity by themself which we call motor primitives (Schaal et al., 2004); in the literature, these are also called 'movement schemas', 'basis behaviors', 'units of action', 'macroactions', 'options', etc. (Arbib, 1981; Dautenhahn & Nehaniv, 2002; Sternad & Schaal, 1998; Sutton & Barto, 1998)

What does this imply for a motor skill learning system? We clearly need at least two levels of abstraction, i.e., the motor task and a motor primitive level. The motor primitives form the building blocks of the system and are stored in a motor primitive library. The motor task level represents how the combination of the different motor primitives form a task together. This presents us with a variety of problems:

- How are both motor tasks and motor primitives encoded?

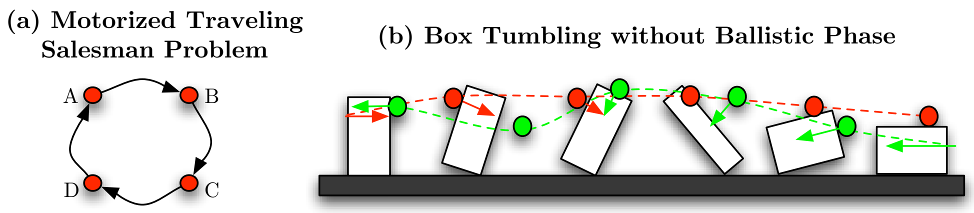

(a) Motorized Traveling Salesman Problem
A
B
C
D
(b) Box Tumbling without Ballistic Phase

Figure 2.1: This figure illustrates two skill learning problems. In (a), it shows the motorized traveling salesman problem where the robot has to connect the four points with a path which should be as smooth as possible. In (b), it shows box tumbling inspired by Pollard (2004) but without a ballistic phase.

- What kind of information needs to be presented in the motor primitives? Positions, forces, impedances? Is the information represented in the joint-space or the task-space of the robot?
- Given a human demonstration, can the presented motor skill be parsed and transferred into the hierachical representation?
- Is the representation suitable methods for improvement by trial and error?
- Can we assure the stability of the motor task if presented in the chosen representation?

Each of these questions centers around the representation of the motor skills. As the success of the applied learning techniques highly depends on the used representation, we need to be fairly careful to consider each of the topics above.

In order to be able to answer the listed questions, we review previous approaches to hierachical representation and/or learning of policies in Section 2.2. For this, we review several approaches from hierachical reinforcement learning, from hybrid discrete-continuous control and from imitation learning. Most of the hierachical reinforcement learning approaches were not directly intended for motor skills and are probably not applicable to motor skill learning as we will discuss in in Section 2.2.2. Hybrid control approaches only attempt to create environment for hand-modeling of motor skills and therefore can only offer a way of understanding how motor skills should be represented. We will give a brief overview on relevant hybrid control approaches in Section 2.2.1. Imitation learning with its goal of learning skills from a teacher provides us with a variety of interesting insights; however, as it also tackles perceptional and intelligence related issues which are beyond the scope of this thesis. We will review a selection of approaches relevant to our goal in Section 2.2.3.

Based upon the review in Section 2.2, we will outline our approach to motor skill learning in Section 3.2.1.1. This approach combines concepts from all three reviewed areas into one unified approach.

2.2 Related Frameworks

Up to today, three different fields have looked at hierachical schemes for learning or execution of tasks, i.e., Hierachical Reinforcement Learning, Hybrid Discrete-Continous Control and Imitation Learning. However, none of these three schemes has been refined to a general framework of motor skill representation, aquisition and execution. We will now give a brief review of these different approaches and outline the most important lessons.

2.2.1 Hybrid (Discrete-Continous) Control Approaches

The earliest field to discuss switching between different motor behaviours has been control theory where switching systems have been of particular importance due to the availability of inexpensive realizations consisting out of small analog circuits. While a complete survey of the field of hybrid control would be beyond the scope of this thesis, we show several lessons which can drawn from the field and we show the few approaches in hybrid control towards motor skills.

2.2.1.1 Representation

In the literature, there is a variety of different approaches for modeling hybrid systems such as a motor skills. However, on closer inspection, this variety boils down to exactly two types of approaches, i.e., (i) *differential equations with context switches* and (ii) *discrete automata with evolving states.* We will discuss these approaches based upon (Yang, 2001; Lygeros, Tomlin, & Sastry, 2005; Buss et al., 2000) in the context of motor learning and focus on skill representation and aquisition from the hybrid control point of view.

Differential Equations with Context Switch. Differential equations with context switch are the traditional way of modeling discrete events in a continuous system. Methods which discuss hybrid modeling and control problems in this form are *switching linear models* (Xu & Antsaklis, 2002), *switched bond graphs* (Edstroem, 1999), *hybrid state machines* (Buss et al., 2000) and others. For example for switching linear models (Xu & Antsaklis, 2002), the task is modeled using a continous state of the system

$\mathbf{x}(t) \in \mathbb{R}^n$ and a discrete context $\mathbf{i}(t) \in \Omega$ in some discrete event set Ω. The evolution of both can be specified by

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{i}(t^-)), \tag{2.1}$$

$$\mathbf{i}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{i}(t^-)), \tag{2.2}$$

where $\mathbf{i}(t^-) = \lim_{\tau \to t^-} \mathbf{i}(t)$ denotes the context in the instant of time before the context switch. When additionally a control variable $\mathbf{u}(t)$ is added to $\mathbf{f}$, $\mathbf{g}$, we obtain the hybrid state machines (Buss et al., 2000) for system modeling.

How could one express a motor skill such as the ones described in Section 2.1? The motorized salesman skill in Figure 2.1 (a) is fairly easy to express as $\mathbf{i}(t)$ would simply denote the next point to visit and $\mathbf{x}(t)$ could denote either the position of the robots endeffector or the position in joint space. It is convenient to model of motor task if a clear coordinate system is given and methods from system theory can be applied with ease. Nevertheless, for the modeling and synthesis of motor skills, this approach has a major deficiency: the system has to remain in its chosen representation at all time – a switch between a joint space and a task space model is not trivial to incorporate. However, such switches are essential for motor skills which often require the switch between different subtasks and their required object related representations. Furthermore, not all possible discrete event properties can be incorporated, e.g., discontinous event properties can create problems (Yang, 2001).

Discrete Automata with Evolving State. As an alternative to the traditional way of thinking about hybrid systems, it has been suggested to model hybrid systems as discrete event systems where each state can be described by the evolution of continuous variables. As the continuous variables can be chosen differently in each state, our previous concern about the switch in representations is addressed in this framework by definition. A variety of different discrete automata with evolving state have been suggested in the literature including *hybrid automata* (Henzinger, 1996; Yang, 2001), *hybrid I/O automata* (Lynch, Segala, Vaandrager, & Weinberg, 1996), *phase-transition systems* (T.Henzinger & Toi, 1996) and *hybrid Petri-nets* (Febbraro, Giua, & Menga, 2001). The general principle behind these approaches is all the same: they use a discrete event system and turn the states into dynamic systems. We will describe it using the example of a hybrid automaton (Henzinger, 1996; Yang, 2001). A hybrid automaton H is a collection

$$H = (\mathbb{Q}, \mathbb{X}, \theta_0, \mathbf{f}, \Gamma, G, R), \tag{2.3}$$

where $\mathbb{Q} = \{1, 2, 3, \ldots, n\}$ is the set of the discrete states, $\mathbb{X} \subseteq \mathbb{R}^n$ is the set of continous states, $\theta_0 \subseteq \mathbb{Q} \times \mathbb{X}$ is the set of start states, $\mathbf{f} : \mathbb{X} \times \mathbb{Q} \to T\mathbb{X}$ describes the continuous evolution of the state, $\Gamma : \mathbb{Q} \to \mathbb{X}_q$ assigns eacht state $q \in \mathbb{Q}$ 'invariant' subset of the state space $\mathbb{X}_q \subseteq \mathbb{X}$, $\mathbb{E} \subseteq \mathbb{Q} \times \mathbb{Q}$ denotes the set of edges of the automaton, $G : \mathbb{E} \to \mathbb{X}$ is called the guard set which maps the edges of the automaton onto continuous state space, and $R : \mathbb{X} \times \mathbb{E} \to \mathbb{X}$ resets the continuous state when a new discrete state is entered.

If additionally an action u and an output y are included, we can obtain a hybrid I/O automaton (Lynch et al., 1996).

From the description of the hybrid automaton, it is apparant that cost of discrete automata with evolving state is they usually require a significantly more complex description and modeling of physical systems is not that straightforward. However, model and controller verification, controllability analysis and synthesis of controls can be treated significantly easier (Yang, 2001). Additionally, modeling motor tasks can be done in a significantly more general way than before and they do open a bridge to methods from computer science and artificial intelligence.

2.2.1.2 Aquisition

The lessons for the aquisition of motor skills which we can learn from hybrid control is fairly limited. To date, there are few general building principles which allow automatic modeling or synthesis of behaviors. In most cases, modeling the system is done by hand; sythesis of behaviors is either achieved by hand or using optimization techniques.

Sequential Composition. According to (Yang, 2001; Lygeros et al., 2005), there are just few major approaches to sequential composition of behavior. In the simplest case, *supervisory discrete event control* (Lemmon, He, & Markovsky, 1999; Ramadge & Wonham, 1987), a supervisory layer of discrete events can be created which create discrete actions which force the continuous state to evolve. The supervisor is usually created using human knowledge. Another approach is *state-space quantization* (Caines & Wei, 1998) where the discrete behavior is specified and then continous control laws are chosen which enforce the desired transitions. Two interelated approaches can be found in the literature, i.e., *gain scheduling* (Leithead, 2000) and *backchaining* (Lozano-Perez, Mason, & Taylor, 1984; Burridge, Rizzi, & Koditschek, 1999). Gain scheduling uses local gains in different discrete states in order to enforce the transition between states and comes with few mechanism for creating control laws. Stability can be assured using piecewise quadratic Lyapunov functions (Johansson & Rantzer, 1998) or multiple Lyapunov functions (Branicky, 1998). Backchaining has a single goal region and a set of local control laws; chaining back from the goal region it creates a hierachy of local control laws which can be visualized as a tree of funels leading into the goal (Burridge et al., 1999). If either the system has to pass through an exit region of the funnel or if the funnels can compose a global Lyapunov function, stability can be guaranteed (Burridge et al., 1999). In some cases, it is desirable to create a hierachy of hybrid I/O automatons where one automaton models the system while the other models a control policy (Yang, 2001).

Optimization Approaches. While the application of optimal control to hybrid systems is obviously of large interest, there have been only few approaches in the literature which can efficiently create optimal controls for hybrid systems (Johansson, 2000). In most cases, the discrete is treated using dynamic programming or branch-and-bound while the continuous layer is optimized using open-loop trajectories or controls based upon Pontriyagin's Maximum principle (Bryson, 1981), in some cases stabilized using feedback (Johansson, 2000; Buss et al., 2000; Branicky & Mitter, 1995). The later problem of the open-loop controls is particularly difficult; the currently most successful method appears to be sparse direct collocation (Stryk, 2000, 1999; Hardt & Stryk, 2000; Buss et al., 2000). The only alternatives in the literature are the optimization of a discrete system which obviously fails to generalize for high-dimensional systems due to the 'curse of dimensionality' (Bellman, 1957) and the parametric optimization with respect to the parameters of the hybrid system or control law.

2.2.2 Hierarchical Reinforcement Learning

Another area of research which has studied hierachical task learning is reinforcement learning. However, in this field there has been little work on continuous time systems – lower levels in the hierachy are usually represented by more finely discretized models or control policies. Unlike in hybrid control, reinforcement learning researchers are rarely just interested in a desired trajectory for the evolution of the states but instead attempt to find complete control policies. Based upon (Barto & Mahadevan, 2003), we will now discuss the different approaches to hierachical reinforcement learning in a fairly coarse manner. Again, we will first describe the representation and subsequently the aquisition of tasks.

2.2.2.1 Representation

In the hierachical reinforcement learning literature to date, three approaches have been dominant (Barto & Mahadevan, 2003), i.e., (i) *options framework* with its *multi-step action representation* (Sutton, Precup, & Singh, 1999), (ii) *hierachy of abstract machines* (Parr & Russell, 1998), and (iii) *MAX-Q value function decomposition* with *taskgraph representations* (Dietterich, 2000). Each of these approaches attempts to create a representation similar to hybrid automata consisting out of layers of representation, however with a fine grid instead of a continuous level and a higher number of layers.

Multistep Action Representation by Options. The general goals of the options framework is to create a natural way of representing multi-step actions (Sutton et al., 1999), i.e., options, so that the option can be treated nearly like a single-step action (Sutton et al., 1999; Barto & Mahadevan, 2003). In general, an option is given by

$$o = (\mathbb{X}_o, \mu_o, \beta_o) \,, \tag{2.4}$$

where $\mathbb{X}_o \subseteq \mathbb{X}$ denotes the states where the option o is applicable, $\mu_o(o|x)$ denotes the probability distribution over available options given the current state of the system x or policy (it usually is semi-Markovian, meaning that the policy can have an internal state during execution of the option or a dependence of time), and $\beta_o(x) \to [0,1]$ denotes the termination probability of the option. For all $x \in \mathbb{X}_o$, we usually have $\beta_0(x) < 1$, i.e., no option terminates before starting.

The execution of an option works as follows: the policy of the current option chooses an option o' to execute by sampling from the policy μ_o based upon the current state and their applicability. This option in turn calls other options until its termination when it gives control back to the higher level option (Barto & Mahadevan, 2003).

Hierachy of Abstract Machines. The hierachy of abstract machines approach is probably the closest to hybrid automata with supervisory discrete event systems among the approaches in this chapter. Similar to a hybrid automaton, we have a hierarchy of where each state of the higher level automaton is a lower level automaton by itself with main difference that many levels are possible and continuous time or state are not considered (Parr & Russell, 1998). The main problem is the automatic creation of state machines which usually has to be done by handcrafting. Note the conceptional difference to options: while options extend the amount of possible choices, hierachies of abstract machines tend to limit the choices. Recently, the resulting limitations have been addressed with the programable hierachies of abstract machines which also have local variables, parameters as actions as wells as interrupts and aborts (Barto & Mahadevan, 2003).

Taskgraph Representation. Taskgraph representations decompose the task M into subtasks M_i with $M = \{M_0, M_1, \ldots, M_n\}$ and have mostly been used in the context of the MAX-Q algorithm (Dietterich, 2000). The root task is the task M_0 and each following task is a subtask. A subtask considers a group of the lower tasks its subtasks forming a hierachy of tasks. Each subtask has policy π_i which allows it to choose among its own subtasks, it has a set of active states $\mathcal{S}_i$ and a set of termination states $\mathcal{T}_i$ which cause the policy to terminate this subtask, and a pseudo-reward function which assigns each task reward values. The resulting hierachy of subtasks forms a so-called task graph (Dietterich, 2000).

2.2.2.2 Aquisition

The acquisition of behaviors is highly driven by the chosen framework and to date has not yet converged to a clearly preferable method. Nevertheless, in order to find the good motor skill learning system, we can learn significant lessons from hierarchical reinforcement learning.

Options generation and learning. Currently, the designer of the learning system creates the options using 'primitive actions'[1] and handcrafted sets options based on prior knowledge (Barto & Mahadevan, 2003). The primary advantage of this approach is that temporally extended actions can be added easily into the repertoire of the learner which can increase the learning speed significantly. To date, all skill acquisition methods for options are based upon dynamic programming or Q-learning. For most cases, an option is only updated upon its termination. However, for Markov decision problems special cases exist where all options which would take that action in that state are updated.

A fairly interesting line of research among options framework is the learning of subgoals (McGovern & Barto, 2001), i.e., regions which the agent has to pass through, similar to a door through which a human has to go in order to move through a new room. Here, hierarchical reinforcement learning touches with hybrid control as subgoal achievement can be achieved through Lyapunov functions design (Perkins & Barto, 2001). Another interesting aspect about options is the concurrent options framework which allows the parallel activation of multiple options.

Learning in Hierachies of Abstract Machines. The hierachy of abstract machines are usually learned using Q-learning on the different levels of the hierarchies. This means the rewards accumulated in the lower level machine are used to improve the lower level machine and their accumulation is passed on to the next higher machine in the hierarchy (Parr & Russell, 1998; Barto & Mahadevan, 2003).

MAX-Q Value Function Learning in Task Graphs. The task graph with its subtask, termination and active state sets as well as pseudo-rewards is usually handcrafted, requiring a lot of prior human knowledge on the task. In order to learn the policies of the subtasks, Dietterich (2000) adapted Q-Learning for this problem to a special form called MAX-Q. The pseudo-rewards make this framework particularly powerful as it allows the specification of subgoals without specifying how to achieve such subgoals.

2.2.3 Approaches from Imitiation Learning

Several other approaches to complex movement recognition and movement generation have been suggested inspired by statistical data analysis. Pook and Ballard (1993) created a robotic system that initially discretized demonstrated movement and force trajectories by means of vector quantization, and then trained a hidden Markov model (HMM) with these discrete states as observables to find the transition probabilities among a given set of movement primitives. An experimental implementation could be realized for one complex task (egg flipping), albeit generalization of the suggested methods to different tasks or changes in the environment was not further explored.

[1]The term primitive actions is used differently in the options framework than in motor skills. Reinforcement learning researchers mean the basic actions executable by the system while we usually mean the atoms of motor skills.

Inamura et al. (2002, 2004) followed a similar approach using HMMs. A movement primitive was defined in terms of Gaussian trajectory clusters in joint position space, and the HMM methodology was employed to recognize sequences through these primitives. The transition probabilities between primitives could afterwards be used to generate movements by sampling trajectories from the stochastic HMM. As motor primitives in terms of Gaussian clusters in joint space can be created automatically, the authors suggested that this method might be some form of protosymbol formation, which could be exploited for the purpose of bootstrapping communication in future work. One more HMM approach to movement sequence recognition and movement sequence generation was suggested by Amit and Mataric (2002). Movement primitives were assumed to be given a priori, and a two stage learning system associated goals with each primitive and sequences through the set of primitives.

Inspired by theories of computational motor control, Miyamoto et al. (1995, 1996, 1996, 1998) employed a movement parameterization in terms of 5th-order time-dependent splines, and characterized movement primitives based on typical sequences of spline nodes. Such sequences can be detected in observed actions by first fitting them with splines, and then searching the spline node representation for known primitives (Wada & Kawato, 1994; Kawato, Gandolfo, Gomi, & Wada, 1994; Wada & Kawato, 1995; Miyamoto et al., 1996, 1996). The spline representation lends itself naturally to movement generation in an optimal control framework (Kawato, 1999). Robustness of this method towards spatial and temporal scaling of movements, however, was not further explored. Miyamoto et al.'s work was a precursor for a more refined approach of a reciprocally constrained system of movement recognition and movement generation, suggested by Wolpert and Kawato (1998) and adapted for sequence learning in Samejima et al. (Doya, Samejima, Katagiri, & Kawato, 2002; Samejima, Doya, & Kawato, 2003). Reinforcement learning was employed to create complex movements from basic control primitives, and action recognition was possible with the help of predictive forward models. A limitation of this approach may lie in its slow learning performance due to the inherent limitation of the current state-of-the-art reinforcement learning algorithms.

Instead of HMMs, motor sequences can also be encoded in recurrent neural networks, i.e., neural network with closed loopy connectivity. Paine and Tani (2004) trained a well-designed recurrent network system on observed movement data such that the same network could represent multiple primitives depending on the setting of certain parameters. The network topology also supported the development of specialized neurons, called mirror neurons, which classified which motor primitive was currently active. After training, when exposed to new movement observation, these mirror neurons could be used to automatically parse the observed action into motor primitives, i.e., periods of quasi-constant activation of these mirror neurons. Another recurrent network approach was proposed by Billard (Billard & Mataric, 2001), who employed integrate-and-fire neurons to code movement primitives in the spirit of associative memory neural networks. Teaching and recognition of complex movement on humanoid robots could be realized in this fashion (Billard & Schaal, 2002; Schaal et al., 2004)

2.3 Towards a General Framework for Motor Skill Learning

Building upon previous work (Peters et al., 2003a; Schaal, Peters, Nakanishi, & Ijspeert, 2003; Schaal et al., 2004), and based upon our review of related approaches, we outline steps towards a general framework for motor skill learning in this section. The main structural lessons which we can draw from the previous sections on hybrid control, hierachical reinforcement learning and imitation learning approaches are as follows. We need a three layered approach where: (i) the execution layer ensures the stable execution of the movement, (ii) the primitive action layer creates continuous building blocks in form of dynamical systems and (iii) a discrete layer for the sequential, concurrent and hierarchical composition of primitive motor actions. In order to make this approach general, we will need feasible approaches for fast learning at each of these levels. We will illustrate this general framework in a top-down manner.

2.3.1 Motor Tasks

Let us assume that we are given a set of motor primitives

$$\mathcal{M} = \{M_1, M_2, \ldots, M_n\}, \tag{2.5}$$

where each motor primitive M_i is described a dynamic system as outlined in Section 2.3.2 or (Ijspeert et al., 2001, 2003). Our representation is discrete and can reacts towards perceptual inputs. We intend to make use of two basic mechanisms for task generation, i.e., superposition and sequencing, and use lessons from both imitation learning as well as hierachical reinforcement learning for motor task aquisition. An example for superposition and sequencing in motor skills is shown in Figure 2.2.

2.3.1.1 Concurrent and Sequential Compositions of Motor Primitives

The relation to hybrid control is obvious, i.e., it is discrete automaton with evolving states; however, which specific type of automaton is still left as a choice to us. From the perspective of hierachical reinforcement learning, we can treat motor primitives either as primitive actions in an options framework or as abstract machines which are the states of a hierachy of abstract machines (while the taskgraph representation is inapplicable as the motor primitives are indivisible from the viewpoint of the discrete layer). However, the options framework appears to be well-suited for motor tasks as we can create options by superposition of motor primitives. Therefore is creation of options in motor skills easier than its counterpart for higher order options or discrete temporally extended options.

Superposition of Motor Primitives. In most biological systems, there are usually multiple motor behaviors being executed in parallel. Among the most intuitive examples is bipedal posture control, during which humans accomplish all kinds of other tasks, like

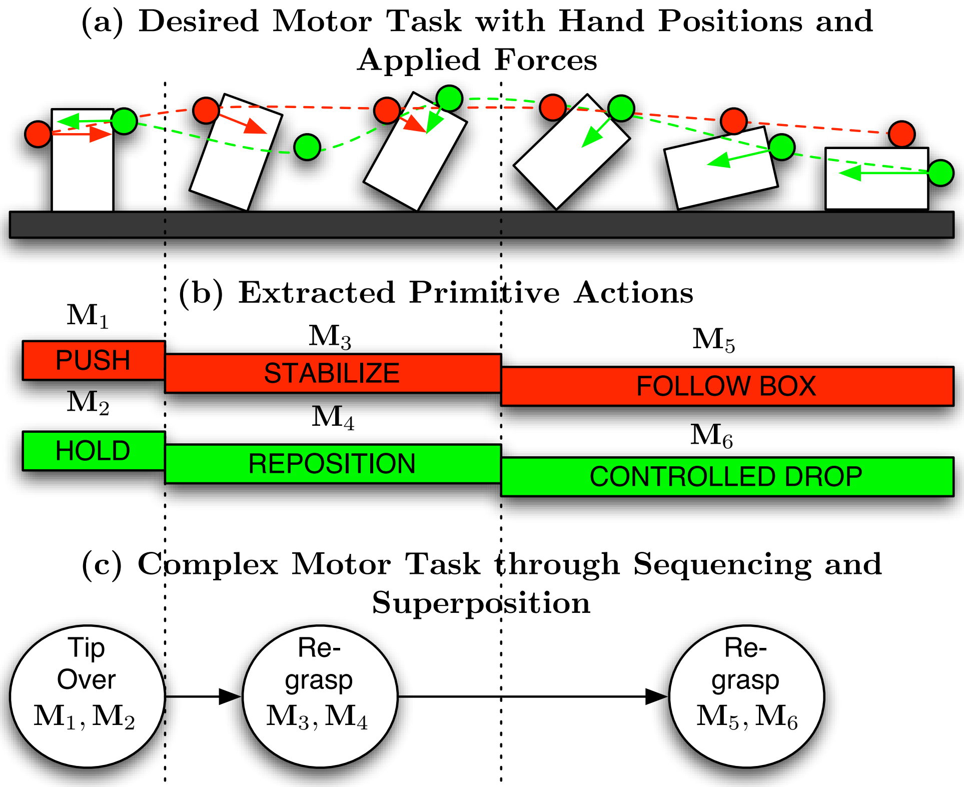

(a) Desired Motor Task with Hand Positions and Applied Forces
(b) Extracted Primitive Actions
$\mathbf{M}_1$
PUSH
$\mathbf{M}_2$
HOLD
$\mathbf{M}_3$
STABILIZE
$\mathbf{M}_4$
REPOSITION
$\mathbf{M}_5$
FOLLOW BOX
$\mathbf{M}_6$
CONTROLLED DROP
(c) Complex Motor Task through Sequencing and Superposition
Tip Over $\mathbf{M}_1, \mathbf{M}_2$
Re-grasp $\mathbf{M}_3, \mathbf{M}_4$
Re-grasp $\mathbf{M}_5, \mathbf{M}_6$

Figure 2.2: Superposition and sequencing of motor primitives are being illustrated in this example. Each hand movement creates a motor primitive $\mathbf{M}_i$. Together these primitives form a motor task automaton $\mathcal{M}$.

grasping and manipulating objects, head movement for active perception, or esthetic movements, as for instance in ballet. Obviously, some of these superimposed behaviors can actually interfere with each other (e.g., reaching for an object while maintaining bipedal balance), while others do not, like head and eye movements, that can largely be considered independent of the rest of the body.

As one of the simplest superposition tasks, we will investigate whether two movement primitives can be executed in parallel, which, assuming both primitives are in the same coordinate system (e.g., joint space), is technically quite trivial in our motor primitive framework by simply using the average outputs of two movement primitives as the target signal. When the two motor primitives have seperate coordinate systems, the main question is whether they can be action in parallel or whether they require a trade-off among themselves. For these more advanced interactions, the motor primitives can be treated as hierachically ordered constraints ranked according to their importance,

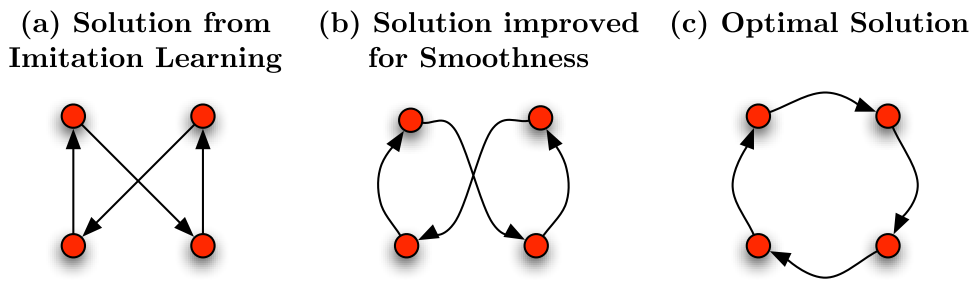

(a) Solution from Imitation Learning
(b) Solution improved for Smoothness
(c) Optimal Solution

Figure 2.3: This figure shows different solutions for the motorized traveling problem where the robot has to connect the points in a movement with maximal smoothness. In (a), we see the solution learned by imitation. The imitated solution from (a) is improved by optimizing both motor primitives (i.e., the goal of the primitive and its parameters) as well as transitions between the primitives. The result is smoother but not optimal as shown in (b). The optimal solution would also need a reconnecting of the motor primitives and is shown in (c).

e.g., posture control is the highest objective, reaching the goal is the second highest objective, esthetics the third highest ob-jective, etc. Given the highly redundant human motor system, it is possible to select the contribution of different degrees of freedom for each task objective such that interference is limited. Similar to the approach of Sentis and Khatib (Sentis & Khatib, 2004), we can create hierachical orderings also in the Gauss' framework in Chapter 4.

Sequencing of Motor Primitives. For creating complex sequential motor tasks, we can model the problem as solving a semi Markov decision process (MDP) where the motor primitives are the internal state of the policy or the current option. In either case, we attempt to basically model the motor skill as a discrete automaton with evolving states. This opens a major question, i.e., what kind of a representation shall be chosen for the discrete level? The simplest way would be to use a finite state machine on the discrete level. However, when used with superposition of motor skills, this kind of an automaton has an exponential explosion of states (Henzinger, 1996) and might therefore not be applicable. Probably the smartest way to represent it using either a non-deterministic finite state machine or a hybrid Petri-automaton; the later appears particularly suited for sequencing primitives.

2.3.1.2 Aquisition through a Combination of Supervised and Reinforcement Learning

When shown a complex motor task, how can we learn this motor task into the chosen representation? As we have not made our discrete layer completely fixed, we can only outline

Aquisition by Supervised Learning. A complex movement observed by the motor learning system can be decomposed into a series of concatenated discrete primitives as illustrated in Figures 2.1 and 2.3. Without co-articulation, primitives can be separated by simultaneous zero velocity and zero acceleration crossings or an instant, discontinuous sign change in both velocity and acceleration. With co-articulation, the first scenario has to be relaxed towards some thresholding which will be needed in terms of what is acceptable as a velocity and acceleration value that is close enough to zero. If co-articulation is high, it is likely that such parsing into movement primitives will produce errors. More complex movements may be created out of a sequence of different rhythmic behaviors. A change in amplitude and/or frequency should allow telling these behaviors apart; co-articulation is not likely to be a problem as the transient between primitives is much shorter than a normal periodic behavior. Third, a complex movement may have discrete and rhythmic parts. Those should be separable based on frequency analysis and duration. Finally, there may be superimposed discrete and rhythmic movement primitives in the complex movement. When analyzing such data in the Fourier domain, a subtraction of the most basic Fourier terms from the trajectory should uncover the discrete movement. All the ideas await yet an algorithm and experimental realization, and there has been no previous work to examine such methods.

Improvement through Reinforcement Learning. Reinforcement learning can be used to refine execution of the sequence of motor primitives in terms of both refining the motor primitive itself, by refining the transition to the next motor primitive and by finding new paths through this motor primitive graph in order to achieve improved behavior. The first two of these problems are easier as they just require the task to be changed locally as illustrated in Figure 2.3 (a,b) while reconnecting is more difficult as illustrated in Figure 2.3 (c). Motor task refinement through reinforcement learning of motor primitives has been developed (Schaal et al., 2003, 2004; Peters et al., 2003b, 2003a) and will be outlined later.

From the viewpoint of dynamic movement primitives, the transitions to another motor primitive could be based on the behavioral phase variable, such that learning needs to determine at which value of the phase to trigger the next primitive this is a typical component of learning with abstract actions in MDPs (Barto & Mahadevan, 2003). E.g., if time optimality were required and it is not important to fully achieve the subgoal of each primitive, one can easily image some form of co-articulation in the execution of the sequence that smoothes the primitives more and more together. Such a process would not be unlike human learning of complex motor skills. The motorized travelling salesman problem in Figure 2.3 (a-b) illustrates an example of the smoothing together of motor primitives in the task of drawing simple figures of few strokes. Future work will be needed to fully flesh out how to apply the framework of reinforcement learning to the learning of motor sequences with movement primitives, and also how to address the finding new paths through this motor primitive graph, which is the most challenging.

2.3.2 Motor Primitives Level

A dynamic motor primitive represents the desired state either in joint- or task-space of the robot or in the space of the actual task; in either case, the state of the motor primitive is specified by some appropriate variable $\mathbf{x}_i$, which can represent kinematic variables, i.e., desired positions, velocities, and accelerations. However, for manipulation tasks, it might be useful to include variables related to contact forces. In either way, a motor primitive M_i is given by $M_i = (\mathbf{A}_{\theta_i}(\mathbf{x}_i, \dot{\mathbf{x}}_i), \mathbf{b}_{\theta_i}(\mathbf{x}_i, \dot{\mathbf{x}}_i), \ddot{\mathbf{x}}_i)$, or equivalently by

$$\mathbf{A}_{\theta_i}(\mathbf{x}_i, \dot{\mathbf{x}}_i)\,\ddot{\mathbf{x}}_i = \mathbf{b}_{\theta_i}(\mathbf{x}_i, \dot{\mathbf{x}}_i). \tag{2.6}$$

For the representation of such motor primitives, we follow Ijspeert et al. (2001, 2003) as well as for the imitation of given movements. The reinforcement learning of these primitives is outlined in Chapter 3 as well as in (Peters et al., 2003b, 2003a; Schaal et al., 2003, 2004) .

2.3.2.1 Motor Primitive Representation by Dynamical Systems

In order to accommodate discrete and rhythmic movement plans, two kinds of DMPs are needed: point attractive systems and limit-cycle systems (Ijspeert et al., 2001, 2003). The key question of DMPs is how to formalize nonlinear dynamic equations such that they can be flexibly adjusted to represent complex motor behaviors without the need for manual parameter tuning and the danger of instability of the equations. We will sketch our approach in the example of a discrete dynamic system for reaching movements, the analogous development for rhythmic systems can be found in (Ijspeert et al., 2001, 2003).

Assume we have a basic point attractive system, instantiated by the second order dynamics

$$\tau\dot{z} = \alpha_z(\beta_z(g-y)-z), \; \tau\dot{y} = z + f, \tag{2.7}$$

where g is a known goal state, α_z and β_z are time constants, τ is a temporal scaling factor and y, $\dot{y}$ correspond to the desired position and velocity generated by Equation (2.7). These two variables can be interpreted as a movement plan and could for example be the desired states for a one degree-of-freedom motor system. Without the function f, Equation (2.7) is nothing but the first order formulation of a linear spring-damper system. For appropriate parameter settings and $f = 0$, these equations form a globally stable linear dynamic system with g as a unique point attractor, which means that for any start position the limb would reach g after a transient, just like a stretched spring, upon release, will return to its equilibrium point. The key goal, however, is to instantiate the nonlinear function f in Equation (2.7) to change the rather trivial exponential and monotonic convergence of y towards g to allow trajectories that are more complex on the way to the goal. As such a change of Equation (2.7) enters the domain of nonlinear dynamics, an arbitrary complexity of the resulting equations might be expected. To the best of our knowledge, this problem has prevented research from employing generic learning in nonlinear dynamic systems so far. In order to force Equation (2.7) to become more complex while avoiding explicit functions of time and direct feedback of f from , we need an additional dynamic system (x, v) with

$$\tau \dot{v} = \alpha_v \left(\beta_v \left(g - x \right) - v \right), \tau \dot{x} = v, \tag{2.8}$$

and the nonlinear function approximator f in form of

$$f(x, v, g) = \frac{\sum_{j=1}^{N} \psi_j \theta_j^i v}{\sum_{j=1}^{N} \psi_j}, \tag{2.9}$$

where $\psi_j = \exp\left(-h_i \left(x/g - c_i \right)^2 \right)$ are the basis functions of the function approximator and $\boldsymbol{\theta}^i$ the parameters of the approximator.

Equation (2.8) is linear spring-damper system similar to Equation (2.7), however, not modulated by a nonlinear function we will call this equation the canonical system from now on, as it is among the most basic dynamic system available to create a point attractor. The monotonic global convergence of Equation (2.7) to g can be guaranteed with a proper choice of α_v and β_v, e.g., in the same manner that a damped spring returns to its equilibrium point no matter from which stretched position it is released.

In summary, by anchoring a linear learning system with nonlinear basis functions in the phase space of a canonical dynamic system with guaranteed attractor prop-erties, we are able to define complex movement behaviors as an attractor of non-linear differential equations without endangering the asymptotic convergence to the goal state. Both discrete and rhythmic movements can be coded in the DMPs, and almost arbitrarily complex (but smooth) trajectory profiles are possible. By modifying the goal parameter (or amplitude parameter in rhythmic movement), and the overall time constant of the equation, variants of the same movement can be generate, i.e., the movement primitive can be re-used for temporal spatial task variation. This strategy opens a large range of possibilities to create movement primitives, e.g., for reaching, grasping, object manipulation, and locomotion. In particular, imitation learning as well as trial-and-error learning methods can easily be built around this approach, as outlined in the next section and Chapter 3.

2.3.2.2 Motor Primitive Aquisition by Imitation and Refinement

After developing the formal idea of DMPs in the previous sections, we will now turn to imitation learning. In fact, imitation learning with DMPs is technically rather straightforward, which is not surprising, as the theory of DMPs was built with imitation learning in mind. If we assume, for simplicity, that the number of basis functions in Equation (2.9) are known, the crucial parameters of a movement primitive are the weights θ^i_j in the nonlinear function f, as they define the spatiotemporal path of a DMP. Given that f is a normalized basis function representation, linear in the coefficients of interest θ^i_j, see e.g., (Bishop, 1995), a variety of learning algorithms exist to find θ^i_j. In an imitation learning scenario, at an advanced level of preprocessing of sensory data about the teachers demonstration, we can suppose that we are given a sample trajectory with duration T. Based on this in-formation, a supervised learning problem results with the target for f given by

$$f_{\text{target}} = \tau \dot{y}_{\text{demo}} - z_{\text{demo}}, \tag{2.10}$$

where $\tau \dot{z}_{\text{demo}} = \alpha_z(\beta_z(g - y_{\text{demo}}) - z_{\text{demo}})$.

In order to obtain a matching input for f_{target}, the canonical system needs to be integrated. For this purpose, in Equation (2.9), the initial state of the canonical system is set to $v = 0$, $x = y_{\text{demo}}(0)$ before integration. An analogous procedure is performed for the rhythmic DMPs. The time constant τ is chosen such that the DMP with $f = 0$ achieves 95% convergence at $t = T$. With this procedure, a clean supervised learning problem is obtained over the time course of the movement to be approximated with training samples $(\boldsymbol{v}, f_{\text{target}})$ and the solution can be obtrained by modern regression techniques.

Imitation then yields the first shot for the motor primitive it is clear that many tasks cannot be learned this way. We therefore need reinforcement learning for reward-related self-improvement. This will be tackled in Chapter 3 in detail and we will therefore omit this part here.

Chapter 3

Policy Search for Parameterized Motor Primitives

3.1 Introduction

A central problem of this thesis is the aquisition and reward-related refinement of motor skills from trial and error, i.e., through reinforcement learning. Given a hierachical motor skill representation for motor skills such as the one discussed in Chapter 2, we need appropriate methods for adapting the parameters of both motor tasks and motor primitives in order to improve the resulting behavior with respect to a cost. In this chapter, we will focus on the aquisition of motor primitives; however, this does not limit the presented methods which can also be applied to motor tasks or even parameterized problems not related to motor control.

Throughout this chapter, we will proceed as follows: first, in this section, we will discuss the relation of reward-related motor skill aquisition to reinforcement learning. We start with a definition of reinforcement learning and then expand towards policy search methods.

In Section 3.2, we discuss the theory of "vanilla" or "heuristic" policy gradient techniques from a roll-out perspective; we derive both previous as well as novel methods for the estimation of gradients. After noting problems with premature convergence and fragility towards parameter changes for the "vanilla" policy gradient, we realize that the steepest descent has to occur not with respect to the policy parameters but with respect to the probabilities of the actions represented by the policy. This leads to the main realization in Section 3.3, that the "natural" or "covariant" gradient can improve here and therefore ease the resulting problems. We show that several previously successful reinforcement learning approaches are related or even a direct application of the natural policy gradients. In Section 3.3.2.4, we show how these methods can be applied to learning with motor primitives.

One of the major drawbacks of gradient methods is that the length of the step along the gradient, i.e., the learning rate, is an open parameter of large importance which can decide over success or failure. Probabilistic methods such as the EM algorithm allow the elimination of such parameters – we therefore show preliminary work on the question "Can we find a probabilistic policy search method?" in Section 3.4.

3.1.1 The Reinforcement Learning Framework

Reinforcement learning is one of the most general frameworks for reward-related aquisition and refinement of policies from data. In order to demonstrate that motor skills can be thought of as policies in a reinforcement learning problem in Section 3.1.2, we first introduce the basic reinforcement learning definitions and problem statements with a focus on motor control in Sections 3.1.1.1 and 3.1.1.2. Furthermore, we discuss which class of reinforcement learning techniques is applicable to motor learning problems in Section 3.1.1.3.

3.1.1.1 Basic Setup

In reinforcement learning, we generally assume problems which can be seperated into a *system*[1] and an *actor*. At any instant of time, the system is at a *state* $\mathbf{x} \in \mathbb{X}$ and the actor can apply an *action* $\mathbf{u} \in \mathbb{U}$ on the system. In reaction to the action, the system transfers to a next state $\mathbf{x}' \in \mathbb{X}$ and will yield a *reward* $r \in \mathbb{R}$. As most motor learning problems require continuous states $\mathbb{X} \subseteq \mathbb{R}^N$ and actions $\mathbb{U} \subseteq \mathbb{R}^M$, we develop our work in a continuous state-action context but use a discrete-time approach due to the final implementation on a digital computer. In this scenario, we can describe the system by a state transfer probability density distribution and a reward distribution while the actor is given in terms of a policy.

The state transfer distribution is given by $\mathbf{x}_{k+1} \sim p\left(\mathbf{x}_{k+1} | T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k}), k\right)$ where $T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k})$ denotes the sufficient statistics[2] of all previous states $\mathbf{x}_{0:k}$ and actions $\mathbf{u}_{0:k}$ relevant to the state transfer and k denotes the current time-step. The policy of the actor is given by a probability distribution

$$\mathbf{u}_k \sim \pi_{\boldsymbol{\theta}}(\mathbf{u}_k | T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k) = p(\mathbf{u}_k | T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k, \boldsymbol{\theta}), \tag{3.1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^L$ denote the L parameters of the policy $\pi_{\boldsymbol{\theta}}$; its optimization is the central element of all sequential planning techniques such as reinforcement learning. The policy is written here as a probability distribution in order to indicate that exploration of new actions takes place[3]. The noisy reward is distributed in accordance to a reward

[1]The system is also often called the environment.

[2]The sufficient statistics represent our knowledge on the current state. In the simplest case, it would be the observed current state $\mathbf{x}_k$ and the action $\mathbf{u}_k$. In more complicated problems which require state estimation, e.g., POMDPs or control with an unobserved state, the sufficient statistics could denote the state distribution of an optimal filter.

[3]While it can be shown that time-variant stochastic and deterministic policies are equivalent in expectation (Bagnell, Kakade, Ng, & Schneider, 2004), it is clear that for time-invariant policies there exist cases where the optimal policy is stochastic, e.g., if the sufficient statistics $T(\mathbf{x}_{0:k+1}, \mathbf{u}_{0:k})$ are maintained in a suboptimal filter (Bagnell et al., 2004; Jaakkola, Singh, & Jordan, 1995).

An incorrect filter could for example always tell us that a door is open which would result into the robot attempting to drive through it. If the suboptimal filter is then not capable of correcting that state immidiately, a deterministic policy would in this case simply result in a repetition of that scenario.

distribution $r_{k+1} \sim p\left(r_{k+1}|T(\mathbf{x}_{0:k+1}, \mathbf{u}_{0:k}), k+1\right)$; however, it is common to simply write use the average reward $r_{k+1} = r(T(\mathbf{x}_{0:k+1}, \mathbf{u}_{0:k}), k+1) = E_{\tilde{r}}\{\tilde{r}|T(\mathbf{x}_{0:k+1}, \mathbf{u}_{0:k}), k+1\}$ as it simplifies the notation without loss of generality.

Of particular interest for this thesis are systems and policies which arise commonly in motor learning, i.e., The Markovian systems which are either observable (or with an optimal filter available)[4]. Using the Markov assumptions, we have the simplified the sufficient statistics in both the system and policy to

$$T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}) = \mathbf{x}_k,\ T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k}) = (\mathbf{x}_k, \mathbf{u}_k). \tag{3.2}$$

This significantly simplifies the problem; however, it is not in general applicable. Another common assumption is the time-invariance or autonomity of systems, rewards and policy; this desirable property is not always given in practice but can be used in theory if time is represented implicitly through phase.

Definition 1 *A* ***general sequential reinforcement learning setup*** *is given by*

$$\mathbf{x}_{k+1} \sim p\left(\mathbf{x}_{k+1}|T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k}), k\right), \tag{3.3}$$

$$r_{k+1} = r(T(\mathbf{x}_{0:k+1}, \mathbf{u}_{0:k}), k+1), \tag{3.4}$$

$$\mathbf{u}_k \sim \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k) = p(\mathbf{u}_k|T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k, \boldsymbol{\theta}), \tag{3.5}$$

with states $\mathbf{x}_k \in \mathbb{X} \subseteq \mathbb{R}^N$, actions $\mathbf{u}_k \in \mathbb{U} \subseteq \mathbb{R}^M$ for all $k \in \mathbb{N}$, and policy parameters $\boldsymbol{\theta} \in \mathbb{R}^L$.

A ***Markovian system*** *in the sense of Equation (3.2) can be given by*

$$\mathbf{x}_0 \sim p\left(\mathbf{x}_0\right), \tag{3.6}$$

$$\mathbf{x}_{k+1} \sim p\left(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k, k\right), \tag{3.7}$$

$$r_{k+1} = r(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, k+1), \tag{3.8}$$

$$\mathbf{u}_k \sim \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k, k) = p(\mathbf{u}_k|\mathbf{x}_k, k, \boldsymbol{\theta}). \tag{3.9}$$

If the explicit representation of the time-step $k \in \mathbb{N}$ is dropped, these are called ***time-invariant****.*

Throughout this thesis, we will always start with the general system in the sense of Definition 1, and subsequently use the simplifications resulting from Markovian and time-invariance assumptions.

When data is generated from a real system, e.g., from a robot arm in a motor learning task, we can only generate a history $\boldsymbol{\xi}^j_{0:n} = (\mathbf{x}^j_{0:n}, \mathbf{u}^j_{0:n-1})$ of length n with states $\mathbf{x}^j_{0:n}$ and actions $\mathbf{u}^j_{0:n-1}$ where j denotes the number of that history $\boldsymbol{\xi}^j_{0:n}$. Such histories are also known as paths, roll-outs or trajectories. We are in general interested in a combination

[4]Optimal filtering is a difficult problem by itsself and beyond the scope of this thesis. We simply assume that we are either given such a filter or that our filter is sufficiently accurate.

$r(\boldsymbol{\xi}^j_{0:n})$ of all rewards r^j_k obtained throught the history $\boldsymbol{\xi}^j_{0:n}$. The most common types of rewards found in the literature (Neumann & Morlock, 2002) are additive combinations of rewards and most relevant other combinations can be phrased in the same manner.

Definition 2 *An **additive reward** of a history $\boldsymbol{\xi}_{0:n} = (\mathbf{x}_{0:n}, \mathbf{u}_{0:n-1})$ of length n is given by*

$$r(\boldsymbol{\xi}) \equiv g_1 \left(A^{-1} \sum_{k=0}^{n-1} a_k g_2 \left(r \left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, k+1 \right) \right) \right), \tag{3.10}$$

*with a normalization constant $A = \sum_{k=0}^{n-1} a_k$ and $g_1, g_2 : \mathbb{R} \to \mathbb{R}$ being strictly monotonically increasing functions. If g_1, g_2 are identity mappings, we call it the **strictly additive reward** of a history.*

This definition covers most of the definitions in the literature. In reinforcement learning, two cases are common. The *average reward case* defined by strictly additive rewards with $a_k = 1$ is given by $r(\boldsymbol{\xi}) = n^{-1} \sum_{k=0}^{n-1} r(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, k+1)$. The *discounted case*, in contrast, is defined by strictly additive rewards with $a_k = \gamma^k$ and a discount factor $\gamma \in [0, 1)$ can be expressed by

$$r(\boldsymbol{\xi}) = (1-\gamma)^{-1}(1+\gamma^{n+1}) \sum_{k=0}^{n-1} \gamma^k r \left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, k+1 \right) \tag{3.11}$$

where the term $\gamma^{n+1} \ll 1$ for all interesting cases and $\gamma^{n+1} \to 0$ for the infinite horizon case $n \to \infty$. The normalization constant $(1-\gamma)^{-1}(1+\gamma^{n+1})$ is omitted in most of the reinforcement learning literature. Even the multiplicative case $r(\boldsymbol{\xi}) = \prod_{k=0}^{n-1} r(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1},\ k+1)$, often found in the operations research literature (Neumann & Morlock, 2002) can be expressed by choosing $a_k = 1/n$, $g_1(r) = \exp r$ and $g_2(r) = \ln r$. The reinforcement learning literature has focussed primarily on strictly additive rewards and most of the results are coined by that assumption. However, in our preliminary work on probabilistic policy search methods in Section 3.4, we will see that multiplicative rewards can become fairly useful when the rewards are probabilities[5].

3.1.1.2 Objective of Reinforcement Learning

Let us now assume we can generate histories or roll-outs with our motor system and that we have chosen a form of a parameterized motor policy for creating actions. As we have already discussed in our basic definitions in Section 3.1.1.1, we intend to maximize the rewards over the possible histories. This measure is commonly known as the **expected return** (Sutton, McAllester, Singh, & Mansour, 2000b) and is the objective function of classical reinforcement learning. This formulation allows us to define the goal of reinforcement learning.

[5]Note that additive rewards can be transformed into multiplicative ones $\sum_i r_i = \ln \prod_i \exp(r_i)$. The optimization problems $\exp\left(E\left\{\sum_i r_i\right\}\right) \neq E\left\{\prod_i \exp(r_i)\right\}$ have different optimal solutions.

Definition 3 *The **objective of reinforcement learning** is to find the policy $\pi^* = \pi_{\boldsymbol{\theta}^*}$ with parameters $\boldsymbol{\theta}^* \in \mathbb{R}^L$ which is optimal with respect to an evaluation function $J(\boldsymbol{\theta})$ using data from trial and error. The most common evaluation function is the **expected return***

$$J(\boldsymbol{\theta}) \equiv E_{\boldsymbol{\xi}}\{r(\boldsymbol{\xi})\} = \int_{\Xi} p_{\boldsymbol{\theta}}(\boldsymbol{\xi}) r(\boldsymbol{\xi}) d\boldsymbol{\xi}, \tag{3.12}$$

where $p_\theta(\boldsymbol{\xi})$ denotes the probability density of a complete history $\boldsymbol{\xi} \in \Xi$ if the policy parameters are $\boldsymbol{\theta}$.

For any method introduced in this chapter, we will start with Definition 3 and subsequently make use of the appropriate assumptions from Section 3.1.1.1 in order to simplify the problem. We realize that there are three major simplifications, i.e., (1) the strictly additive rewards assumption from Definition 2, (2) the Markov assumption and (3) the time-invariance assumption from Definition 1. The strictly additive rewards assumption allows us to partition the reward into an expectation over a mixture of history pieces given by

$$J_1(\boldsymbol{\theta}) \equiv A^{-1} \sum_{k=0}^{n-1} a_k \int_{\Xi_{0:k+1}} p_{\boldsymbol{\theta}}(\boldsymbol{\xi}_{0:k+1}) r(T(\mathbf{x}_{0:k+1}, \mathbf{u}_{0:k}), k+1) d\boldsymbol{\xi}_{0:k+1}, \tag{3.13}$$

where $\boldsymbol{\xi}_{0:k+1} = (\mathbf{x}_{0:k+1}, \mathbf{u}_{0:k})$ denotes the incomplete path up to time step k. As the trajectory pieces $p_{\boldsymbol{\theta}}(\boldsymbol{\xi}_{0:k+1})$ which are being optimized are shorter, it is usually easier to find a solution in this formulation. When we make additional usage of the Markov assumption, we can simplify this kind of problem from a history-based problem into a state-based one with

$$J_2(\theta) \equiv A^{-1} \sum_{k=0}^{n-1} \int_{\mathbb{X}} a_k d_k^\pi(\mathbf{x}) \int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}, k) \int_{\mathbb{X}} p(\mathbf{x}'|\mathbf{x}, \mathbf{u}, k) r(\mathbf{x}, \mathbf{u}, \mathbf{x}', k) d\mathbf{x}' d\mathbf{u} d\mathbf{x}, \tag{3.14}$$

with $d_{k+1}^\pi(\mathbf{x}_{k+1}) = \int_{\mathbb{X}^k \times \mathbb{U}^k} p(\mathbf{x}_{0:k+1}, \mathbf{u}_{0:k}) \mathbf{d}\mathbf{x}_{0:k} \mathbf{d}\mathbf{u}_{0:k}$ being the distribution of states at time step k when following the policy. Note that the expectation is no longer given over histories but simply over states and actions. At this point, we are leaving path based policy search and are getting towards a state and action based formulation required for value function methods. After making use of the third assumption by additionally dropping the dependence on time, we obtain the standard reinforcement learning scenario

$$J_3(\theta) = \int_{\mathbb{X}} d^\pi(\mathbf{x}) \int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}) r(\mathbf{x}, \mathbf{u}) d\mathbf{u} d\mathbf{x}. \tag{3.15}$$

using the definitions $d^\pi(\mathbf{x}) = A^{-1} \sum_{k=0}^{n-1} a_k d_k^\pi(\mathbf{x})$ and $r(\mathbf{x}, \mathbf{u}) = E_{\mathbf{x}'}\{r(\mathbf{x}, \mathbf{u}, \mathbf{x}')\}$. Note that the state distribution $d^\pi(\mathbf{x})$ is a mixture distribution over the states. While this excercise of simplifying the expected return might appear pointless at this instant, we assure the reader that it will become very useful in Section 3.2.

3.1.1.3 Policy Search or Greedy Value Functions Methods

Already in the early dynamic programming literature which has been one of the inspirations of reinforcement learning, Bellman (1967, 1971) noted that there were two ways how sequential decision problems can be tackled, i.e., the solution could be searched in value function space or in policy space (Bellman, 1967, 1971). Since its emergence as a seperate research field, both solution branches have been present in reinforcement learning and the popularity of both approaches has varied significantly over the last two decades. Early policy search methods such as the policy gradient method REINFORCE (Williams, 1992) as well as the original Actor-Critic (Barto, Sutton, & Anderson, 1983) received significant attention in the late 1980s and beginning 1990s. Due to the success of temporal difference learning, greedy value function methods dominated the field during most of the 1990s (Sutton, 2000) and significantly reduced interest in policy search methods. Only recently, policy search methods have received a revival mostly due to new results in policy gradient methods as well as due to novel simulation methods such as PEGASUS.

In this section, we do not want to make the point that either greedy value function methods or policy search methods are in general superior. However, we do want to show that the combination of the greedy operator with value function methods can be dangerous when applied to high-dimensional motor control problems.

Greedy value function techniques attempt to learn the value function

$$Q_k^\pi(\mathbf{x}, \mathbf{u}) = E\left\{ \sum_{i=k}^{n} a_i r_i \middle| \mathbf{x}_k = \mathbf{x}, \mathbf{u}_k = \mathbf{u} \right\}, \tag{3.16}$$

and then greedily compute the policy which is optimal with respect to this value function by $\pi'(u|x,k) = \delta(u - \operatorname{argmax} Q_k^\pi(\mathbf{x}, \mathbf{u}))$ where $\delta(\cdot)$ is the Dirac impulse or point mass distribution. In most cases, exploration is added by using a softmax distribution instead of the point distribution or a low probability for other actions, e.g., as in Sutton's ε-greedy formulation (Sutton & Barto, 1998b). The major focus in the literature has been on learning the value function $Q_k^\pi(\mathbf{x}, \mathbf{u})$ from data. In this case, we use a function approximator given by $Q_k^\pi(\mathbf{x}, \mathbf{u}) = \boldsymbol{\phi}(x, u, k)^T \boldsymbol{w}$ where $\boldsymbol{w}$ denotes the parameters and $\boldsymbol{\phi}(x, u, k)$ the basis functions of the function approximator. This problem is well-behaved if appropriate basis functions $\phi(x, u, k)$ are known such as in the discrete case where a look-up table suffices or in the linear quadratic case with a quadratic expansion. For such basis functions, the parameters $\boldsymbol{w}$ can be solved using either temporal difference methods, monte-carlo rollouts or, in the presence of analytically tractable system models, one can apply dynamic programming. However, neither look-up table or quadratic expansions generalize to higher dimensional, nonlinear systems and therefore perfect basis functions are rarely available. In most practical cases, approximate basis functions have to be applied, which results in a variety of problems as the optimal value function often cannot be represented accurately even if the intial value functions can (Bartlett, 2002; Tsitsiklis & Roy, 1997) and different reinforcement learning methods converge to different solutions with the more faster methods being biased (Schoknecht, 2003).

Even if we had ideal basis functions, these greedy value function methods are problematic. No matter how many rollouts the learning system generates, it can never fill the high-dimensional state-space required for motor skills. As a greedy policy update cannot guarantee to improve the policy – instead it can get worse proportional to the maximal error of the learned value function at every single update step, often leading to divergence (Bertsekas & Tsitsiklis, 1996)[6]. A simple thought experiment makes the resulting problems clear: assume we have a robot arm with seven degrees of freedom which has a state-space of 14 dimensions with both joint velocities and positions. It is clear that even if we only required p different areas along each joint to be visited, we would need to p^{14} data points to fill the whole space. In discrete dynamic programming this is known as the 'curse of dimensionality' (Bellman, 1957) for the explosion of possible states; for methods which sample data we have the same problem with an explosion of the number of required samples. Thus, the amount of samples required for motor skill learning with value functions can be too large for all practical purposes.

Therefore, value functions methods are problematic in the application to motor skills as they need to learn a value function for the complete attainable state space, i.e., they are global methods. However, let us assume a good initialization of the motor skill which we attempt to learn, for example through supervised imitation learning. This knowledge would then yield an initial policy π_0 which contains a rudimentary behavior. If we additionally make use of attractor policies as outlined in Section 2.3, we can make sure that each policy is defining a tube in the state space in which the exploration tries out variations of the nominal behaviour. Using data from trial and error, we can now improve the policy using the rewards of the histories. These methods are commonly referred to as policy search.

Policy search has numerous advantages in comparison to greedy value function methods. The difference lies in the use of exploration. Value function need to explore all dimensions so that the greedy operator can be applied globally. Policy search methods instead require exploration only as means to determine the direction of improvement close to the current policy. A special need for complicated basis functions is not required and the resulting solutions are not limited to strictly additive rewards. Furthermore, in practice, most successful applications of reinforcement learning to motor skills can be shown to be policy search methods as we will discuss in Section 3.1.2.2. However, the big disadvantage is the global optimality is significantly harder to achieve.

[6]In the special case of localized problems (e.g., discrete grid-worlds), irrelevant or not achievable states can be negelected; however, for motor skills this is not applicable as we would have to limit the robot to not enter these regions of the joint- or task-space.

3.1.2 Motor Skills through Reinforcement Learning

Up to now, we have introduced reinforcement learning with a focus on motor skills, but this relation has remained rather abstract. In this section, we attempt to make this relation more precise through an intoductionary example in Section 3.1.2.1, and a review on previous work on motor learning through reinforcement learning techniques in Section 3.1.2.2.

3.1.2.1 Introductionary Example

In this section, we intend to illustrate through an example that the reinforcement learning framework is an appropriate way to think about motor skills. Assume that we intend to teach a robot the motor skill of hitting a baseball sitting on a stand, i.e., the "T-ball" swing shown, similar as children in the USA or Japan learn their early basic baseball swings. In this case, we have an episodic, discrete task which can be represented by a single motor primitive M_1 and $\mathbf{A}_{\boldsymbol{\theta}}(\mathbf{q},\dot{\mathbf{q}})\,\ddot{\mathbf{q}} = \mathbf{b}_{\boldsymbol{\theta}}(\mathbf{q},\dot{\mathbf{q}})$. Since $\mathbf{A}_{\boldsymbol{\theta}}(\mathbf{q},\dot{\mathbf{q}}) = \tau\mathbf{I}$ for examples in the framework outlined in Section 2.3.2, we can fairly easily turn the learning if such a dynamic system in a reinforcement learning problem. The state of the robot and of the primitive are combined in a state $\mathbf{x}$ and the primitive's output is turned into an action policy $\pi_\theta(\mathbf{u}|\mathbf{x}) = \mathcal{N}(\mathbf{u}|\mu,\boldsymbol{\Sigma})$ with $\mu = \ddot{\mathbf{q}} = \mathbf{b}_{\boldsymbol{\theta}}(\mathbf{q},\dot{\mathbf{q}})$ being the nominal behavior and $\boldsymbol{\Sigma}$ being the exploration of our policy.

Let us assume that we have obtained an example for the T-ball motor skill using supervised learning based on a human presentation. This demonstration would now allow us to learn a initialization of the motor skill, i.e., the initial parameters $\boldsymbol{\theta}$ of the motor primitive while the exploration parameters $\boldsymbol{\Sigma}$ have to be set additionally. We can then create histories $\boldsymbol{\xi}^j_{0:n}$ from repeating the experiment of attempting to hit the ball properly. For each series, the initial state of the robot is brought to a start or initial state $\mathbf{x}_0 \sim p(\mathbf{x}_0)$, subsequently the motor primitive creates the mean behavior $\mathbf{u}_0 \sim \pi_\theta(\mathbf{u}_0|\mathbf{x}_0)$ which in turn results into the transfer to a next state $\mathbf{x}_1 \sim p(\mathbf{x}_1|\mathbf{x}_0,\mathbf{u}_0)$. The last two steps are continued until convergence of the phase state of the primitive at step n. At this instant, the learning system receives a reward $r(\mathbf{x}_{0:n},\mathbf{u}_{0:n})$ – which in the case of T-ball can depend on (a) how properly the ball was hit, (b) how far the ball went and (c) how much power the robot learner consumed. The specification of the reward is left up to the teacher; it is clear that the reward in (a) is available at one instant on the trajectory, the reward (b) is only available upon completion or later while (c) can be fed into the learning system during the path.

3.1.2.2 Previous Work

Throughout the literature, there has been a lot of attention on the application of reinforcement learning to motor learning problems. However, most of these attempts have been limited to low-dimensional or linear systems. The most well-known examples are given here.

The most common motor skills aquired through reinforcement learning are toy skills such as *pole balancing* where a pole has to be balanced on cart, e.g., (Sutton & Barto, 1998b), or robot end-effector (Atkeson & Schaal, 1997; Schaal, 1997). Further examples of toy skills are *pole swing-ups* where the pole has to be brought to the upright position with a limited torque before being balanced[7], and the *ball-on-a-beam* problem (Benbrahim, 1996) where a ball is balanced on a beam (Benbrahim, 1996). In simulation, other toy skills such as *acrobot swing-ups* (Sutton & Barto, 1998b) have been discussed.

More complex skills have been obtained by Gullapalli et al. (1993a, 1994, 1995) who were among the first persons to use reinforcement learning for a real motor skill involving a robot arm. In their case, the robot had to learn how to insert a *peg in a hole* using feedback from the environment. Not so suprisingly, they used policy gradient based policy search methods (Gullapalli, 1993a; Gullapalli et al., 1994; Gullapalli, 1995).

Recently, there has been a lot of work on learning locomotion using reinforcement learning. This work ranges from gait pattern learning , balancing to robot stand-ups. Examples for learning legged locomotion with reinforcement learning can be found in Morimoto and colleagues (Jun Morimoto, 2000; Morimoto, 2002), Benbrahim & Franklin (Benbrahim & Franklin, 1997; Benbrahim, 1996), Zhang & Meng (C. Zhou, 2000) and Salatin et al. (A.W. Salatian, 1997).

In summary, most previous work on aqcuiring motor skills with robotic systems was developed either in a rather task-specific way, i.e., it is not clear how a presented learning or representation framework would generalize to another motor skill. It is this lack to a general learning approach to motor skills which motivates this thesis proposal.

3.2 Policy Gradient Techniques

Policy search methods are the more appropriate kind of reinforcement learning for motor skill aquisition and refinement since they scale better than global value function based methods as we have discussed in Section 3.1. Among the most straightforward forms of policy search for parameterized policies are policy gradient methods. These follow the gradient of the expected return $\nabla_\theta J(\theta)$ with respect to the policy parameters θ, i.e.,

$$\theta_{t+1} = \theta_t + \alpha_t \, \nabla_\theta J(\theta)|_{\theta=\theta_t} + \alpha_t \boldsymbol{\varepsilon}_t, \tag{3.17}$$

where $\alpha_k \in \mathbb{R}^+$ is a time dependent sequence of learnig rates, and $\boldsymbol{\varepsilon}_t \in \mathbb{R}^L$ denotes the error in the gradient estimate. Gradient methods are guaranteed to converge to at least

[7]Note that this corresponds the more complicated version of the *mountain-car problem* studied in the reinforcement learning literature.

a local maximum (Duflo, 1997) if the following conditions hold, i.e., (i) the learning rates fulfill $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$, (ii) the gradient estimate is crescent[8], i.e.,

$$\nabla_\theta J(\theta)|_{\theta=\theta_t}^T \left(\nabla_\theta J(\theta)|_{\theta=\theta_t} + E\{\boldsymbol{\varepsilon}_t\}\right) > \mathbf{0}, \tag{3.18}$$

(iii) the errors in the estimator are random but non-zero[9], i.e., $E\{\boldsymbol{\varepsilon}_t^T \boldsymbol{\varepsilon}_t\} > \mathbf{0}$ and bounded $\boldsymbol{\varepsilon}_t^T \boldsymbol{\varepsilon}_t < \infty$.

Model-based gradient methods been applied to the optimization of control systems since at least the 1960s (Hasdorff, 1976; Jacobson & Mayne, 1970; Dyer & McReynolds, 1970) and have remained a useful approach in current research (Morimoto & Atkeson, 2003; Li & Todorov, 2004). However, these traditional techniques require a precise model of the system and reward function (including the derivatives with respect to the parameters of both state-transition model and reward functions), and only few of these are applicable in the presence of uncertainty. In most cases, the evaluation of the gradient for a given system, given reward, and deterministically executed actions still has to be achieved numerically as an analytical solution does not exist in most cases. The numerics of the gradient calculation are not even always stable (Hasdorff, 1976; Jacobson & Mayne, 1970; Dyer & McReynolds, 1970). Thus, these methods are not generally applicable when models of the environment are not completely known (which includes hidden state situations), and also when the environment has a stochastic component, as typically the case when robots have contact with the world (e.g., as in baseball, locomotion, or object grasping).In our own work we are interested in pursuing alternative ways of obtaining the gradient by perturbing the system. Instead of building and using approximate models of the system to obtain the gradient, we can approximate the gradient from histories of the real system. Such methods are known from a variety of fields including sensitivity analysis (Fuerverger, McLeish, Kreimer, & Rubinstein, 1989), simulation optimization (Glynn, 1987, 1990; Sutton et al., 2000b), operations research (Marbach & Tsitsiklis, 1999, 2003; Konda & Tsitsiklis, 2000) and also from reinforcement learning (Williams, 1986, 1992; Gullapalli, 1993b; Baxter & Bartlett, 1999).

In this section, we will discuss how policy gradients can be estimated efficiently from data without knowledge of a model. First, we will derive a general form for policy gradient estimation, and subsequently, we will show how this estimator can be improved in terms of reducing the variance of the gradient estimate.

[8]This condition is formalization of the statement that the expected angle between the true gradient $\nabla_\theta J(\theta)|_{\theta=\theta_t}$ and its estimate $\nabla_\theta J(\theta)|_{\theta=\theta_t} + \boldsymbol{\varepsilon}_t$ is less than 90°. Unlike unbiasedness, i.e., $E\{\boldsymbol{\varepsilon}_t\} = \mathbf{0}$, this condition can be fulfilled even for finite samples.

[9]The requirement for random errors $\boldsymbol{\varepsilon}_t$ in the gradient estimate is only a formality so that the gradient estimator cannot get stuck near non-isolated saddle points which is unlikely in practice.

3.2.1 Policy Gradient Estimation

The main problem in policy gradient methods is to estimate the gradient from data without biasing the gradient direction such that it is no longer crescent. It is well-known in the simulation community that there are at least two types of gradient estimators for the optimization of stochastic systems (Glynn, 1989), i.e., *finite-difference methods* and *likelihood ratio methods*. Finite-difference methods perturb the parameters by a small amounts $\boldsymbol{\delta\theta}_i$ and then solve the regression problem

$$\mathbf{g}^T \boldsymbol{\delta\theta}^i = \tilde{J}(\boldsymbol{\theta} + \boldsymbol{\delta\theta}^i) - \tilde{J}(\boldsymbol{\theta}), \tag{3.19}$$

where the approximations $\tilde{J}(\cdot)$ of the expected return $J(\cdot)$ are obtained by averaging over histories. Despite being the simplest approach and being very efficient for deterministic problems (Spall, 2003), finite-difference estimators are well-known to be very inefficient in stochastic settings (Glynn, 1989). The reason for this is that the variance in the expected return estimates is often larger than the change caused by parameter variation which complicates the credit assignment problem for the respective parameters[10]. However, as they require the least knowledge on the system, they are still used in reinforcement learning (Fidelman & Stone, 2004; Kohl & Stone, 2004) and can be relatively successful[11].

The other branch of gradient estimators, likelihood ratio methods, are theoretically more involving. Most of the gradient methods applied to multistage optimization rely upon the "likelyhood ratio trick", i.e., when differentiating an expectation $E_{\boldsymbol{\xi}}\{r(\boldsymbol{\xi})\}$ with respect to a variable θ, we have

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\Xi} \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{\xi}) r(\boldsymbol{\xi}) d\,\boldsymbol{\xi} = \int_{\Xi} p_{\boldsymbol{\theta}}(\boldsymbol{\xi}) \nabla_{\theta} \log p_{\boldsymbol{\theta}}(\boldsymbol{\xi}) r(\boldsymbol{\xi}) d\boldsymbol{\xi} = E_{\boldsymbol{\xi}}\{\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\xi}) r(\boldsymbol{\xi})\}, \tag{3.20}$$

assuming $r(\boldsymbol{\xi})$ does not depend on the variable $\boldsymbol{\theta}$. Most policy gradient estimators in the reinforcement learning are based upon Equation (3.20) and which is also true for the ones on which we will focus in this section. To our knowledge, this is the most general formulation of the policy gradient estimation problem.

3.2.1.1 General Likelyhood Ratio Gradient Estimation

Before entering the topic of policy gradients for sequential control problems, we first need to make clear how Equation (3.20) allows us to estimate the gradient of the expected

[10]In fact, a single rollout can do significantly more damage as there is no credit assignment to the taken actions. However, note that this problem is no longer present if deterministic sampling techniques such as PEGASUS are employed.

[11]However a further problem exists as the exploration is limited to admissible policy variations $\theta + \delta\theta_i$ which have to be carefully checked. Otherwise, for instance, the control system may become unstable.

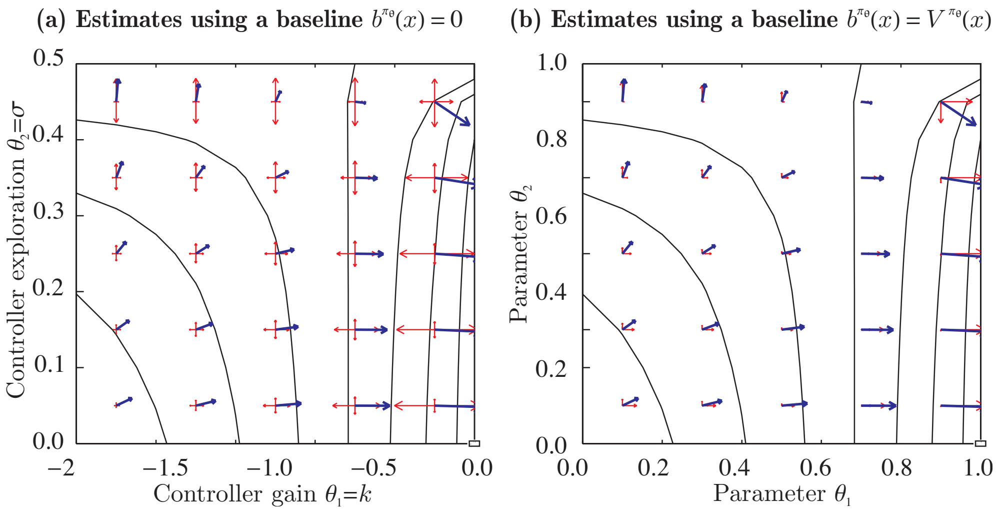


Figure 3.1: This figure shows the effects of baselines on the policy gradient estimates for a simple toy problem with two states and actions. With bad baselines, the components point into all directions and the gradient estimator has to average our the error in all directions.

return. The straightforward estimator of the gradient can be obtained by replacing the expectation $E_{\boldsymbol{\xi}}\{f(\boldsymbol{\xi})\}$ by the sample average $\langle f(\boldsymbol{\xi}^j)\rangle_{j=1:m}$ in Equation (3.20) yielding

$$\boldsymbol{\Delta} = \left\langle \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j) r(\boldsymbol{\xi}^j) \right\rangle_{j=1:m} = \frac{1}{m}\sum_{j=1}^{m} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j) r(\boldsymbol{\xi}^j), \tag{3.21}$$

with the history parted in $\boldsymbol{\xi}^j = [\xi_1^j, \xi_2^j]$ where ξ_2^j denotes all variables depending on $\boldsymbol{\theta}$ while ξ_1^j not directly dependent on ξ_1^j. Averaging over samples will result into an unbiased gradient estimate. However, the variance in the gradient estimate can be significant. The cause of this variance is that the magnitude of gradient components from different histories can often be significantly larger than the actual gradient while pointing in stringly different directions as indicated in Figure 3.1. Therefore, the direction of the gradient estimate depends completely on the accuracy of the rewards and the sampling of paths which act as weighting factors for the different log-derivatives. In many cases, the components $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j) r(\boldsymbol{\xi}^j)$ are much larger than the actual gradient $\nabla_{\boldsymbol{\theta}} J$ and point in all directions as in Figure 3.1. Ideally, we want to extract this large part out of the gradient components so that it no longer needs to be averaged out. There is probably only one way to achieve this, i.e., through a *baseline*. To our knowledge, the first to notice this detail were Dayan (Dayan, 1990) in a slightly different context and

Williams (1986, 1992) in the context of policy gradients. We can make the reasoning behind baselines concrete in Lemma 4.

Lemma 4 *If a random variable* $\boldsymbol{\xi}$ *can be partitioned into* $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2)$ *with* $p(\boldsymbol{\xi}) = p(\boldsymbol{\xi}_1)p_{\boldsymbol{\theta}}(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)$ *then a baseline* $\mathbf{b}(\boldsymbol{\xi}_1)$ *will not bias the gradient estimate and the likelihood ratio gradient is given by*

$$\nabla_{\theta_i} J(\boldsymbol{\theta}) = E_{\boldsymbol{\xi}}\{\nabla_{\theta_i} \log p_{\boldsymbol{\theta}}(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)\,(r(\boldsymbol{\xi}) - b_i(\boldsymbol{\xi}_1))\}. \tag{3.22}$$

If $\boldsymbol{\xi}_1 = \emptyset$ *is empty, then* $\mathbf{b}(\boldsymbol{\xi}_1) = \mathbf{b} \in \mathbb{R}^L$ *can only be a constant vector.*

Proof. We can show that the baseline averages out in the expectation. As $\int_{\Xi_2} p_{\boldsymbol{\theta}}(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)d\boldsymbol{\xi}_2 = 1$ with any path partition we also have

$$\int_{\Xi_2} p_{\boldsymbol{\theta}}(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)\nabla_{\theta_i} \log p_{\boldsymbol{\theta}}(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) b_i(\boldsymbol{\xi}_1) d\boldsymbol{\xi}_2 = b_i(\boldsymbol{\xi}_1) \int_{\Xi_2} \nabla_{\theta_i} p_{\boldsymbol{\theta}}(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) d\boldsymbol{\xi}_2 = 0,$$

and therefore the gradient estimate is not biased by the components. ■

Note that Lemma 4 has a more general baseline than common in the literature - the extensive literature on baselines to date (Berny, 2000; Weaver & Tao, 2001a, 2001b; Greensmith, Bartlett, & Baxter, 2004; Williams, 1992; Greensmith, Bartlett, & Baxter, 2001; Lawrence, Cowan, & Russell, 2003) appears to have overlooked the fact that a seperate baseline can be selected for each policy parameter.

Baselines can have a significant effect on the convergence speed of the gradient estimator. This is illustrated in Figure 3.1 where a zero baseline results into gradients in all directions while a smartly chosen one secures that the gradient estimate always correlates with the true gradient.

3.2.1.2 Policy Gradient Estimation for Sequential Decision Problems

We can make use of the fact that the probability density of trajectories can be given by

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{\xi}_{0:n}\right) = \prod_{k=0}^{n-1} p\left(\mathbf{x}_{k+1}|T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k}), k\right) \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k), \tag{3.23}$$

for any general sequential decision task. It is straightforward from Equation (3.23) that the log-likelihood derivative of the history can be given by

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}\left(\boldsymbol{\xi}_{0:n}\right) = \sum_{k=0}^{n-1} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k). \tag{3.24}$$

This derivative does not include any notion of the system model and therefore the likelihood ratio gradient can be written by inserting Equation (3.20) into Equation (3.24) which yields

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = E_{\boldsymbol{\xi}} \left\{ \left(\sum_{k=0}^{n-1} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k | T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k) \right) (r(\boldsymbol{\xi}) - b(\mathbf{x}_0)) \right\}, \tag{3.25}$$

without any further assumptions. In here, we can only partition $\boldsymbol{\xi}_2 = [\mathbf{x}_{1:n}, \mathbf{u}_{0:n-1}]$, $\boldsymbol{\xi}_1 = \mathbf{x}_0$, thus the baseline is given by $b(\mathbf{x}_0)$. From Equation (3.25) we can proceed in several ways. Equation (3.25) can be expanded to have more complicated baselines into

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = E_{\boldsymbol{\xi}} \left\{ \sum_{k=0}^{n-1} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k | T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k) \, (r(\boldsymbol{\xi}) - b(T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k)) \right\}, \tag{3.26}$$

which allows a significantly more complicated baseline $b(T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k)$ as we can partition into $\boldsymbol{\xi}_2 = [\mathbf{u}_k]$, $\boldsymbol{\xi}_1 = [\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}]$, thus the baseline is given by $b(\mathbf{x}_0)$. Does Equation (3.26) gain anything over Equation (3.25)? Not in general – but in the presence of parametric forms of the baseline this can yield a significant advantage.

Both Equation (3.25) and (3.26) are equivalent to the well-known estimator *Episodic REINFORCE* (Williams, 1992) if we assuming that the reward $r(\boldsymbol{\xi})$ is just a final reward, i.e., $r(\boldsymbol{\xi}) = r(\mathbf{x}_n)$. Note that Williams (1992) unnecessarily assumed that the policy and system had to be Markovian and that he only allowed final rewards. There is no explicit need for the system to be Markovian or the reward to be additive in order to obtain a policy gradient.

3.2.1.3 Policy Gradients in Markovian Strictly Additive Problems

By assuming that our problem is Markovian and that our rewards are strictly additive, we take both Equation (3.25) and (3.26) one step further. First, we realize that Equation (3.25) can be expanded if the assumption that the problem was strictly additive holds, and if we furthemore make use of the Markov assumption, we can simplify the gradient to

$$\nabla_\theta J(\theta) = A^{-1} E_{\boldsymbol{\xi}} \left\{ \sum_{k=0}^{n-1} a_k \left(\sum_{h=0}^{k} \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_h | T(\mathbf{x}_{0:h}, \mathbf{u}_{0:h-1}), h) \right) (r(T(\mathbf{x}_{0:k+1}, \mathbf{u}_{0:k})) - b(\mathbf{x}_0, k)) \right\}, \tag{3.27}$$

$$= A^{-1} E_{\boldsymbol{\xi}} \left\{ \sum_{k=0}^{n-1} a_k \left(\sum_{h=0}^{k} \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_h | \mathbf{x}_h, h) \right) (r(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, k) - b(k)) \right\}. \tag{3.28}$$

Note that $E_{\boldsymbol{\xi}}\left\{\nabla_\theta \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_h|\mathbf{x}_h, h) r(\mathbf{x}_k, \mathbf{u}_k))\right\} = \mathbf{0}$ for $k < h$ as a past reward cannot depend on future actions. We realize that $\sum_{h=0}^{k} \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_h|\mathbf{x}_h, h)$ can be estimated recursively in Markovian problems by

$$d_{k+1}^{\pi}(\mathbf{x}) \nabla_\theta \log d_{k+1}^{\pi}(\mathbf{x}) = \int_{\mathbb{X}^t \times \mathbb{U}^t} d_k^{\pi}(\mathbf{x}) \pi_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x}, k) \left(\nabla_\theta \log d_k^{\pi}(\mathbf{x}) + \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x}, k)\right) d\mathbf{x} d\mathbf{u}, \tag{3.29}$$

as $d_{k+1}^{\pi}(\mathbf{x}) = \int_{\mathbb{X}^t \times \mathbb{U}^t} d_k^{\pi}(\mathbf{x}) \pi_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x}, k) d\mathbf{x} d\mathbf{u}$. This insight allows another simplification, particularly when also assuming time-invariance, start-state and next-state independance, i.e.,

$$\nabla_\theta J(\theta) = A^{-1} E_{\boldsymbol{\xi}} \left\{ \sum_{k=0}^{n-1} a_k \left(\nabla_\theta \log d_k^{\pi}(\mathbf{x}_k) + \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k, h)\right) \left(r(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, k) - b(k)\right) \right\}, \tag{3.30}$$

$$= E_{\mathbf{x},\mathbf{u}} \left\{ \left(\nabla_\theta \log d^{\pi}(\mathbf{x}) + \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x})\right) \left(r(\mathbf{x}, \mathbf{u})) - b\right) \right\}. \tag{3.31}$$

This type of algorithms is considered the REINFORCE-type and methods such as GPOMDP employ this gradient.

The focussed reader has probably noticed that the derivatives $\nabla_\theta \log d^{\pi}(\mathbf{x})$ are non-trivial to estimate for large horizons as they often become derivatives through stationary distributions. For obtaining such derivatives, Equation (3.29) becomes an integral equation (Tricomi, 1985). This lead Sutton et al. (2000a) and Konda et al. (2002, 2000, 2001) to instead marginalize apply the simplifications to Equation (3.26). If furthermore, we have one of the standard cases of reinforcement learning, we also have $a_{k+h} = a_k a_h$ and can obtain

$$\nabla_\theta J(\theta) = E_{\boldsymbol{\xi}} \left\{ \sum_{k=0}^{n-1} a_k \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k, k) \left(\sum_{h=k}^{n-1} a_{h-k} r(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, k) - b(\mathbf{x}_k, k) \right) \right\}, \tag{3.32}$$

$$= E_{\boldsymbol{\xi}} \left\{ \sum_{k=0}^{n-1} a_k \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k, k) \left(Q_k^{\pi}(\mathbf{x}, \mathbf{u}) - b(\mathbf{x}_k, k)\right) \right\}, \tag{3.33}$$

by defining $Q_k^{\pi}(\mathbf{x}, \mathbf{u}) = E\left\{ \sum_{h=k}^{n-1} a_{h-k} r(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, k) \middle| \mathbf{x}_k = \mathbf{x}, \mathbf{u}_k = \mathbf{u} \right\}$ as the value function. If we additionally assume time-invariance, we obtain

$$\nabla_\theta J(\theta) = E_{\mathbf{x},\mathbf{u}} \left\{ \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x}) \left(Q^{\pi}(\mathbf{x}, \mathbf{u}) - b(\mathbf{x})\right) \right\}, \tag{3.34}$$

the main result by Konda et al. (2000) and Sutton et al. (2000). Note that this gradient – while the prettiest in theory, is actually the most complicated as obtaining the function $Q^{\pi}(\mathbf{x}, \mathbf{u})$ usually requires the solution of Bellman integral equations. The notion that these were easier to solve than solving the integral equation of the stationary distribution derivatives is incorrect as these are the adjungated homogenious form of the later. This means that both are equally hard to solve and in most cases the adjungated homogeneous has to be solved in order to solve the later (Sabelfels, 1987).

3.2.2 Variance Reduction Techniques for Policy Gradient Methods

In general, we would want a likelihood ratio policy gradient estimators $\mathbf{\Delta}$ to have two features, (i) we require it to be unbiased, i.e., it always converges to the true gradient in expectation

$$\nabla_\theta J(\theta) = E_{\boldsymbol{\xi}}\{\mathbf{\Delta}\}, \tag{3.35}$$

while at the same time, (ii) we need the covariance of the gradient estimate

$$\mathbf{\Sigma} = E_{\boldsymbol{\xi}}\{(\mathbf{\Delta} - \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}))(\mathbf{\Delta} - \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}))^T\} \to \min \tag{3.36}$$

to be minimal. In many cases, we can settle for minimum variance given by

$$\sigma^2 = \operatorname{Tr} \mathbf{\Sigma} = E_{\boldsymbol{\xi}}\{((\mathbf{\Delta} - \nabla_\theta J_1(\theta))^T (\mathbf{\Delta} - \nabla_\theta J_1(\theta))\} \to \min,$$

which can be achieved easier.

In this section, we will deal with two kind of variance reduction techniques. Firstly, we are going to discuss how to minimize the variance by choosing a minimum variance baseline. We derive this result in general, such that it will be applicable to any of the previosuly discussed gradient estimators. However, while baselines can be used to reduce the variance of the estimate resulting from the sampling of the different histories, they often cannot handle the credit assignment problem and reduce the variance by adjusting how much credit is given to past actions.

3.2.2.1 Unbiased Variance Reduction through Optimal Baselines

In this section, we will attempt to find minimum variance baselines. We will first start with minimum variance scalar baselines and then extend the analysis to parameterized baselines. Nevertheless, we realize that by minimizing

$$\sigma^2(b) \propto E_{\boldsymbol{\xi}}\{\mathbf{\Delta}^T(b)\mathbf{\Delta}(b)\} \tag{3.37}$$

with respect to a scalar baseline, we only minimize the length of the gradient estimate $\mathbf{\Delta}(b)$ and not the variance in the direction of the gradient estimate. As not the length of the gradient determines the performance of gradient methods but the direction, we need to find the gradient which minimizes the variance in direction. It makes clear that not the variance of the gradient estimator but its covariance matrix

$$\mathbf{\Sigma} = E_{\boldsymbol{\xi}}\{((\mathbf{\Delta} - \nabla_\theta J(\theta))(\mathbf{\Delta} - \nabla_\theta J(\theta))^T\} \tag{3.38}$$

has to be minimized – an optimization problem which cannot be solved in general as this problem does not have a single unique solution. The reason for this nonexistence is that not all objectives functions, i.e., the $0.5\left(L^2 - L\right)$ covariance terms, can be minimized through setting L baseline components at once.

Scalar Minimum Variance Baselines. In this section, we focus on scalar baselines as in the literature (Berny, 2000; Weaver & Tao, 2001a, 2001b; Greensmith et al., 2004, 2001; Lawrence et al., 2003) and we realize that we can make a very strong statement on these. We can in fact derive the unbiased likelyhood ratio gradient estimator with the minimum variance baseline given in Theorem .

Theorem 5 *The unbiased gradient estimator with the minimum variance baseline is given by*

$$\boldsymbol{\Delta} = \left\langle \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j)\left(r(\boldsymbol{\xi}^j) - b(\boldsymbol{\xi}_1^j)\right)\right\rangle_{j=1:m}, \tag{3.39}$$

with the baseline

$$b_i(\boldsymbol{\xi}_1) = \frac{\left\langle \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j)^T \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j) r(\boldsymbol{\xi}^j)\right| \boldsymbol{\xi}_1^j = \boldsymbol{\xi}_1 \Big\rangle_{j=1:m}}{\left\langle \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j)^T \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j)\right| \boldsymbol{\xi}_1^j = \boldsymbol{\xi}_1 \Big\rangle_{j=1:m}}, \tag{3.40}$$

where $\left\langle f(\boldsymbol{\xi}^j) \middle| \boldsymbol{\xi}_1^j = \boldsymbol{\xi}_1 \right\rangle_{j=1:m} = m^{-1}\sum_{j=1}^m f(\boldsymbol{\xi}^j)\delta(\boldsymbol{\xi}_1^j = \boldsymbol{\xi}_1)$ *with* $\sum_{j=1}^m \delta(\boldsymbol{\xi}_1^j = \boldsymbol{\xi}_1) > 1$ *denotes the sample average conditioned on* $\boldsymbol{\xi}_1$*. If* $\boldsymbol{\xi}_1 = \emptyset$ *is empty, then* $b(\boldsymbol{\xi}_1) = b$ *is a constant.*

Proof. The average over histories is unbiased in the sense of Equation (3.35) as

$$\boldsymbol{\Delta} = \frac{1}{m}\sum_{j=1}^{m} E_{\boldsymbol{\xi}}\left\{\nabla_\theta \log p_\theta(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j)\left(r(\boldsymbol{\xi}^j) - b(\boldsymbol{\xi}_1^j)\right)\right\} = \nabla_\theta J(\theta).$$

We can rewrite the variance of Equation (3.37) using the unbiasedness as

$$\begin{aligned}\sigma^2(b) &= E_{\boldsymbol{\xi}}\{\boldsymbol{\Delta}^T(b)\boldsymbol{\Delta}(b)\} - 2\nabla_\theta J(\theta)^T E_{\boldsymbol{\xi}}\{\boldsymbol{\Delta}(b)\} + \|\nabla_\theta J(\theta))\|_2^2, \\ &= E_{\boldsymbol{\xi}}\{\boldsymbol{\Delta}^T(b)\boldsymbol{\Delta}(b)\} - \|\nabla_\theta J(\theta))\|_2^2, \end{aligned} \tag{3.41}$$

where $\|\nabla_\theta J(\theta))\|_2^2$ does not depend on the baseline and can therefore be dropped. We intend to find the baseline function $b(\cdot)$ which minimizes $\boldsymbol{\Delta}^T(b)\boldsymbol{\Delta}(b)$. We realize that the partition of our random variable simplifies that choice as for minimizing $E_{\boldsymbol{\xi}}\left\{f(\boldsymbol{\xi}, b(\boldsymbol{\xi}_1))\right\}$ for any function $f(\boldsymbol{\xi}, b(\boldsymbol{\xi}_1))$ a lower bound is given by

$$\begin{aligned}
\min_{b(\cdot)} E_{\boldsymbol{\xi}}\left\{f(\boldsymbol{\xi}_2, b(\boldsymbol{\xi}_1))\right\} &= \int_{\Xi_1} p(\boldsymbol{\xi}_1) \int_{\Xi_2} p(\boldsymbol{\xi}_2) f(\boldsymbol{\xi}, b(\boldsymbol{\xi}_1)) d\boldsymbol{\xi}_2 d\boldsymbol{\xi}_1, \\
&\geq \int_{\Xi_1} p(\boldsymbol{\xi}_1) \min_{\tilde{b}=b(\boldsymbol{\xi}_1)} \int_{\Xi_2} p(\boldsymbol{\xi}_2) f(\boldsymbol{\xi}, \tilde{b}) d\boldsymbol{\xi}_2 d\boldsymbol{\xi}_1, \\
&= E_{\boldsymbol{\xi}_1}\left\{\min\nolimits_{\tilde{b}=b(\boldsymbol{\xi}_1)} E_{\boldsymbol{\xi}_2}\left\{ f(\boldsymbol{\xi}, \tilde{b}) \middle| \boldsymbol{\xi}_1\right\}\right\}.
\end{aligned}$$

Therefore we can choose the optimal baseline by minimizing just $\boldsymbol{\Delta}^T(\tilde{b}, \boldsymbol{\xi}_1)\boldsymbol{\Delta}(\tilde{b}, \boldsymbol{\xi}_1)$ instead of $\boldsymbol{\Delta}^T(b)\boldsymbol{\Delta}(b)$. This solution to

$$\boldsymbol{\Delta}^T(\tilde{b}, \boldsymbol{\xi}_1)\boldsymbol{\Delta}(\tilde{b}, \boldsymbol{\xi}_1) = E_{\boldsymbol{\xi}_2}\left\{ \left(r(\boldsymbol{\xi}) - b(\boldsymbol{\xi}_1)\right) \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)^T \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) \left(r(\boldsymbol{\xi}) - b(\boldsymbol{\xi}_1)\right) \middle| \boldsymbol{\xi}_1\right\} \to \min$$

with respect to $b(\boldsymbol{\xi}_1)$ can be given by

$$b(\boldsymbol{\xi}_1) = \frac{E_{\boldsymbol{\xi}_2}\left\{ \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)^T \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) r(\boldsymbol{\xi}) \middle| \boldsymbol{\xi}_1\right\}}{E_{\boldsymbol{\xi}_2}\left\{ \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)^T \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) \middle| \boldsymbol{\xi}_1\right\}}.$$

We can replace the expectations by the sample averages and obtain Equation (3.48). ■

Various forms of Theorem 5 have occured in (Berny, 2000; Weaver & Tao, 2001a, 2001b; Greensmith et al., 2004, 2001; Lawrence et al., 2003) for applications to special cases.

Minimum Variance Parametric Baselines. However, as we are usually interested into continuous or very large discrete problems where a random variable $\boldsymbol{\xi}_1$ rarely reoccurs, we see that the condition $\sum_{j=1}^m \delta(\boldsymbol{\xi}_1^j = \boldsymbol{\xi}_1) > 1$ is fairly academic. Unless $\boldsymbol{\xi}_1 = \emptyset$ is empty with a constant baseline, this can rarely occur in an implementation of the algorithm. Instead if we assume that the baseline generalizes over a larger region than just the point $\boldsymbol{\xi}_1^j = \boldsymbol{\xi}_1$, we can rewrite the baseline as

$$b(\boldsymbol{\xi}_1) = \sum_{h=1}^{K} b_h p(h|\boldsymbol{\xi}_1), \tag{3.42}$$

where $b_h \in \mathbb{R}$ is the baseline for this region and $p(h|\boldsymbol{\xi}_1)$ is the conditional probability that region h contains variable $\boldsymbol{\xi}_1$. We can write this baseline a parametric form of

$$b(\boldsymbol{\xi}_1) = \boldsymbol{\psi}(\boldsymbol{\xi}_1)^T \mathbf{b}_p, \tag{3.43}$$

with basis functions $\boldsymbol{\psi}(\boldsymbol{\xi}_1) = [p(1|\boldsymbol{\xi}_1), p(2|\boldsymbol{\xi}_1), \ldots, p(K|\boldsymbol{\xi}_1)]^T$, and baseline parameters $\mathbf{b}_p = [b_1, b_2, \ldots, b_K] \in \mathbb{R}^K$. Finding the optimal baseline now reduces to finding the optimal baseline parameters $\mathbf{b}_p$.

Theorem 6 *If the baseline is given in a parametric form $b(\boldsymbol{\xi}_1) = \boldsymbol{\psi}(\boldsymbol{\xi}_1)^T \mathbf{b}_p$, we can estimate the minimum variance baseline parameters by*

$$\mathbf{b}_p = \tilde{\mathbf{U}}^{-1} \tilde{\boldsymbol{v}}, \tag{3.44}$$

with

$$\tilde{\mathbf{U}} = \left\langle \boldsymbol{\psi}(\boldsymbol{\xi}_1) \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)^T \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) \boldsymbol{\psi}(\boldsymbol{\xi}_1)^T \right\rangle, \tag{3.45}$$

$$\tilde{\boldsymbol{v}} = \left\langle \boldsymbol{\psi}(\boldsymbol{\xi}_1) \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)^T \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) r(\boldsymbol{\xi}) \right\rangle. \tag{3.46}$$

Proof. We take up Equation (3.41) and rewrite the problem as

$$\sigma^2(\mathbf{b}_p) \propto E_{\boldsymbol{\xi}}\{\boldsymbol{\Delta}^T(\mathbf{b}_p)\boldsymbol{\Delta}(\mathbf{b}_p)\} = \mathbf{b}_p^T \mathbf{U} \mathbf{b}_p - 2\boldsymbol{v}^T \mathbf{b}_p + \boldsymbol{w},$$

with

$$\begin{aligned}
\mathbf{U} &= E_{\boldsymbol{\xi}}\{\boldsymbol{\psi}(\boldsymbol{\xi}_1) \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)^T \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) \boldsymbol{\psi}(\boldsymbol{\xi}_1)^T\}, \\
\boldsymbol{v} &= E_{\boldsymbol{\xi}}\{\boldsymbol{\psi}(\boldsymbol{\xi}_1) \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)^T \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) r(\boldsymbol{\xi})\}, \\
\boldsymbol{w} &= E_{\boldsymbol{\xi}}\{\nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)^T \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) r^2(\boldsymbol{\xi})\}.
\end{aligned}$$

The optimal solution is given by $\mathbf{b} = \mathbf{U}^{-1}\boldsymbol{v}$. When replacing the expectations by sample averages, we obtain $\tilde{\mathbf{U}}$, $\tilde{\boldsymbol{v}}$, and Equation (3.44). ■

The optimal baseline for $K = 1$ is obviously equivalent to the constant baseline for the case where $\boldsymbol{\xi}_1 = \emptyset$ is empty. The usage of parametric baselines has to our knowledge been suggested both by Berny (2000) and Lawrence et al. (2003). Theorem is fairly similar to the one in (Lawrence et al., 2003). Lawrence et al. (2003) also showed that parametric baselines can yield a dramatic improvement over constant baselines.

Multidimensional Baselines. As noted before in Section 3.2.1.1, it has to our knowledge not yet been noted in the literature that the baseline is not confined to a scalar but can be a vector. The extension multidimensional baselines allows us to estimate the baseline for each gradient direction seperately and therefore minimize the variance in the gradients direction instead of its length. The baselines are suprisingly simple to find as we show in Theorem 7.

Theorem 7 *The unbiased policy gradient estimator $\boldsymbol{\Delta} = [\Delta_1, \Delta_2, \ldots, \Delta_L] \in \mathbb{R}^L$ with a baseline $\mathbf{b}(\boldsymbol{\xi}_1^j) = [b_1, b_2, \ldots, b_L] \in \mathbb{R}^L$ is given by*

$$\boldsymbol{\Delta}_i = \left\langle \nabla_{\theta_i} \log p_{\theta_i}(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j) \left(r(\boldsymbol{\xi}^j) - b_i(\boldsymbol{\xi}_1^j) \right) \right\rangle_{j=1:m}, \tag{3.47}$$

for $i = 1, 2, \ldots, L$. The baseline which minimizes each element of the trace of the covariance matrix (i.e., the variance) is given by

$$b_i(\boldsymbol{\xi}_1) = \frac{\left\langle \nabla_{\theta_i} \log p_\theta(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j) \nabla_{\theta_i} \log p_\theta(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j) r(\boldsymbol{\xi}^j) \middle| \, \boldsymbol{\xi}_1^j = \boldsymbol{\xi}_1 \right\rangle_{j=1:m}}{\left\langle \nabla_{\theta_i} \log p_\theta(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j) \nabla_{\theta_i} \log p_\theta(\boldsymbol{\xi}_2^j|\boldsymbol{\xi}_1^j) \middle| \, \boldsymbol{\xi}_1^j = \boldsymbol{\xi}_1 \right\rangle_{j=1:m}}, \tag{3.48}$$

while the baseline which minimizes the unweighted sum of all covariance matrix elements is given by

$$\mathbf{b} = \mathbf{F}^{-1}\mathbf{H}\mathbf{1}, \tag{3.49}$$

where $\mathbf{F}$ denoted the Fisher information matrix, $\mathbf{H}$ the Hessian.

If $\boldsymbol{\xi}_1 = \emptyset$ is empty, then $\mathbf{b}(\boldsymbol{\xi}_1) = \mathbf{b}$ is a constant vector.

Proof. As in Theorem 7, by requiring unbiasedness, we can simplify the gradient estimator into

$$\boldsymbol{\Sigma} = E_{\boldsymbol{\xi}}\{\boldsymbol{\Delta}\boldsymbol{\Delta}^T\} - \|\nabla_\theta J(\theta)\|^2 \propto E_{\boldsymbol{\xi}}\{\boldsymbol{\Delta}\boldsymbol{\Delta}^T\}.$$

As we can rewrite the components of the covariance matrix in Equation (3.38) into

$$\Sigma_{ih} \propto F_{ih} b_h b_i - H_{ih}(b_h + b_i) + C_{ih},$$

with

$$\begin{aligned}
\mathbf{F}_{ih} &= E_{\boldsymbol{\xi}}\{\nabla_{\theta_i} \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) \nabla_{\theta_h} \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)\}, \\
\mathbf{H}_{ih} &= E_{\boldsymbol{\xi}}\{\nabla_{\theta_i} \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) \nabla_{\theta_h} \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) r(\boldsymbol{\xi})\}, \\
\mathbf{C}_{ih} &= E_{\boldsymbol{\xi}}\{\nabla_{\theta_i} \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) \nabla_{\theta_h} \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1) r^2(\boldsymbol{\xi})\}.
\end{aligned}$$

The term $\mathbf{C}_{ih}$ is not relevant for the optimization problem. When just minimizing $\sigma^2 = \mathrm{Tr}\,\boldsymbol{\Sigma}$, we obtain Equation (3.48). When minimizing the average of all elements Σ_{ih} of $\boldsymbol{\Sigma}$, we obtain

$$\bar{\Sigma} = \sum_{i=1}^{L} \sum_{h=1}^{L} \Sigma_{ih} = \mathbf{b}^T \mathbf{F} \mathbf{b} - 2\mathbf{h}^T \mathbf{b} + c, \tag{3.50}$$

where $\mathbf{h} = \sum_{h=1}^{L} \mathbf{H}_{ih}$, $c = \sum_{i=1}^{L} \sum_{h=1}^{L} \mathbf{C}_{ih}$. This has an obvious solution given by Equation (3.49). ■

Theorem 7 shows that the optimal scalar baselines is nothing only but a projection of the multidimensional baselines.

3.2.2.2 Biased Variance Reduction through Credit Assignment

In sequential problems, we have seen the eligibility terms

$$\nabla_\theta \log d_k^\pi(\mathbf{x}) \approx \sum_{h=0}^{k} \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_h|\mathbf{x}_h, h), \tag{3.51}$$

and have realized that these grow over time k. This growth only holds when the mixing time k^* of the Markov chain has been reached where $d_k^\pi(\mathbf{x}) \approx d_\infty^\pi(\mathbf{x})$. From that point the summands start averaging each other out. However, in many practical cases, the mixing time is large and can never be reached in practice. Furthermore, the stationary distribution may be so large that it will never be completely visited. In fact, in many cases the actions past the mixing time are not even relevant, e.g., the actions taken after the T-ball player in the example in Section 3.1.2.1 has hit the ball will barely be relevant to achieving the hitting in the first place. Therefore, alternate ways have to be found in order to prevent these sums to become too large.

Nevertheless, not all past actions have contributed to the reward but will obtain a large amount of credit simply because they happened early on in the action sequence. This makes our goal clear: How can we give past actions less credit for the newly received reward? If appropriately chosen, such credit assignment will limit the size and the variance of the terms.

Exponentially Weighted Eligibilities The problem of credit assignment to past actions is well-known in the value function based literature; Sutton and Barto (1998) spends a whole chapter on the application of exponentially weighted eligibilities when used for estimating value functions[12]. For gradient methods, it has been noted exponential forgetting of the contributing eligibility terms

$$\nabla_\theta \log d_k^\pi(\mathbf{x}) \approx (1-\lambda) \sum_{h=0}^{k} \lambda^{k-h} \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_h|\mathbf{x}_h, h), \tag{3.52}$$

with a forgetting factor $\lambda \in [0, 1]$ can reduce the variance in the gradient estimate significantly (Kimura & Kobayashi, 1998; Baxter, Bartlett, & Weaver, 2001; Baxter & Bartlett, 1999) while biasing the gradient estimate (Baxter et al., 2001; Baxter & Bartlett, 1999). As long as the gradient is still decrescent, i.e., Equation (3.18) holds, the convergence to the optimal solution can be guaranteed. Nevertheless, this is obviously not the case in general as a reweighting can bias the gradient arbitrarily. However, if the factor λ is set to $0 \leq \lambda_0 < 1$, this can be used to reduce the variance in the gradient estimate initially by biasing the gradient to go for the optimal solution for

[12]However, note that for value function methods, eligibilities do not bias the value function estimate for ideal basis functions. This is not the case for eligibilities in gradient methods where they change the optimal solution like a discounting factor.

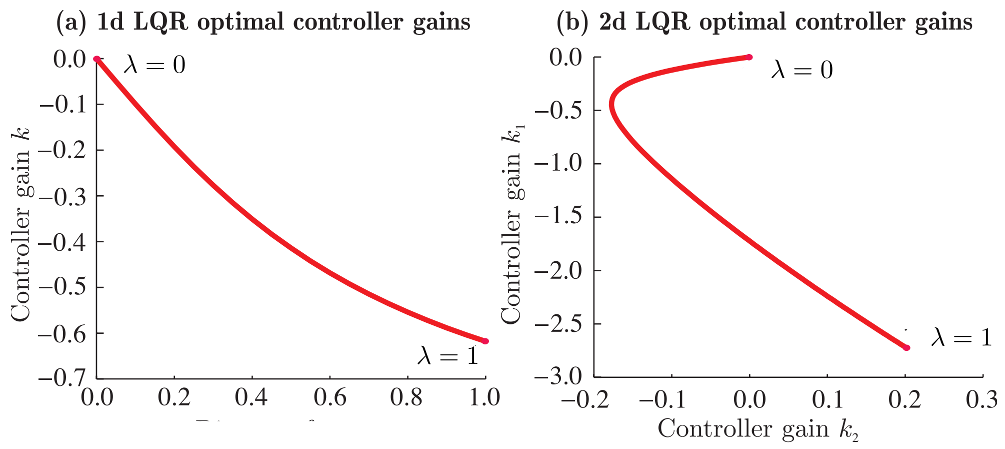


Figure 3.2: This figure shows that the optimal controller parameters are in fact functions of the eligibility rate λ. In (a), you can see the optimal controller parameter k for a one dimensional LQR system with $A = b = Q = R = 1$ as a function of the discount rate γ. In (b), the optimal controller parameters k_1 and k_2 of a two dimensional LQR system with $\mathbf{A} = \text{diag}(1, 2)$, $\mathbf{b} = [2, 1]^T$, $\mathbf{Q} = \mathbf{I}$, and $R = 1$ are given as in implicit function of γ. It is clear from these plots, that the discount rate affects the optimal controller parameters.

a short horizon initially. When gradually increasing λ from λ_0 to 1, we increase the planning horizon and can make sure that we will converge to the optimal solution.

While there are significant results using such biased gradient methods (Baxter et al., 2001; Baxter & Bartlett, 1999), it is not clear whether this concept is really in general helpful. It is quite easy to show that already for simple problems such as linear quadratic regulation, the exponentially weighted gradient converges to significantly different optimal solution $\boldsymbol{\theta}^*$ depending on the parameter λ as shown in Figure 3.2. The chosen optimal solutions can be distributed over most of the search space – we clearly gain little if go away from our optimal solution by using a lower horizon $\lambda_0 < 1$ as demonstrated in Figure 3.2.

Estimated Elibilities As an interesting alternative to exponential forgetting, we can also estimate the eligibility weightings of the different time steps. Let us discuss this for general sequential problems with no assumptions on the rewards using Equation (3.25), where we approximate the gradient with the estimatior

$$\nabla_{\theta_i} J(\theta) \approx E_{\boldsymbol{\xi}} \left\{ \left(\sum_{k=0}^{n-1} \tilde{\lambda}_{k,i} \log \pi_{\boldsymbol{\theta}_i}(\mathbf{u}_k | T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k) \right) (r(\boldsymbol{\xi}) - b(\mathbf{x}_0)) \right\}, \tag{3.53}$$

with $\tilde{\lambda}_{k,i} \in \mathbb{R}$ being a sequence of numbers which tells us how much credit we give this time-step for achieving the rewards. Note, by setting $\tilde{\lambda}_{k,i} = \lambda^{n-1-k}$, we can turn this estimate into exponential forgetting. As pointed out in (Lawrence et al., 2003), for the eligibility weightings $\tilde{\lambda}_{k,i}$, we can often do guaranteed better than this. Just as in (Lawrence et al., 2003), we rewrite the gradient estimator into

$$\boldsymbol{\Delta} = \boldsymbol{\Lambda} \mathbf{V}^T, \tag{3.54}$$

with $\boldsymbol{\Lambda}_{ki} = \tilde{\lambda}_{k,i}$ and $\mathbf{V}_i^T = \log \pi_{\boldsymbol{\theta}_i}(\mathbf{u}_k | T(\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}), k) \, (r(\boldsymbol{\xi}) - b(\mathbf{x}_0))$. For $\boldsymbol{\Lambda} = \mathbf{1}$, we have the unbiased estimator with $\nabla_\theta J(\theta) = \mathbf{1} E_{\boldsymbol{\xi}}\{\mathbf{V}^T\}$.

Theorem 8 *The optimal eligibility matrix is given by*

$$\boldsymbol{\Lambda} = E_{\boldsymbol{\xi}}\{\mathbf{V}^T \mathbf{V}\}^{-1} E_{\boldsymbol{\xi}}\{\mathbf{V}\}^T \nabla_\theta J(\theta), \tag{3.55}$$

Proof. As we cannot guarantee unbiasedness, we can only simplify the gradient estimator into

$$\begin{aligned} \boldsymbol{\Sigma} &= E_{\boldsymbol{\xi}}\{\boldsymbol{\Delta}\boldsymbol{\Delta}^T\} - 2E_{\boldsymbol{\xi}}\{\boldsymbol{\Delta}\}\nabla_\theta J(\theta)^T + \|\nabla_\theta J(\theta)\|^2, \\ &\propto E_{\boldsymbol{\xi}}\{\boldsymbol{\Lambda}\mathbf{V}^T\mathbf{V}\boldsymbol{\Lambda}^T\} - 2E_{\boldsymbol{\xi}}\{\boldsymbol{\Lambda}\mathbf{V}^T\}\nabla_\theta J(\theta)^T, \\ &= \boldsymbol{\Lambda} E_{\boldsymbol{\xi}}\{\mathbf{V}^T\mathbf{V}\}\boldsymbol{\Lambda}^T - 2\boldsymbol{\Lambda} E_{\boldsymbol{\xi}}\{\mathbf{V}\}^T \nabla_\theta J(\theta)^T. \end{aligned}$$

We can minize the whole covariance matrix for every single element which results into Equation (3.55). ■

As $\nabla_\theta J(\theta)$ is unknown, Theorem 8 appears rather useless on the first inspection. Lawrence et al. obtain upper bounds $\hat{\boldsymbol{\Lambda}}_{ki}$ on the matrix elements $\boldsymbol{\Lambda}_{ki} = \tilde{\lambda}_{k,i}$ so that $\boldsymbol{\Lambda}_{ki} \leq \hat{\boldsymbol{\Lambda}}_{ki} \leq 1$. However, their results show little improvements over the compared methods without estimated eligibilities.

3.2.3 Compatible Function Approximation

As we have noted before, the accumulated reward $r(\boldsymbol{\xi})$ of a history can be noisy, i.e., $r(\boldsymbol{\xi}) = \bar{r}(\boldsymbol{\xi}) + \epsilon$. Particularly, this is the case if we are estimating just $Q_k^\pi(\mathbf{x}, \mathbf{u})$ for the gradient given in Equation (3.33) which is after all the central problem of value function reinforcement learning. In such cases, function approximation for the reward function is required – one of the main difficulties of value function methods. However, policy gradient methods come with an important additional feature, i.e., they give a clear form of what kind of linear function approximation $f_{\boldsymbol\beta}(\boldsymbol\xi) = \boldsymbol\phi(\boldsymbol\xi)^T\boldsymbol\beta$ can be used for approximating $r(\boldsymbol\xi)$ without biasing the gradient.

Theorem 9 *The only linear function approximation $f_{\boldsymbol\beta}(\boldsymbol\xi)$ which approximates $r(\boldsymbol\xi)$ in a mean-squared error sense*

$$\min_{\boldsymbol\beta} \mathrm{MSE}(\boldsymbol\omega) = \min_{\boldsymbol\beta} E_{\boldsymbol\xi}\left\{\left(r(\boldsymbol\xi) - f_{\boldsymbol\beta}(\boldsymbol\xi)\right)^2\right\}. \tag{3.56}$$

which does not bias the gradient estimator

$$\nabla_\theta J(\theta) = E_{\boldsymbol\xi}\left\{\nabla_\theta \log p_\theta(\boldsymbol\xi_2|\boldsymbol\xi_1)\left(f_{\boldsymbol\beta}(\boldsymbol\xi) - b(\boldsymbol\xi_1)\right)\right\}, \tag{3.57}$$

his given by

$$f_{\boldsymbol\beta}(\boldsymbol\xi) = \boldsymbol\phi(\boldsymbol\xi_1, \boldsymbol\xi_2)^T\boldsymbol w + \boldsymbol\psi(\boldsymbol\xi_1)^T\boldsymbol\nu, \tag{3.58}$$

with the basis functions

$$\boldsymbol\phi(\boldsymbol\xi_1, \boldsymbol\xi_2) = \nabla_\theta \log p_\theta(\boldsymbol\xi_2|\boldsymbol\xi_1), \tag{3.59}$$

while $\boldsymbol\psi(\boldsymbol\xi_1)$ can be chosen arbitrarily. The parameters of the approximator are $\boldsymbol\beta = [\boldsymbol w, \boldsymbol\nu]$.

Proof. If the mean-squared error is minimized with respect to the parameters $\boldsymbol\omega$, we have

$$E_{\boldsymbol\xi}\left\{\left(r(\boldsymbol\xi) - f_{\boldsymbol\beta}(\boldsymbol\xi)\right)\nabla_{\boldsymbol\beta} f_{\boldsymbol\beta}(\boldsymbol\xi)\right\} = \mathbf{0}.$$

which combined with Equations (3.58) yields

$$E_{\boldsymbol\xi}\left\{\left(r(\boldsymbol\xi) - f_{\boldsymbol\beta}(\boldsymbol\xi)\right)\left[\boldsymbol\phi(\boldsymbol\xi_1, \boldsymbol\xi_2)^T, \boldsymbol\psi(\boldsymbol\xi_1)^T\right]^T\right\} = \mathbf{0}.$$

As Equation (3.59) yields

$$E_{\boldsymbol\xi}\left\{\left(r(\boldsymbol\xi) - f_{\boldsymbol\beta}(\boldsymbol\xi)\right)\nabla_\theta \log p_\theta(\boldsymbol\xi_2|\boldsymbol\xi_1)\right\} = \mathbf{0}, \tag{3.60}$$

we realize that the error in $f_{\boldsymbol\beta}(\boldsymbol\xi)$ has to be orthogonal to the gradient $\nabla_\theta p_\theta(\boldsymbol\xi_2|\boldsymbol\xi_1)$. As the expression is zero we can subtract it from the likelyhood ratio gradient with baseline in Equation (3.22) and obtain

$$\nabla_\theta J(\theta) = E_{\boldsymbol\xi}\left\{\nabla_\theta \log p_\theta(\boldsymbol\xi_2|\boldsymbol\xi_1)\left(f_{\boldsymbol\beta}(\boldsymbol\xi) - b(\boldsymbol\xi_1)\right)\right\}. \tag{3.61}$$

We see that component $\boldsymbol\psi(\boldsymbol\xi_1)^T\boldsymbol\nu$ can be chosen arbitrarily. It cannot bias the gradient as it can be seen as part of the baseline, i.e., $b(\boldsymbol\xi_1) = \tilde b(\boldsymbol\xi_1) + \boldsymbol\psi(\boldsymbol\xi_1)^T\boldsymbol\nu$. ■

In the proof of Theorem 9, we have realized that all additional basis functions are simply part of the baseline. This poses the question whether the estimated baseline $b(\boldsymbol\xi_1) = \boldsymbol\psi(\boldsymbol\xi_1)^T\boldsymbol\nu$ is of any significance. Let us assume the scenario that we estimate $\boldsymbol w$ with a seperate estimator, i.e., we have a mean-squared error as a function of $\boldsymbol\nu$ given by

$$\mathrm{MSE}(\boldsymbol{v}) = E_{\boldsymbol{\xi}}\left\{\boldsymbol{v}^T\boldsymbol{\psi}(\boldsymbol{\xi}_1)\boldsymbol{\psi}(\boldsymbol{\xi}_1)^T\boldsymbol{v} + \left(r(\boldsymbol{\xi}) - \boldsymbol{\phi}(\boldsymbol{\xi}_1,\boldsymbol{\xi}_2)^T\boldsymbol{w}\right)2\boldsymbol{\psi}(\boldsymbol{\xi}_1)^T\boldsymbol{v} \right. \tag{3.62}$$

$$\left. + \left(r(\boldsymbol{\xi}) - \boldsymbol{\phi}(\boldsymbol{\xi}_1,\boldsymbol{\xi}_2)^T\boldsymbol{w}\right)\left(r(\boldsymbol{\xi}) - \boldsymbol{\phi}(\boldsymbol{\xi}_1,\boldsymbol{\xi}_2)^T\boldsymbol{w}\right)\right\}. \tag{3.63}$$

As $E_{\boldsymbol{\xi}_2}\left\{\boldsymbol{\phi}(\boldsymbol{\xi}_1,\boldsymbol{\xi}_2)^T\boldsymbol{w}\right\} = 0$, we have a least-squares problem with the solution

$$\boldsymbol{v} = E_{\boldsymbol{\xi}}\left\{\boldsymbol{\psi}(\boldsymbol{\xi}_1)\boldsymbol{\psi}(\boldsymbol{\xi}_1)^T\right\}^{-1}E_{\boldsymbol{\xi}}\left\{\boldsymbol{\psi}(\boldsymbol{\xi}_1)r(\boldsymbol{\xi})\right\}. \tag{3.64}$$

This solution is not the same as the one for the parametric minimum-variance baseline in Section 3.2.2.1. However, it is the minimum variance baseline with respect to a metric – if we reformulate the baseline derivations in Section 3.2.2.1, we realize that if the problem of having the minimum variance baseline for an unbiased gradient with respect to the metric

$$m(\boldsymbol{\xi}) = \nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1)^T\nabla_\theta \log p_\theta(\boldsymbol{\xi}_2|\boldsymbol{\xi}_1), \tag{3.65}$$

we arrive at the baseline in Equation (3.64). At this point, this metric defies our comprehension of the topic and it might even appear “wrong” to estimate the baseline like this. Nevertheless, it will become very important in Section 3.3 where the metric will suddenly re-appear from a different point of view. However, this is not the only strange change in comparison to the optimal baselines. A baseline in Section 3.2.2.1 could also be a vector, i.e., there would be one baseline per parameter. Baseline in the sense of the compatible function approximation has to be a scalar.

We intend to find the minimum variance unbiased estimator so that we can find the optimal $\boldsymbol{\omega}$ in the sense of Equation (3.56) based upon data from roll-outs; this data corresponds to the rewards and basis functions of the m rollouts of length n. In order to achieve this, we rewrite Equation (3.56) as

$$\mathrm{MSE} = (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}), \tag{3.66}$$

with $\boldsymbol{\beta} = [\boldsymbol{w}, \boldsymbol{v}]$ denoting possible compatible function approximations, $\mathbf{X}_i = [\boldsymbol{\phi}(\boldsymbol{\xi}_1^i, \boldsymbol{\xi}_2^i); \boldsymbol{\psi}(\boldsymbol{\xi}_1^i)]^T$ denoting the basis functions for the data points, and $\mathbf{Y}_i = r(\boldsymbol{\xi}^i)$ denoting the targets of the regression. The minimum variance unbiased estimator to this regression problem is given by

$$\boldsymbol{\beta}^* = \mathrm{argmin}_{\boldsymbol{\beta}}(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}. \tag{3.67}$$

We will now derive such estimators for the compatible function approximation of all three kind of gradients explained before, i.e., we attempt to find a compatible function estimator for general gradient estimators such as in Equation (3.25), and for both the REINFORCE/GPOMDP in Equation (3.26) and Policy Gradient Theorem in Equation (3.34). While these are all the same in expectation, their baselines result in fairly specialized estimators.

3.2.3.1 Reward Sequence Approximation for General Gradient Estimators

As we have seen in Section 3.2.1.2 for the estimator g_1, the baseline b can only be a constant, the compatible function approximation is obviously given by

$$f_w(\boldsymbol{\xi}^j) = \left(\sum_{\tau=0}^{N} \nabla_\theta \log \pi(\mathbf{u}_\tau^j | T(\mathbf{x}_{0:\tau}^j, \mathbf{u}_{0:\tau-1}^j), \tau)\right)^T \boldsymbol{w}, \tag{3.68}$$

for history $\boldsymbol{\xi}^j$ and the targets are the accumulated rewards along the trajectory $r(\boldsymbol{\xi}^j)$. When we bring this into the standard regression form in Equation (3.66), the basis functions are given by

$$\mathbf{X}^T = \begin{bmatrix} \boldsymbol{\phi}_{1:n}^1, & \boldsymbol{\phi}_{1:n}^2, & \ldots, & \boldsymbol{\phi}_{1:n}^m \\ 1, & 1, & \ldots, & 1 \end{bmatrix}, \tag{3.69}$$

where $\boldsymbol{\phi}_{1:n}^j = \sum_{\tau=0}^{N} \nabla_\theta \log \pi(\mathbf{u}_\tau^j | T(\mathbf{x}_{0:\tau}^j, \mathbf{u}_{0:\tau-1}^j), \tau)$ denotes the log-probability of the j-th roll-out, and the targets are given by

$$\mathbf{Y}^T = \begin{bmatrix} r_{1:n}^1, & r_{1:n}^2, & \ldots, & r_{1:n}^m \end{bmatrix}, \tag{3.70}$$

where $r_{1:n}^j = r(\boldsymbol{\xi}^j)$ denotes the sum of the rewards of the j-the roll-out. The solution $\boldsymbol{\beta}^* = [\boldsymbol{w}^T, b]^T$ can then be derived as shown in Theorem 10.

Theorem 10 *The solution* $\boldsymbol{\beta}^* = [\boldsymbol{w}_1^T, b_1]^T$ *to the regression problem can be given by*

$$\boldsymbol{w}_1 = \mathbf{F}_1^{-1}\mathbf{g}_1, \tag{3.71}$$

$$b_1 = m^{-1}\left(1 + \bar{\boldsymbol{\phi}}^T \left(m\mathbf{F}_1 - \bar{\boldsymbol{\phi}}\bar{\boldsymbol{\phi}}^T\right)^{-1} \bar{\boldsymbol{\phi}}\right)\left(\bar{r} - \bar{\boldsymbol{\phi}}^T \mathbf{F}_1^{-1}\mathbf{g}\right) \tag{3.72}$$

with Fisher information $\mathbf{F}_1$*, average eligibility* $\bar{\boldsymbol{\phi}}$*, average reward* $\bar{r}$*,policy gradient with baseline* $\mathbf{g}_1$ *and without baseline* $\mathbf{g}$ *given by*

$$\mathbf{F}_1 = \sum_{j=1}^{m} \boldsymbol{\phi}_{1:n}^j (\boldsymbol{\phi}_{1:n}^j)^T,\ \bar{\boldsymbol{\phi}} = \sum_{j=1}^{m} \boldsymbol{\phi}_{1:n}^j,\ \bar{r} = \sum_{j=1}^{m} r_{1:n}^j, \tag{3.73}$$

$$\mathbf{g}_1 = \sum_{j=1}^{m} \boldsymbol{\phi}_{1:n}^j \left(r_{1:n}^j - b_1\right),\ \mathbf{g} = \sum_{j=1}^{m} \boldsymbol{\phi}_{1:n}^j r_{1:n}^j. \tag{3.74}$$

Proof. We make use of Theorem 17 in the Appendix. We first obtain

$$\mathbf{X}_1^T\mathbf{X}_1 = \mathbf{F}_1,\ \mathbf{X}_1^T\mathbf{X}_2 = \bar{\boldsymbol{\phi}},\ \mathbf{X}_2^T\mathbf{X}_2 = m,\ \mathbf{X}_1^T\mathbf{Y} = \mathbf{g} = \mathbf{g}_1 + \bar{\boldsymbol{\phi}}\, b,\ \mathbf{X}_2^T\mathbf{Y} = \bar{r}. \tag{3.75}$$

We insert these into Equations (A.2, A.3, A.4) from Theorem 17, and obtain

$$\boldsymbol{w} = \boldsymbol{\beta}_1 = \mathbf{F}_1^{-1}\left(\mathbf{g} - \bar{\boldsymbol{\phi}}\mathbf{b}\right) = \mathbf{F}_1^{-1}\mathbf{g}_1, \tag{3.76}$$

$$b = \beta_2 = \mathbf{Q}^{-1}\left(\bar{r} - \bar{\boldsymbol{\phi}}^T\mathbf{F}_1^{-1}\mathbf{g}\right), \tag{3.77}$$

with $\mathbf{Q}^{-1} = m^{-1}(1 + \bar{\boldsymbol{\phi}}^T(m\mathbf{F}_1 - \bar{\boldsymbol{\phi}}\bar{\boldsymbol{\phi}}^T)^{-1}\bar{\boldsymbol{\phi}})$. ■

Note that this gradient estimator is in fact using exactly the REINFORCE gradient and just one, constant baseline. This can alternatively be derived using Suttons form by adding up the advantages along a path and is also known as episodic natural actor-critic (Peters et al., 2003a, 2003b). As this point, it might appear that we have thrown the child out together with the water as we need the reward sequence to approximate the gradient and at the same time need the gradient to approximate the reward sequence.

3.2.3.2 Reward Prediction for G(PO)MDP Estimators

As we have seen in Section 3.2.1.2 for the estimator $\mathbf{g}_2$, the baseline b depends only on time (and the initial state), the compatible function approximation is obviously given by

$$f_w(\boldsymbol{\xi}^j_{0:k}) = \left(\sum_{\tau=0}^{k} \nabla_\theta \log \pi(u_\tau|x_\tau, \tau)\right)^T w, \tag{3.78}$$

and the targets are the actual rewards along the trajectory, i.e., $r(\boldsymbol{\xi}_{0:t}) = r_t$. When we bring this into the standard regression form in Equation (3.66), the basis functions are given by

$$\mathbf{X} = \begin{bmatrix} \boldsymbol{\phi}^1_{1:1}, & \boldsymbol{\phi}^1_{1:2}, & \ldots, & \boldsymbol{\phi}^1_{1:n}, & \boldsymbol{\phi}^2_{1:1}, & \ldots, & \boldsymbol{\phi}^m_{1:n} \\ \mathbf{u}_1, & \mathbf{u}_2, & \ldots, & \mathbf{u}_n, & \mathbf{u}_1, & \ldots, & \mathbf{u}_n \end{bmatrix}, \tag{3.79}$$

where $\boldsymbol{\phi}^j_{1:n} = \sum_{t=1}^n \nabla_\theta \log \pi(u^j|x^j, t)$ denotes the log-probability of the j-th roll-out, and $\mathbf{u}_i$ denotes the i-th unit vector basis function of length n. The targets are given by

$$\mathbf{Y} = \begin{bmatrix} r^1_1, & r^1_2, & \ldots, & r^1_n, & r^2_1, & \ldots, & r^m_n \end{bmatrix}, \tag{3.80}$$

where r^j_t denotes the rewards at time t of the j-the roll-out. The solution $\boldsymbol{\beta}^* = [\boldsymbol{w}^T, b]^T$ can then be derived as shown in Theorem 11.

Theorem 11 *The solution $\boldsymbol{\beta}^*_2 = [\boldsymbol{w}^T_2, b_2]^T$ to the regression problem can be given by*

$$\boldsymbol{w}_2 = \mathbf{F}_2^{-1}\mathbf{g}_2, \tag{3.81}$$

$$\mathbf{b}_2 = m^{-1}\left(\mathbf{I}_n + \bar{\boldsymbol{\Phi}}^T\left(m\mathbf{F}_2 - \bar{\boldsymbol{\Phi}}\bar{\boldsymbol{\Phi}}^T\right)^{-1}\bar{\boldsymbol{\Phi}}\right)\left(\bar{\mathbf{r}} - \bar{\boldsymbol{\Phi}}^T\mathbf{F}_1^{-1}\mathbf{g}\right) \tag{3.82}$$

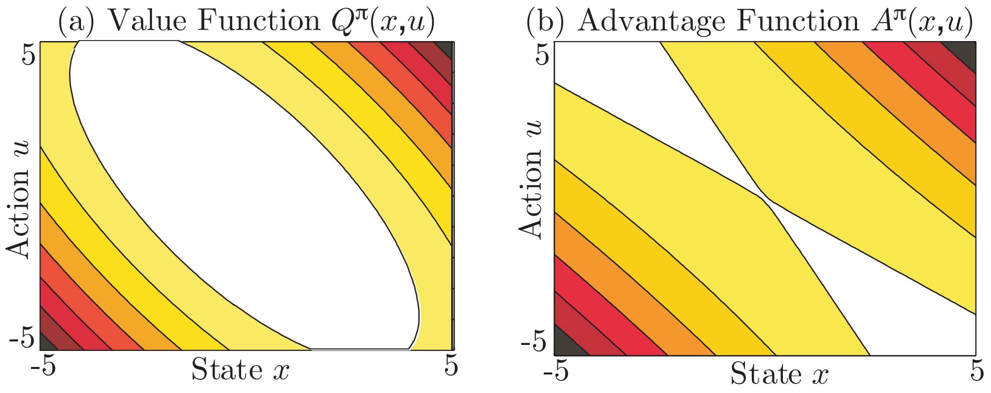


Figure 3.3: This figure shows the state-action value function and the advantage function for LQR. It is obvious that these have very little structure in common as one is bowl while the other is a saddle.

with Fisher information $\mathbf{F}_1$*, average eligibility* $\bar{\boldsymbol{\phi}}$*, average reward* $\bar{\mathbf{r}}$*, policy gradient with baseline* $\mathbf{g}_1$ *and without baseline* $\mathbf{g}$ *given by*

$$\mathbf{F}_2 = \sum_{j=1}^{m}\sum_{i=1}^{n} \boldsymbol{\phi}_{1:i}^{j}\left(\boldsymbol{\phi}_{1:i}^{j}\right)^T, \; \bar{\boldsymbol{\Phi}} = \sum_{j=1}^{m}\sum_{i=1}^{n} \boldsymbol{\phi}_{1:i}^{j}\mathbf{e}_i^T, \; \bar{\mathbf{r}} = \sum_{j=1}^{m}\sum_{i=1}^{n} r_i^j \mathbf{e}_i, \tag{3.83}$$

$$\mathbf{g}_2 = \sum_{j=1}^{m}\sum_{i=1}^{n} \boldsymbol{\phi}_{1:i}^{j}\left(r_i^j - b_i\right), \; \mathbf{g} = \sum_{j=1}^{m}\sum_{i=1}^{n} \boldsymbol{\phi}_{1:i}^{j} r_i^j. \tag{3.84}$$

Proof. We make use of Theorem 17 in the Appendix. We first obtain

$$\mathbf{X}_1^T\mathbf{X}_1 = \mathbf{F}_2, \; \mathbf{X}_1^T\mathbf{X}_2 = \bar{\boldsymbol{\Phi}}, \; \mathbf{X}_2^T\mathbf{X}_2 = m\mathbf{I}_n, \; \mathbf{X}_1^T\mathbf{Y} = \mathbf{g} = \mathbf{g}_2 + \bar{\boldsymbol{\Phi}}\mathbf{b}_2, \; \mathbf{X}_2^T\mathbf{Y} = \bar{\mathbf{r}}. \tag{3.85}$$

We insert these into Equations (A.2, A.3, A.4) from Theorem 17, and obtain

$$\boldsymbol{w} = \boldsymbol{\beta}_1 = \mathbf{F}_2^{-1}\left(\mathbf{g} - \bar{\boldsymbol{\Phi}}\mathbf{b}\right) = \mathbf{F}_2^{-1}\mathbf{g}_2, \tag{3.86}$$

$$\mathbf{b} = \beta_2 = \mathbf{Q}^{-1}\left(\bar{r} - \bar{\boldsymbol{\Phi}}^T\mathbf{F}_2^{-1}\mathbf{g}\right), \tag{3.87}$$

with $\mathbf{Q}^{-1} = m^{-1}(\mathbf{I}_n + \bar{\boldsymbol{\Phi}}^T(m\mathbf{F}_2 - \bar{\boldsymbol{\Phi}}\bar{\boldsymbol{\Phi}}^T)^{-1}\bar{\boldsymbol{\Phi}})$. ■

Note that this gradient compatible reward estimator is in fact using exactly the GPOMDP gradient with a time-variant scalar baseline b_k. However, just like in the last section, we not the apparant problem.

3.2.3.3 Compatible Value Function Approximation for the Policy Gradient Theorem

The whole idea of compatible function approximation was originally phrased in terms of approximating the value function $Q^\pi(\mathbf{x}, \mathbf{u})$ without biasing the gradients (Konda, 2002; Sutton et al., 2000a; Konda & Tsitsiklis, 2000, 2001), i.e., it was thought up for the third case. While it is obvious that it is for the third case more important than for the other two as $Q^\pi(\mathbf{x}, \mathbf{u})$ is never available without large noise, it is also the most difficult as the baseline depends on the state $\mathbf{x}$ and a good parametric baseline is often not available apriori.

At this point, a most important observation is that the compatible function approximation $f^\pi_{\boldsymbol{w}}(\mathbf{x}, \mathbf{u})$ is mean-zero with respect to the action distribution, i.e.,

$$\int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}) f^\pi_{\boldsymbol{w}}(\mathbf{x}, \mathbf{u}) d\mathbf{u} = \int_{\mathbb{U}} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \pi(\mathbf{u}|\mathbf{x})^T d\mathbf{u} \; \boldsymbol{w} = 0, \tag{3.88}$$

since from $\int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}) d\mathbf{u} = 1$, differention with respect to θ results in $\int_{\mathbb{U}} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \pi(\mathbf{u}|\mathbf{x}) d\mathbf{u} = \mathbf{0}$ (which is equivalent to the baseline in Lemma). Thus, $f^\pi_w(\mathbf{x}, \mathbf{u})$ can in general only represent an *advantage function* $A^\pi(\mathbf{x}, \mathbf{u}) = Q^\pi(\mathbf{x}, \mathbf{u}) - V^\pi(\mathbf{x})$. This different structure makes the learning problem significantly more difficult as these two functions have very little structure in common as can be observed for the example of linear quadratic regulation in Figure 3.3. We will show in this section, how this problem can be tackled from three different points of view.

Monte Carlo Approach. In (Sutton et al., 2000b) and (Konda & Tsitsiklis, 2000) it was suggested to estimate $f^\pi_w(\mathbf{x}, \mathbf{u})$ from unbiased estimates $\hat{Q}^\pi(\mathbf{x}, \mathbf{u})$ of the action value function, e.g., obtained from roll-outs and using least-squares minimization between f_w and $\hat{Q}$. While possible in theory, one needs to realize that this approach implies a function approximation problem where the parameterization of the function approximator only spans a much smaller subspace of the training data – e.g., imagine approximating a quadratic function with a line. In practice, the results of such an approximation depends crucially on the training data distribution and has thus unacceptably high variance – e.g., fit a line to only data from the right branch of a parabula, the left branch, or data from both branches.

However, when we add the parametric baseline $b(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x})^T \boldsymbol{\nu}$ into the least-squares problem as in previous sections, we can obtain an easiert estimation problem. In this case, we can write our least squares problem in the standard regression form in Equation (3.66) and the basis functions are given by

$$\mathbf{X} = \begin{bmatrix} \boldsymbol{\phi}^1_1, & \boldsymbol{\phi}^1_2, & \ldots, & \boldsymbol{\phi}^1_n, & \boldsymbol{\phi}^2_1, & \ldots, & \boldsymbol{\phi}^m_n \\ \boldsymbol{\psi}_1, & \boldsymbol{\psi}_2, & \ldots, & \boldsymbol{\psi}_n, & \boldsymbol{\psi}_1, & \ldots, & \boldsymbol{\psi}_n \end{bmatrix}, \tag{3.89}$$

where $\boldsymbol{\phi}_k^j = \nabla_\theta \log \pi(u^j|x^j, k)$ denotes the log-probability derivative of the k-th time-step of the j-th roll-out, and $\boldsymbol{\psi}_k = \boldsymbol{\psi}(\mathbf{x}_k)$ denotes the additional basis functions at that time-step. The targets are given by

$$\mathbf{Y} = \left[r^1_{1:n}, \quad r^1_{2:n}, \quad \ldots, \quad r^1_{n:n}, \quad r^2_{1:n}, \quad \ldots, \quad r^m_{n:n}\right], \tag{3.90}$$

where r_t^j denotes the rewards at time t of the j-the roll-out. The solution $\boldsymbol{\beta}^* = [\boldsymbol{w}^T, \boldsymbol{v}]^T$ can then be derived as shown in Theorem 11. While we can apply Theorem 17 like before, it will not result in any simplificantions in comparison to Equation (3.67) and can only be used in order to show the baseline does not bias the estimator.

Temporal Difference Learning. Another method suggested in (Konda & Tsitsiklis, 2000) was to apply temporal difference learning to the compatible function approximation. The advantage function *cannot* be learned with temporal difference bootstrapping without knowledge of the value function as it is the essence of temporal difference learning is to compare the value $V^\pi(\mathbf{x})$ of the two adjacent states – but this value has been subtracted out in $A^\pi(\mathbf{x}, \mathbf{u})$. Hence, a TD-like bootstrapping using exclusively the compatible function approximator is impossible. To remedy this situation, we observe that we can write the Bellman equations (e.g., see (Baird, 1993)) in terms of the advantage function and the state-value function

$$Q^\pi(\mathbf{x}, \mathbf{u}) = A^\pi(\mathbf{x}, \mathbf{u}) + V^\pi(\mathbf{x}) = r(\mathbf{x}, \mathbf{u}) + \gamma \int_{\mathbb{X}} p(\mathbf{x}'|\mathbf{x}, \mathbf{u}) V^\pi(\mathbf{x}') d\mathbf{x}',$$

given here in the discounted form. Inserting $A^\pi(\mathbf{x}, \mathbf{u}) = f_w^\pi(\mathbf{x}, \mathbf{u})$ and an appropriate basis functions representation of the value function as $V^\pi(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x})^T \boldsymbol{v} = b(\mathbf{x})$, we can rewrite the Bellman Equation as a set of linear equations

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\mathbf{u}_t|\mathbf{x}_t)^T \boldsymbol{w} + \boldsymbol{\psi}(\mathbf{x}_t)^T \boldsymbol{v} = r(\mathbf{x}_t, \mathbf{u}_t) + \gamma \boldsymbol{\psi}(\mathbf{x}_{t+1})^T \boldsymbol{v} + \epsilon(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) \tag{3.91}$$

where $\epsilon(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$ denotes an mean-zero error term. These equations enable us to formulate some novel algorithms in the next sections. For solving these equations, we cannot plainly apply the normal least-squares reasoning, i.e., just treat $\boldsymbol{\psi}(\mathbf{x}_t) - \boldsymbol{\psi}(\mathbf{x}_{t+1})$ as additional basis function (Sutton et al., 2000a) unless we have a model which lets us explore more actions in the same state[13]. However, at the cost of being biased (Schoknecht, 2003) by the additional basis functions, we can apply LSTD regression methods to this problem (Peters et al., 2003a, 2003b). However, this form requires significantly better basis functions than monte carlo methods just to give us a *good* gradient – bad basis functions can arbitrarily turn the gradient around and for bad basis functions there are examples where the policy can become *worse at every single step* (Peters et al., 2003a, 2003b).

[13]This class of methods is called the *residual gradient methods*. From our experience, they are only efficient if the problem and policy are deterministic or discrete.

Episodic Natural Actor-Critic. Surprisingly, the Bellman equation offers another way out of this dilemma. We can sum up the state-action-next state-reward samples up along a trajectory and obtain the equation

$$\left(\sum_{t=0}^{n} \gamma^n \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\mathbf{u}_t|\mathbf{x}_t)^T \boldsymbol{w}\right) + \gamma^n \boldsymbol{\psi}(\mathbf{x}_n)^T \boldsymbol{v} - \boldsymbol{\psi}(\mathbf{x}_0)^T \boldsymbol{v} = \sum_{t=0}^{n} r(\mathbf{x}_t, \mathbf{u}_t) + \epsilon_t. \qquad (3.92)$$

As $\gamma^n \to 0$ for $n \to \infty$ at exponential pace, we can omit that term and the term $\boldsymbol{\psi}(\mathbf{x}_0)^T \boldsymbol{v}$ can be arbitrary; however, in most cases, a scalar basis function is sufficient. We realize that our regression problem simply becomes

$$\left(\sum_{t=0}^{n} \gamma^n \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\mathbf{u}_t|\mathbf{x}_t)^T\right) \boldsymbol{w} + V = \sum_{t=0}^{n} r(\mathbf{x}_t, \mathbf{u}_t) + \epsilon(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}), \qquad (3.93)$$

which can be shown to be exactly equivalent to the general problem given in Section 3.2.3.1.

3.2.3.4 Notes on the Compatible Function Approximation

In this context, we discuss a few further notes on the compatible function approximation in order to clarify the results of this section.

Is the Compatible Function Approximation useful? What did we gain with the compatible function approximation? The hope was that the compatible function approximation would elivate problems with the gradient estimation as it would reduce the error in the gradient estimate by projecting the reward or value function depending on the problem in a lower-dimensional space. Seemingly, this is not the case as every time when we intend to estimate the compatible function approximation, we seemingly have to estimate the gradient, a complex matrix and the baseline first – just to obtain the parameters of the compatible function approximation. Furthermore, we have noticed that the baseline which is estimated together with the compatible function approximation does not minimize the variance of the vanilla policy gradient estimator – do we need a second baseline?

All Action. However, if we had a relatively good value of $\boldsymbol{w}$, it can make the gradient estimation problem easier to a certain extent. Based on both the policy gradient theorem and the compatible function approximation, we derive an estimate of the policy gradient as

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\mathbf{x}) \int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\mathbf{u}|\mathbf{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\mathbf{u}|\mathbf{x})^T d\mathbf{u} d\mathbf{x}\; \boldsymbol{w} = \mathbf{G}(\boldsymbol{\theta}) \boldsymbol{w}. \qquad (3.94)$$

Since $\pi(\mathbf{u}|\mathbf{x})$ is chosen by the user, even in sampled data, the integral

$$\mathbf{G}(\boldsymbol{\theta},\mathbf{x}) = \int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\mathbf{u}|\mathbf{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\mathbf{u}|\mathbf{x})^T d\mathbf{u} \tag{3.95}$$

can be evaluated analytically or empirically without actually executing all actions. It is also noteworthy that the baseline does not appear in Eq. (3.94) as it integrates out, thus eliminating the need to find the second baseline postulated in the previous paragraph. Nevertheless, the estimation of $\mathbf{G}(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\mathbf{x})\mathbf{G}(\boldsymbol{\theta},\mathbf{x})d\mathbf{x}$ is still expensive since $d^{\pi}(\mathbf{x})$ ist not known.

3.3 Natural Policy Gradients

While there has been a lot of work on obtaining policy gradients as we have seen in the last section, even perfectly estimated or analytically derived policy gradients converge tremendously slow or prematurely and are fragile towards reparameterization. The problems are already apparant for toy problems as can be seen in Figure 3.4 for two toy problems. As shown in (Kakade, 2001) even higher order methods such as Hessian based gradients do not help. This has led Kakade to question whether we are obtaining the *right* gradient, i.e., the gradient which is invariant to reparameterization and converges robustly for most parameterizations.

Amari (1998) suggested that in such cases, the natural policy gradient is preferable to the policy gradient. A natural gradient is the one with respect to the Fisher information matrix

$$\widetilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbf{F}^{-1}(\boldsymbol{\theta}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}). \tag{3.96}$$

This led Kakade (2001) to investigate the possibility of researching of natural gradients in reinforcement learning. As we have noted before in Section 3.2.3.4, when inserting the compatible function approximation into the gradient, we obtain

$$\nabla_{\theta} J(\theta) = E_{\boldsymbol{\xi}}\{\nabla_{\theta} \log p_{\theta}(\boldsymbol{\xi}) \nabla_{\theta} \log p_{\theta}(\boldsymbol{\xi})^T\} w = \mathbf{G}(\theta) w, \tag{3.97}$$

where the matrix $\mathbf{G}(\theta)$ does not depend on the rewards at all. In previous work, this has been named the all-action matrix (Peters et al., 2003a, 2003b; Sutton, McAllester, Singh, & Mansour, 2001). Kakade (Kakade, 2001) argued that $\mathbf{G}(\boldsymbol{\theta},\mathbf{x})$ is the point Fisher information matrix for state $\mathbf{x}$, and that $\mathbf{G}(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\mathbf{x})\ \mathbf{G}(\boldsymbol{\theta},\mathbf{x})d\mathbf{x}$, therefore, denotes the 'average Fisher information matrix'. However, going one step further, we demonstrate in Appendix A.2 that $\mathbf{G}(\boldsymbol{\theta}) = \mathbf{F}(\boldsymbol{\theta})$ is indeed the true Fisher information matrix and does not have to be interpreted as the 'average' of the point Fisher information matrices. Therefore, the natural gradient can be computed as

$$\widetilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta}) F(\boldsymbol{\theta}) \boldsymbol{w} = \boldsymbol{w}, \tag{3.98}$$

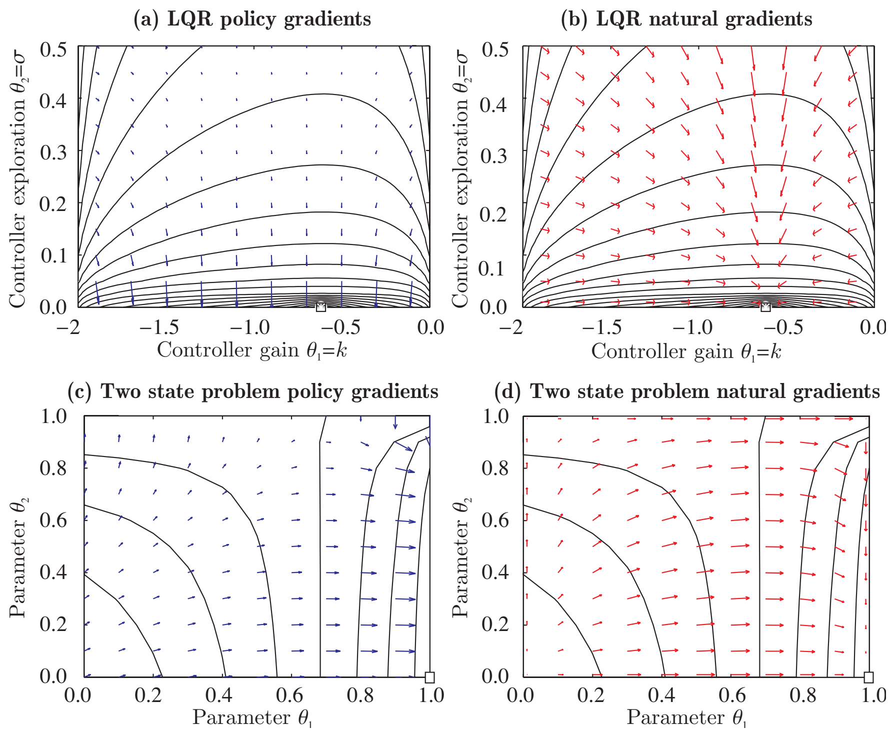


Figure 3.4: This figure compares the natural gradient to the policy gradient. In (a), the policy gradient, and in (b) the natural gradient of the LQR problem with a Gaussian policy is shown. The LQR again had the parameters $A = b = R = Q = 1$, and $\gamma = 0.95$. The natural gradient had to be normalized to be nicely visible. In (c), the policy gradient, and in (d) the natural gradient of the two state problem with a decision border policy. The natural gradient of the two-state problem has not been normalized. The discount factor of the two-state problem is $\gamma = 0.95$.

since $G(\boldsymbol{\theta}) = F(\boldsymbol{\theta})$ (c.f. Appendix A.2). The resulting policy improvement step is thus $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha \boldsymbol{w}$ where α denotes a learning rate.

Several properties of the natural policy gradient are worthwhile highlighting (and will be more substantiated in later sections):

- Convergence guarantees are the same as for 'vanilla gradients' (Amari, 1998)
- By choosing a more direct path to the optimal solution in parameter space, the natural gradient has, from empirical observations, faster convergence and avoids premature convergence of 'vanilla gradients' (cf. Figure 3.4).

- The natural policy gradient can be shown to be **covariant**, i.e., if independent of the coordinate frame chosen for expressing the policy parameters (cf. Section 3.3.1.1).
- As the natural gradient analytically averages out the influence of the stochastic policy (including the baseline of the function approximator), it requires fewer data point for a good gradient estimate than 'vanialla gradients'.

3.3.1 Properties of the Natural Policy Gradient

In this section, we will emphasize certain properties of the natural actor-critic. In particular, we want to give a simple proof of covariance of the natural policy gradient, and discuss Kakade's (2002) observation that in his experimental settings the natural policy gradient was non-covariant. Furthermore, we will discuss another surprising aspect about the Natural Actor-Critic (NAC) which is its relation to previous algorithms. We briefly demonstrate that established algorithms like the classic Actor-Critic archietcture(Sutton & Barto, 1998a), ϵ-soft SARSA (Sutton & Barto, 1998a), and Bradtke's Q-Learning (Bradtke, Ydstie, & Barto, 1994) can be seen as special cases or approximations of NAC.

3.3.1.1 Clarification of the Covariance Discussion

When Kakade (2002) originally suggested natural policy gradients, he came to the disappointing conclusion that they were not covariant. As counterexample, he suggested that for two different linear Gaussian policies, (one in the normal form, and the other in the information form) the probability distributions represented by the natural policy gradient would be affected differently, i.e., the natural policy gradient would be non-covariant. We intend to give a proof at this point showing that the natural policy gradient is in fact covariant under certain conditions, and clarify why Kakade (2002) experienced these difficulties.

Theorem 12 *Natural policy gradients updates are covariant for two policies $\pi_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ and $\pi_{\mathbf{h}}$ parameterized by $\mathbf{h}$ if (i) for all parameters θ_i a function there exists a function $\theta_i = f_i(h_1, \ldots, h_k)$, (ii) the derivative $\boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}$ and its inverse $\boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}^{-1}$ exist, and (iii) the stepsize α is infinitesimally small.*

Proof. For small parameter changes $\boldsymbol{\Delta}\mathbf{h}$ and $\boldsymbol{\Delta\theta}$, we have $\boldsymbol{\Delta\theta} = \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}^T\boldsymbol{\Delta}\mathbf{h}$. If the natural policy gradient is a covariant update rule, a change $\boldsymbol{\Delta}\mathbf{h}$ along the gradient $\boldsymbol{\nabla}_{\mathbf{h}}J(\mathbf{h})$ would result in the same change $\boldsymbol{\Delta\theta}$ along the gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}}J(\boldsymbol{\theta})$ for the same scalar step-size α. By differentiation, we can obtain $\boldsymbol{\nabla}_{\mathbf{h}}J(\mathbf{h}) = \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}\boldsymbol{\nabla}_{\boldsymbol{\theta}}J(\boldsymbol{\theta})$. It is straightforward to show that the Fisher information matrix includes the Jacobian $\boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}$ twice as factor, i.e.,

$$\begin{aligned}
\mathbf{F}(\mathbf{h}) &= \int_{\mathbb{X}} d^{\pi}(\mathbf{x}) \int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}) \boldsymbol{\nabla}_{\mathbf{h}} \log \pi(\mathbf{u}|\mathbf{x}) \boldsymbol{\nabla}_{\mathbf{h}} \log \pi(\mathbf{u}|\mathbf{x})^T d\mathbf{u} d\mathbf{x}, \\
&= \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta} \int_{\mathbb{X}} d^{\pi}(\mathbf{x}) \int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\mathbf{u}|\mathbf{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\mathbf{u}|\mathbf{x})^T d\mathbf{u} d\mathbf{x} \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}^T, \\
&= \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta} \mathbf{F}(\boldsymbol{\theta}) \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}^T.
\end{aligned}$$

This shows straightforwardly that natural gradient in the $\mathbf{h}$ parameterization is given by

$$\widetilde{\boldsymbol{\nabla}}_{\mathbf{h}} J(\mathbf{h}) = \mathbf{F}^{-1}(\mathbf{h}) \boldsymbol{\nabla}_{\mathbf{h}} J(\mathbf{h}) = \left(\boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta} \mathbf{F}(\boldsymbol{\theta}) \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}^T \right)^{-1} \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta} \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}).$$

This has a surprising implication as it makes it straightforward to see that the natural policy is covariant since

$$\begin{aligned}
\boldsymbol{\Delta\theta} &= \alpha \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}^T \boldsymbol{\Delta}\mathbf{h} = \alpha \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}^T \widetilde{\boldsymbol{\nabla}}_{\mathbf{h}} J(\mathbf{h}), \\
&= \alpha \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}^T \left(\boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta} \mathbf{F}(\boldsymbol{\theta}) \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}^T \right)^{-1} \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta} \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \\
&= \alpha \mathbf{F}^{-1}(\boldsymbol{\theta}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \alpha \widetilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}),
\end{aligned}$$

assuming that $\boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}$ is invertable. This concludes that the natural policy gradient is in fact a **covariant** gradient update rule. ■

Practical experiments show that the problems occurred for Gaussian policies in (Kakade, 2002) are in fact due to the selection the stepsize α which determines the length of $\boldsymbol{\Delta\theta}$. As the linearization $\boldsymbol{\Delta\theta} = \boldsymbol{\nabla}_{\mathbf{h}}\boldsymbol{\theta}^T \boldsymbol{\Delta}\mathbf{h}$ does not hold for large $\boldsymbol{\Delta\theta}$, this can cause divergence between the algorithms even for analytically determined natural policy gradients. Theorem 1 can therefore explain the difficulties occurred by Kakade.

Bagnell & Schneider (2002) recently clarified that natural policy gradients were covariant; however, this theorem extends their results by explaining the difficulties occurred by Kakade.

3.3.1.2 Relation to previous algorithms

In this section, we want to clarify the relation of our algorithm to several previous algorithms, i.e., the original Actor-Critic, SARSA and Bradtke's Q-Learning.

Original Q-Learning. Surprisingly, the original Actor-Critic algorithm (Sutton & Barto, 1998a) is a form of the Natural Actor-Critic. By choosing a Gibbs policy $\pi(u_t|x_t) = \exp(\theta_{xu})/\sum_b \exp(\theta_{xb})$, with all parameters θ_{xu} lumped in the vector $\boldsymbol{\theta}$, (denoted as $\boldsymbol{\theta} = [\theta_{xu}]$) in a discrete setup with tabular representations of transition probabilities and rewards. A linear function approximation $V^{\pi}(x) = \boldsymbol{\psi}(x)^T \boldsymbol{\nu}$ with

$\boldsymbol{\nu} = [v_x]$ and unit basis functions $\boldsymbol{\psi}(x) = \mathbf{u}_x$ was employed. Sutton et al.'s (Sutton & Barto, 1998a) online update rule is given by

$$\begin{aligned} \theta_{xu}^{t+1} &= \theta_{xu}^{t} + \alpha_1 \left(r(x,u) + \gamma v_{x'} - v_x \right), \\ v_{x}^{t+1} &= v_{x}^{t} + \alpha_2 \left(r(x,u) + \gamma v_{x'} - v_x \right), \end{aligned}$$

where α_1, α_2 denote learning rates. The update of the critic parameters v_x^t equals the one of the Natural Actor-Critic in expectation as TD(0) critics converges to the same values as LSTD(0) and LSTD-Q(0) for discrete problems (Boyan, 1999). Since for the Gibbs policy we have

$$\frac{\partial \log \pi(b|a)}{\partial \theta_{xu}} = \begin{cases} 1 - \pi(b|a) & \text{if } a = x \text{ and } b = u, \\ -\pi(b|a) & \text{if } a = x \text{ and } b \neq u, \\ 0 & \text{if otherwise,} \end{cases}$$

and $\sum_b \pi(b|x) A^\pi(x,b) = 0$, we can evaluate the advantage function and derive

$$\begin{aligned} A^\pi(x,u) &= A^\pi(x,u) - \underbrace{\textstyle\sum_b \pi(b|x) A^\pi(x,b)}_{=0} \\ &= \sum_b \frac{\partial \log \pi(b|x)}{\partial \theta_{xu}} A^\pi(x,b) = \frac{\partial \log \pi(u|x)}{\partial \boldsymbol{\theta}}^T \mathbf{a}. \end{aligned} \tag{3.99}$$

with $\mathbf{a} = [A^\pi(x,u)]$. Since the compatible function approximation represents the advantage function, i.e., $f_{\boldsymbol{w}}^\pi(\mathbf{x},\mathbf{u}) = A^\pi(x,u)$, we realize that the advantages equal the natural gradient, i.e., $\mathbf{a} = \boldsymbol{w}$. Furthermore, the TD(0) error of a state-action pair (x,u) equals the advantage function in expectation, and therefore the natural gradient update

$$w_{xu} = A^\pi(x,u) = E_{x'}\{r(x,u) + \gamma V^\pi(x') - V^\pi(x)|x,u\},$$

corresponds to the average online updates of Actor-Critic. As both update rules of the Actor-Critic correspond to the ones of NAC, we can see the Original Actor-Critic algorithm as a Natural Actor-Critic algorithm.

SARSA. SARSA with a tabular, discrete state-action value function $Q^\pi(x,u)$ and an ϵ-soft policy improvement $\pi(\mathbf{u}_t|\mathbf{x}_t) = \exp(Q^\pi(x,u)/\epsilon)/\sum_{\hat{u}} \exp(Q^\pi(x,u)/\epsilon)$ can also be seen as an approximation of NAC. When treating the table entries as parameters of a policy $\boldsymbol{\theta}_{xu} = Q^\pi(x,u)$, we realize that the TD update of these parameters corresponds approximately to the natural gradient update since $w_{xu} = \epsilon A(x,u) \approx \epsilon E_{x'}\{r(x,u) + \gamma Q(x',u') - Q(x,u)|x,u\}$. However, the SARSA-TD error equals the advantage function only for policies where a single action u^* has much better action values $Q(x,u^*)$ than all other actions; *for such special cases,* ϵ-soft SARSA can be seen as an approximation of NAC. This also corresponds to Kakade's (2002) observation that greedy update step (such as the ϵ-soft greedy update), approximates the natural policy gradient.

Bradtke's Q-Learning Bradtke et al. (1994) proposed an algorithm with policy $\pi(u_t|\mathbf{x}_t) = \mathcal{N}(u_t|\mathbf{k}_i^T\mathbf{x}_t, \sigma_i^2)$ and parameters $\boldsymbol{\theta}_i = [\mathbf{k}_i^T, \sigma_i]^T$ (where σ_i denotes the exploration, and i the policy update time step) in a linear control task with linear state transitions $\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{b}u_t$, and quadratic rewards $r(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T\mathbf{H}\mathbf{x}_t + Ru_t^2$. They evaluated $Q^\pi(\mathbf{x}_t, \mathbf{u}_t)$ with LSTD(0) using a quadratic polynomial expansion as basis functions, and applied greedy updates:

$$\begin{aligned} \mathbf{k}_{i+1} &= \operatorname{argmax}_{\mathbf{k}_{i+1}} Q^\pi(\mathbf{x}_t, \mathbf{u}_t = \mathbf{k}_{i+1}^T\mathbf{x}_t) && (3.100) \\ &= -(R + \gamma\mathbf{b}^T\mathbf{P}_i\mathbf{b})^{-1}\gamma\mathbf{b}\mathbf{P}_i\mathbf{A}, && (3.101) \end{aligned}$$

where $\mathbf{P}_i$ denotes policy-specific value function parameters related to the gain $\mathbf{k}_i$; no update the exploration σ_i was included. Similarly, we can obtain the natural policy gradient $\boldsymbol{w} = [\boldsymbol{w}_\mathbf{k}, w_\sigma]^T$, as yielded by LSTD-Q($\lambda$) analytically using the compatible function approximation and the same quadratic basis functions. As discussed in detail in (Peters, Vijayakumar, & Schaal, 2004), this gives us

$$\begin{aligned} \boldsymbol{w}_\mathbf{k} &= (\gamma\mathbf{A}^T\mathbf{P}_i\mathbf{b} + (R + \gamma\mathbf{b}^T\mathbf{P}_i\mathbf{b})\mathbf{k})^T\sigma_i^2, && (3.102) \\ w_\sigma &= 0.5(\mathbf{R} + \gamma\mathbf{b}^T\mathbf{P}_i\mathbf{b})\sigma_i^3. && (3.103) \end{aligned}$$

Similarly, it can be derived that the expected return is $J(\boldsymbol{\theta}_i) = -(R + \gamma\mathbf{b}^T\mathbf{P}_i\mathbf{b})\sigma_i^2$ for this type of problems, see (Peters et al., 2004). For a learning rate $\alpha_i = 1/\left\|J(\boldsymbol{\theta}_i)\right\|$, we see that

$$\begin{aligned} \mathbf{k}_{i+1} &= \mathbf{k}_i + \alpha_t\boldsymbol{w}_\mathbf{k} \\ &= \mathbf{k}_i - (\mathbf{k}_i + (R + \gamma\mathbf{b}^T\mathbf{P}_i\mathbf{b})^{-1}\gamma\mathbf{A}^T\mathbf{P}_i\mathbf{b}) \\ &= (R + \gamma\mathbf{b}^T\mathbf{P}_i\mathbf{b})^{-1}\gamma\mathbf{A}^T\mathbf{P}_i\mathbf{b}, && (3.104) \end{aligned}$$

which demonstrates that *Bradtke's Actor Update is a special case of the Natural Actor-Critic.* NAC extends Bradtke's result as it gives an update rule for the exploration – which was not possible in Bradke's greedy framework.

3.3.2 Evaluations and Applications

In this section, we present several evaluations comparing the natural policy gradient estimation with general reward function approximation with previous algorithms. As comparisons, we consider the GPOMDP algorithm as good example for a vanilla policy gradient algorithm, the estimation of the compatible function approximation for the policy gradient theorem (Konda & Tsitsiklis, 2000) and (Sutton et al., 2000b), which we call projection $Q^\pi \to A^\pi$. First, we evaluate how fast these different methods get to a good gradient estimate on analytically tractable problems in Section 3.3.2.1 and compare them to both forms of NAC. Second, we compare them in optimization tasks such as cart-pole balancing, underactuated pole swing-ups, and simple motor primitive

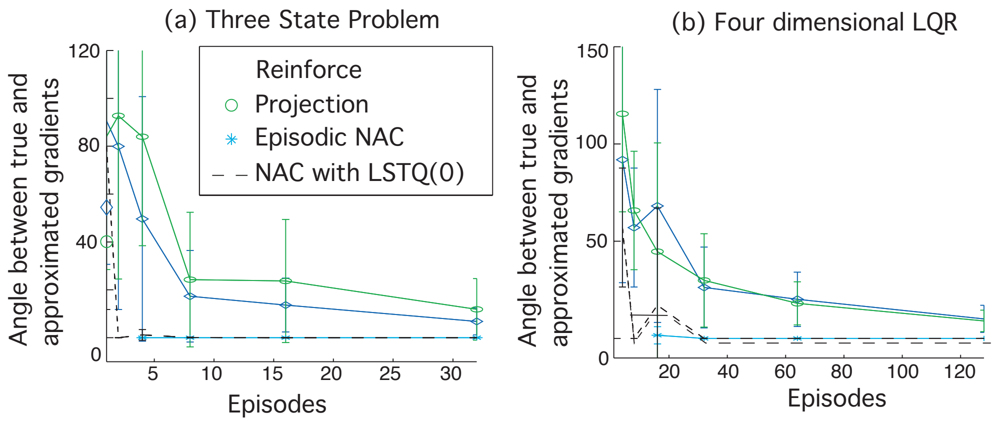
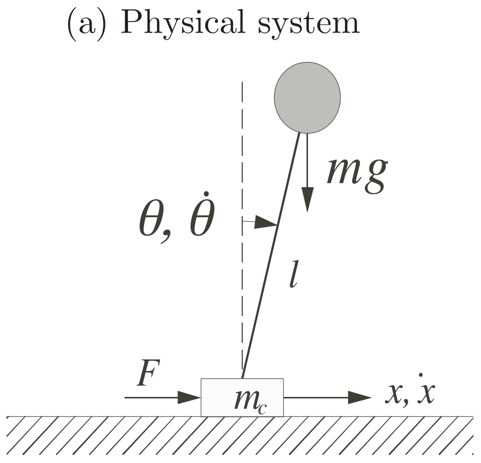


Figure 3.5: This figure shows the angle between the gradient estimate and the true gradient for (a) the three state and three action MDPs with randomly chosen rewards, transition probabilities, and policies, (b) for randomly chosen stable 4 dimensional linear quadratic Gaussian regulation problems.

evaluations and comapre them only with episodic NAC. Furthermore, we apply the combination of episodic NAC and the motor primitive framework to a robotic task on a real robot, i.e., 'hitting a T-ball with a baseball bat'.

3.3.2.1 Gradient Estimation Comparison

Intially, we are interested in the convergence speed of gradient estimators to the true (natural or 'vanilla') policy gradient. We therefore chose to study problems which are analytically tractable, i.e., discrete problems and linear-quadratic Gaussian regulation problems. We randomly generated both discrete systems with Gibbs policies with three states, and four dimensional linear quadratic Gaussian systems with stable linear Gaussian policies as benchmark. Furthermore, we determined the analytical policy gradients. We compared the angle between true and approximated gradients for the reasons that it easy to understand (e.g., a gradient should be within $\pm 90°$ of the true gradient in order to be useful), and because it is more objective than an L_2 norm which would not reflect small but important direction differences. In Figure 3.5, you can see that this method easily out performs both the projection $Q^\pi \to A^\pi$, the separate estimation of Fisher information matrix and gradient suggested in (Kakade, 2002), and GPOMDP (which estimates the 'Vanilla Policy gradient'). However, LSTD-Q(λ) clearly outperforms Episodic Natural Actor-Critic at the price of needing basis functions which do not bias the gradient. It becomes clear from this plot that the convergence of GPOMDP to the true policy gradient has similar convergence properties as the projection of $Q^\pi(\mathbf{x}, \mathbf{u})$ onto the natural gradient. However, the convergence of Natural Actor-Critic with LSTD-Q(0) and episodic Natural Actor-Critic to the true natural policy gradients is at least several order of magnitude faster.



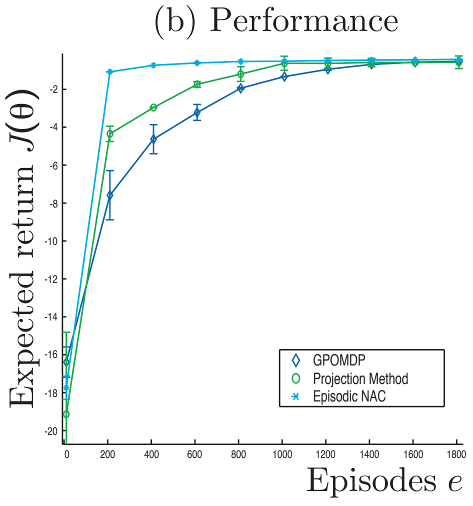


Figure 3.6: This figure shows the physical set-up of a cart-pole balancing in (a), and in (b) the performance of GPOMDP, the projection natural gradient, and the Episodic Natural Actor-Critic in comparison. The latter clearly outperforms the first two - one of the main reasons is the significantly lower variance.

3.3.2.2 Cart-Pole Balancing

Cartpole balancing is a well-known benchmark for reinforcement learning. We assume the cart as shown in Figure 3.6 (a) is given by the dynamics equations of

$$ml\ddot{x}\cos\theta + ml^2\ddot{\theta} - mgl\sin\theta = 0, \tag{3.105}$$

$$(m + m_c)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F, \tag{3.106}$$

with $l = 0.75$m, $m = 0.15$kg, $g = 9.81\text{m/s}^2$ and $m_c = 1.0$kg. The resulting state is given by $\mathbf{x} = [x, \dot{x}, \theta, \dot{\theta}]^T$, and the action $\mathbf{u} = F$. The system is treated as if it was sampled at a rate of $h = 60$Hz, and the reward is given by $r(\mathbf{x}, \mathbf{u}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u}$ with $\mathbf{Q} = \text{diag}(1.25, 1, 12, 0.25)$, $\mathbf{R} = 0.01$. We chose a linear Gaussian policy given by

$$\pi(\mathbf{u}|\mathbf{x}) = \mathcal{N}(\mathbf{u}|\mathbf{k}^T\mathbf{x}, 1/(1 + \exp(-\xi))), \tag{3.107}$$

with parameters $\boldsymbol{\theta}^T = [\mathbf{k}^T, \xi]$. While this can also be treated with LSTD-Q(λ), see (Peters, Vijaykumar, & Schaal, 2003), we will focus on comparing it with GPOMDP and the projection suggested in (Konda & Tsitsiklis, 2000), and in (Kakade, 2002). The results can be seen in Figure 3.6 which makes clear that episodic natural actor-critic clearly outperforms both other methods.

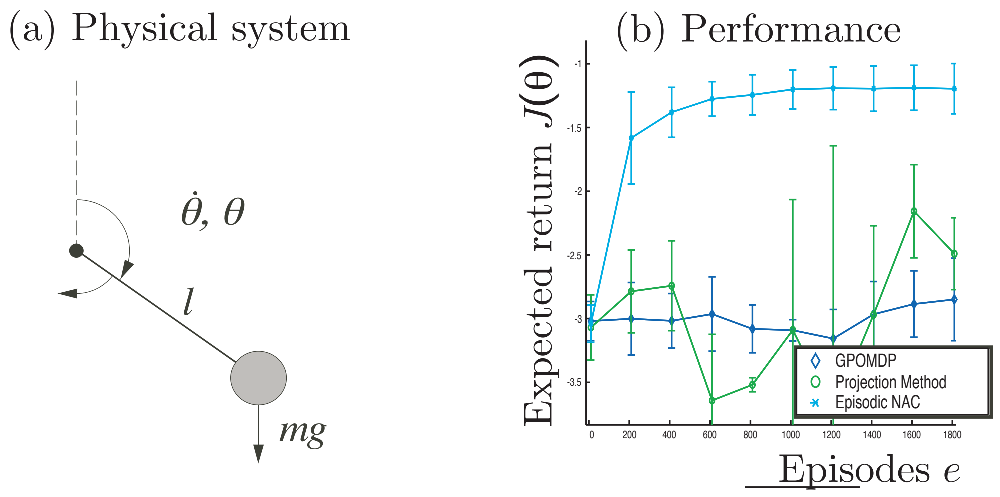


Figure 3.7: This figure shows the physical set-up of a pole swing-up in (a), and in (b) the performance of GPOMDP, the projection natural gradient, and the Episodic Natural Actor-Critic in comparison. The latter clearly outperforms the first two.

3.3.2.3 Underactuated Pole Swing-ups

The dynamics equations of an underactuated pole swing-up as shown in Figure 3.7 (a) can be give by

$$ml^2\ddot{\theta} = -\mu\dot{\theta} + mgl\sin\theta + \tau, \tag{3.108}$$

for $\theta \in [-\pi, +\pi]$. Furthermore, we have $m = 1\text{kg}$, $l = 1\text{m}$, $g = 9.81\text{m/s}^2$, $\mu = 0.05\text{Ns/m}$, and$\tau_{\max} = 5\text{Nm}$. We define the state to be $\mathbf{x} = [\theta, \dot{\theta}]^T$, and the action $\mathbf{u} = \max\{\min\{\tau, \tau_{\max}\} - \tau_{\max}\}$). The reward is given by $r(\mathbf{x}, \mathbf{u}) = (x_1/\pi)^2 - 0.01(2/\pi)^2 \log\cos\left(\pi\tau/(2\tau_{\max})\right)$.

As a policy, we chose novel approach to stochastic policies, i.e., a mixture of stochastic policies. In this approach, we have a number of local policies $\pi_i(\mathbf{u}|\mathbf{x}, i)$, which are selected via the gating policies $\pi_i(i|\mathbf{x})$. This gives us the following structure

$$\pi(\mathbf{u}|\mathbf{x}) = \sum_{i=1}^{N} \pi_i(i|\mathbf{x})\pi_i(\mathbf{u}|\mathbf{x}, i). \tag{3.109}$$

In this particular example, we choose $\pi_i(i|\mathbf{x}) = \mathcal{N}(x_1|c_i, \hat{\sigma}_i^2) / \left(\sum_{i=1}^{N} \mathcal{N}(x_1|c_i, \hat{\sigma}_i^2)\right)$, and $\pi_i(\mathbf{u}|\mathbf{x}, i) = \mathcal{N}(\mathbf{u}|\mathbf{k_i}^T[\mathbf{x}, 1]^T, \sigma_i^2)$. In practice, this policy is used as follows: first draw a local policy $i \sim \pi_i(i|\mathbf{x})$, and then draw an action $\mathbf{u} \sim \pi_i(\mathbf{u}|\mathbf{x}, i)$ from the local policy. This mixture of stochastic policies combined with Natural Actor-Critic can also be seen as an hierarchical approach to reinforcement learning. For the swing-up we use three

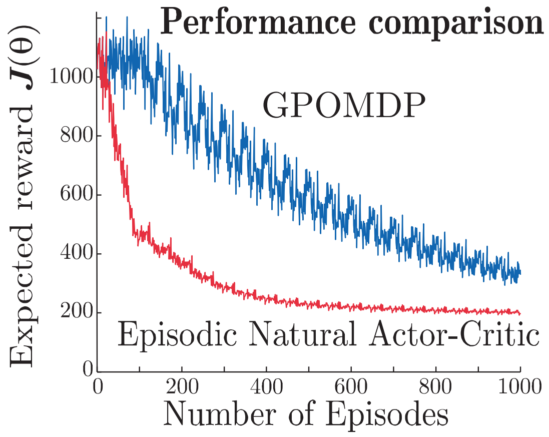


Figure 3.8: This figure shows the convergence of both GPOMDP and the episodic Natural Actor-Critic in a one DOF motor primitive planning tasks.

models ($N = 3$) with centers $\mathbf{c} = [-\pi, 0, +\pi]^T$. All local controllers are initialized to be unstable, and the local exploration is set high in order to make local minima less likely to occur.

The results of the usage of this policy for the swing-up are somehow surprising. As can be seen in Figure 3.7(b), both GPOMDP and the natural gradient obtain from the projection fail do not always converge to a swing-up. Despite that they occasionally converge to an swing-up, they very often remain unstable and yield limit cycles. It is of course possible that further performance enhancements can improve these algorithms. Episodic Natural Actor-Critic yields a surprising high pace in learning, needed little tuning, and is hardly affected by the perturbations on the initial parameters.

3.3.2.4 Motor Primitive Learning for Baseball

While the previous examples demonstrated the feasibility and performance of the episodic Natural Actor Critic in a classical example of motor control, this section will turn towards optimizing nonlinear dynamic motor primitives for robotics. In (Ijspeert et al., 2001, 2003), a novel form of representing movement plans ($\mathbf{q}_d, \dot{\mathbf{q}}_d$) for the degrees of freedom (DOF) robot systems was suggested in terms of the time evolution of the nonlinear dynamical systems $\dot{q}_{d,k} = h(q_{d,k}, \mathbf{z}_k, g_k, \tau, \theta_k)$ where $(q_{d,k}, \dot{q}_{d,k})$ denote the desired position and velocity of a joint, z_k the internal state of the dynamic system, g_k the goal (or point attractor) state of each DOF, τ the movement duration shared by all DOFs, and θ_k the open parameters of the function h. The original work in (Ijspeert et al., 2001, 2003) demonstrated how the parameters θ_k can be learned to match a template trajectory by means of supervised learning – this scenario is, for instance, useful

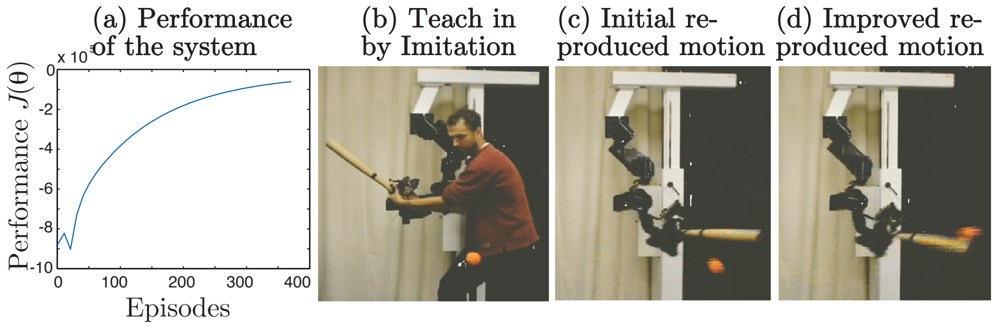


Figure 3.9: This figure shows (a) the performance of a baseball swing task when using the motor primitives for learning. In (b), you can see how it is initialized by teach-in or imitation learning, in (c) you see a failing reproduction of the motor behavior, and in (d) you see the system several hundred episodes further exhibiting a nicely learned batting.

as the first step of an imitation learning system. Here we will add the ability of self-improvement of the movement primitives by means of reinforcement learning, which is the crucial second step in imitation learning. The system is a point-to-point movement, i.e., this task is rather well suited for episodic Natural Actor-Critic. In Figure 3.8, we show a comparison with GPOMDP for simple, single DOF task with a reward of $r_k(x_{0:N}, u0 : N) = \sum_{i=0}^{N} c_1 \dot{q}_{d,k,i}^2 + c_2(qd; k; N - g_k)$; where $c_1 = 1$, $c_2 = 1000$, and g_k is chose appropriately. We also evaluated the same setup in a challenging robot task, i.e., the planning of these motor primitives for a seven DOF robot task. The task of the robot is to hit the ball properly so that it flies as far as possible. Initially, it is taught in by supervised learning as can be seen in Figure 3.9 (b); however, it fails to reproduce the behavior as shown in (c); subsequently, we improve the performance using the episodic Natural Actor-Critic which yields the performance shown in (a) and the behavior in (c).

3.4 Probabilistic Policy Search

In supervised learning and function approximation, research which employs gradient methods is slowly fading out as these methods have a very complicated open parameter, i.e., the learning rate. However, it has become apparant that natural gradients in supervised learning are highly related to parameterized probabilistic methods such as the EM algorithm. In this spirit, we review the sparse literature on probabilitic methods for reinforcement learning and show that it is also related to natural policy gradients.

3.4.1 Probability Matching

In the early 1990s, two papers in the the idea of probability matching have shown probably the only attempt of achieving probabilistic methods in reinforcement learning

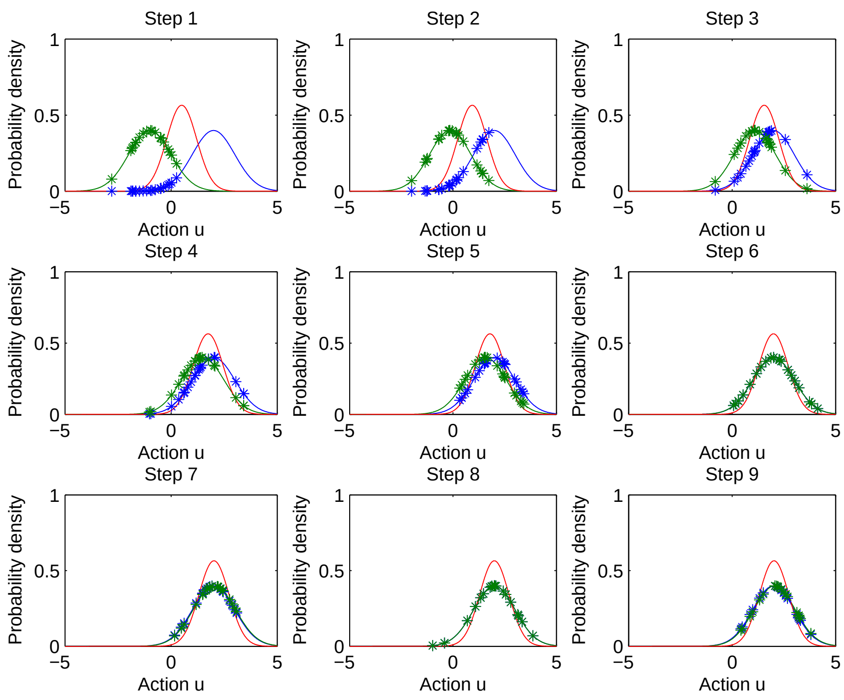


Figure 3.10: This figure visualizes the probability matching principles. The actions are sampled from an initial policy $\pi_\theta(u) = \mathcal{N}(u|\theta, 1)$ shown in green, and the actor receives the rewards $r(u) = c\exp(-0.5(u-2)^2)$ shown in blue. By adjusting the single parameter, the mean θ, the new policy will attempt to match $\pi_\theta(u)r(u)$ as well as possible. After just nine iterations, the policy matches the optimal policy in this parameterized policy class perfectly.

(Dayan & Hinton, 1997; Sabes & Jordan, 1996). This probability matching principle states the following:

> The expected return of a single decision problem can be maximized by sequentially picking a new policy $\pi_{n+1}(u)$ over actions u which matches the reward-weighted action probabilities $\pi_n(u)r(u)$ of the previous solution.

Both papers give basic evaluation in a single-stage decision context with no temporal credit assignment, and their papers have been largely overlooked. The reason for this appears to be not only the pessimism of the authors but also the fact the 1990s were dominated by value function based policy search methods (Sutton et al., 2000b).

Let us illustrate the resulting behavior in the single-stage decision case as they were described originally. In the first step, we would sample a collection of actions $\{u_1, \ldots, u_n\}$ from an initial policy $\pi_0(u)$, for example with a Gaussian policy, and obtain a variety of rewards $r(u_1), \ldots, r(u_n)$, as shown in Figure 3.10. Then we attempt to match the next policy to the reward weighted one $\pi_1(u)r(u)$. We will match the policy to the probability weighted reward as well as possible in our policy class. In the next step, we sample from the new policy, and obtain a new set of rewards - and again match the reward-weighted probabilities. After 9 steps we converge to a limit point and stay there stably as shown in Figure 3.10.

3.4.2 Convergence of Probability Matching

At this point, the reader will certainly have three questions on his mind: how do I decided that a policy matches another? How does it relate to the cost function

$$J(\theta) = \int_{\mathbb{U}} \pi_\theta(u) r(u) du? \tag{3.110}$$

Why should this always improve the policy? The most common way to compare probability distributions is the Kullback-Leibler divergence, i.e., we would try to find $\pi_{\theta'}(u)$ so that $D(\pi_\theta(u)r(u)||\pi_{\theta'}(u))$ is minimal. Let us define a function $Q(\theta,\theta') = \int_{\mathbb{U}} \pi_\theta(u) r(u) \log \pi_{\theta'}(u) du$. Using this function, we can show that $D(\pi_\theta(u)r(u)||\pi_{\theta'}(u)) = Q(\theta,\theta) - Q(\theta,\theta') + \int_{\mathbb{U}} \pi_\theta(u)r(u)\log r(u) du$, and that minimizing $D(\pi_\theta(u)r(u)||\pi_{\theta'}(u))$ is equivalent to maximizing the expected return. We use Dayan & Hintons (Dayan & Hinton, 1997) proof that

$$\log \frac{J(\theta')}{J(\theta)} = \log \int_{\mathbb{U}} \pi_{\theta'}(u) \frac{r(u)}{J(\theta)} du = \log \int_{\mathbb{U}} \frac{\pi_\theta(u) r(u)}{J(\theta)} \frac{\pi_{\theta'}(u)}{\pi_\theta(u)} du, \tag{3.111}$$

$$\geq \int_{\mathbb{U}} \frac{\pi_\theta(u) r(u)}{J(\theta)} \log \frac{\pi_{\theta'}(u)}{\pi_\theta(u)} du = \frac{1}{J(\theta)} \left(Q(\theta,\theta') - Q(\theta,\theta) \right), \tag{3.112}$$

using the Jensen-inequality. It is clear that for any θ' for which $Q(\theta,\theta') \geq Q(\theta,\theta)$ is true, we also have $J(\theta') \geq J(\theta)$. As minimizing $D(\pi_\theta(u)r(u)||\pi_{\theta'}(u))$ is equivalent to maximizing $Q(\theta,\theta')$, we can guarantee that we always improve the policy. This implies that an algorithm which first computes the expectation $Q(\theta,\theta') = \int_{\mathbb{U}} \pi_\theta(u) r(u) \log \pi_{\theta'}(u) du$, and then maximizes $Q(\theta,\theta')$ with respect to θ' is guaranteed to converge, i.e., $\theta_{n+1} = \operatorname{argmax}_{\theta'} Q(\theta_n, \theta')$ and has no open parameters like learning rates. Such algorithms are called Expectation-Maximization algorithms. Similarly, we can guarantee the improvement in incremental methods update with the gradient of $Q(\theta,\theta')$ with respect to θ'. However, it is clear that this method assumes that we are not affected by the sampling dynamics – this assumption does not hold for multi-step learning problems.

3.4.3 Relation to Previously Discussed Methods

While we cannot solve the problems resulting from sampling dynamics yet and therefore cannot show more efficient algorithms, we can make clear that probability matching methods when applied to histories are highly related to natural policy gradient methods. Let us demonstrate this by using the cost function

$$J_{\text{PM}}(\boldsymbol{\theta}, \boldsymbol{\theta}_n) = -D\left(r(\boldsymbol{\xi})p\left(\boldsymbol{\xi}|\boldsymbol{\theta}_n\right)||p\left(\boldsymbol{\xi}|\boldsymbol{\theta}\right)\right) = -\int_{\mathbb{T}} r(\boldsymbol{\xi})p\left(\boldsymbol{\xi}|\boldsymbol{\theta}_n\right)\log\frac{r(\boldsymbol{\xi})p\left(\boldsymbol{\xi}|\boldsymbol{\theta}_n\right)}{p\left(\boldsymbol{\xi}|\boldsymbol{\theta}\right)}d\boldsymbol{\xi}, \tag{3.113}$$

which we intend to maximize. This yields the first derivative of $J_{\text{PM}}(\boldsymbol{\theta})$ is equal in $\boldsymbol{\theta}_0$, i.e.,

$$\nabla_{\boldsymbol{\theta}} J_{\text{PM}}(\boldsymbol{\theta}_n, \boldsymbol{\theta}_n) = \left.\int_{\mathbb{T}} r(\boldsymbol{\xi})p\left(\boldsymbol{\xi}|\boldsymbol{\theta}_n\right)\nabla_{\boldsymbol{\theta}}\log p(\boldsymbol{\xi}|\boldsymbol{\theta})d\boldsymbol{\xi}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_n} = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_n). \tag{3.114}$$

That proves that a probability matching gradient approach is a generalization of a policy gradient approach, equivalent to REINFORCE. The second derivative is more interesting as it yields

$$\nabla^2_{\boldsymbol{\theta}} J_{\text{PM}}(\boldsymbol{\theta}_n, \boldsymbol{\theta}_n) = \left.\int_{\mathbb{T}} r(\boldsymbol{\xi})p\left(\boldsymbol{\xi}|\boldsymbol{\theta}_n\right)\nabla^2_{\boldsymbol{\theta}}\log p(\boldsymbol{\xi}|\boldsymbol{\theta})d\boldsymbol{\xi}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_n}, \tag{3.115}$$

$$\neq \left.\int_{\mathbb{T}} r(\boldsymbol{\xi})\nabla^2_{\boldsymbol{\theta}} p(\boldsymbol{\xi}|\boldsymbol{\theta})d\boldsymbol{\xi}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_n} = \nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_n). \tag{3.116}$$

This result is very interesting as it shows that the curvature between both function differs.

$$\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_n) = \int_{\mathbb{T}} r(\boldsymbol{\xi})p(\boldsymbol{\xi}|\boldsymbol{\theta}_n)\nabla^2_{\boldsymbol{\theta}}\log p(\boldsymbol{\xi}|\boldsymbol{\theta}_n)d\boldsymbol{\xi} + \int_{\mathbb{T}} r(\boldsymbol{\xi})p(\boldsymbol{\xi}|\boldsymbol{\theta}_n)\nabla_{\boldsymbol{\theta}}\log p(\boldsymbol{\xi}|\boldsymbol{\theta}_n)\nabla_{\boldsymbol{\theta}}\log p(\boldsymbol{\xi}|\boldsymbol{\theta}_n)^T d\boldsymbol{\xi}, \tag{3.117}$$

$$= \nabla^2_{\boldsymbol{\theta}} J_{\text{PM}}(\boldsymbol{\theta}_n, \boldsymbol{\theta}_n) + \int_{\mathbb{T}} r(\boldsymbol{\xi})p(\boldsymbol{\xi}|\boldsymbol{\theta}_n)\nabla_{\boldsymbol{\theta}}\log p(\boldsymbol{\xi}|\boldsymbol{\theta}_n)\nabla_{\boldsymbol{\theta}}\log p(\boldsymbol{\xi}|\boldsymbol{\theta}_n)^T d\boldsymbol{\xi}. \tag{3.118}$$

However, it is not clear what the second term represents[14].

Particularly interesting is the case where we take a long, greedy step along the gradient, i.e., we do $\boldsymbol{\theta}_{n+1} = \operatorname{argmax}_{\boldsymbol{\theta}'} Q\left(\boldsymbol{\theta}_n, \boldsymbol{\theta}'\right)$. In this case, we have $p\left(\boldsymbol{\xi}|\boldsymbol{\theta}_{n+1}\right) \approx r(\boldsymbol{\xi})p\left(\boldsymbol{\xi}|\boldsymbol{\theta}_n\right)$. This implies that in this case, we have

$$\nabla^2_{\boldsymbol{\theta}} J_{\text{PM}}(\boldsymbol{\theta}_{n+1}, \boldsymbol{\theta}_n) = \left.\int_{\mathbb{T}} r(\boldsymbol{\xi})p\left(\boldsymbol{\xi}|\boldsymbol{\theta}_n\right)\nabla^2_{\boldsymbol{\theta}}\log p(\boldsymbol{\xi}|\boldsymbol{\theta})d\boldsymbol{\xi}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{n+1}}, \tag{3.119}$$

$$\approx \left.\int_{\mathbb{T}} p\left(\boldsymbol{\xi}|\boldsymbol{\theta}_{n+1}\right)\nabla^2_{\boldsymbol{\theta}}\log p(\boldsymbol{\xi}|\boldsymbol{\theta})d\boldsymbol{\xi}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{n+1}} = -F(\boldsymbol{\theta}_{n+1}). \tag{3.120}$$

[14]Similar evaluations for the distance metric $D\left(p\left(\omega|\theta\right)||r(\tau)p\left(\tau|\theta_0\right)\right)$ yields that there we have $\nabla^2_{\theta} J_{\text{PM}}(\theta_n, \theta_n) = \nabla^2_{\theta} J(\theta_n) + F(\theta_n)$. This result is easier to understand, however, does not apply as the Kullback-Leibler divergence is not symmetric.

Therefore, close to the optimal solution, natural gradients should perform similarly well to the a second order method in probability matching. Note that the Hessian $\boldsymbol{\nabla}^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_n)$ is very small at this point while the Fisher information matrix can be arbitrarily large. This also means that in the limit point when both sampling and target policy are close to each other, i.e., $p\left(\boldsymbol{\xi}|\boldsymbol{\theta}_n\right) \approx r(\boldsymbol{\xi})p\left(\boldsymbol{\xi}|\boldsymbol{\theta}_n\right)$, we will have the second order taylor-expension

$$J_{\mathrm{PM}}(\boldsymbol{\theta}_n + \Delta\boldsymbol{\theta}, \boldsymbol{\theta}_n) = J_{\mathrm{PM}}(\boldsymbol{\theta}_n, \boldsymbol{\theta}_n) + \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_n)\Delta\boldsymbol{\theta} - \Delta\boldsymbol{\theta}^T \mathbf{F}(\boldsymbol{\theta}_n)\Delta\boldsymbol{\theta}. \tag{3.121}$$

This gives us the optimal update step $\Delta\boldsymbol{\theta} = \mathbf{F}^{-1}(\boldsymbol{\theta}_n)\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_n)$. With other words, the direction of the update approximates the natural policy gradient update.

Chapter 4

Executing Motor Skills through a Generalization of Gauss' Principle

4.1 Introduction

Despite the progress in robotics over the last decades, only a few general building principles for designing robot controllers have been obtained. To date, robot controllers are often derived from insights such as the reduction of the controlled system onto a linear system by linearization or by inversion of the dynamics of the robot (Yoshikawa, 1990; De Wit et al., 1996). While this approach is viable for many problems, it is in a sense limiting because it ignores potentially useful properties of the inherent nonlinearities. Only few statements can be made about the quality of such controllers underlying cost functions which sometimes cannot even be obtained. General optimal control techniques on the other hand are often not applicable as a closed-form solution usually does not exist and numerical solutions for high-dimensional systems are often prohibitively expensive in terms of computations due to the 'Curse of Dimensionality' (Bellman & Kalaba, 1965), (Bryson, 1981).

Recently, a novel way of thinking about the control of mechanical systems was suggested in (Udwadia, 2003) inspired by results from analytical dynamics with constrained motion. The major insight in (Udwadia, 2003) is that tracking control can be reformulated in terms of constraints, which in turn allows the application of a generalization of Gauss' principle of least constraint[1] (Udwadia & Kalaba, 1996) in order to derive a controller. As it is outlined already in (Udwadia, 2003), this insight leads to a specialized optimal control framework for controlled mechanical systems. While it is not applicable to non-mechanical control problems with arbitrary cost functions, it yields an important class of optimal controllers, i.e., the class where the problem requires task achievement under minimal squared motor commands with respect to a specified metric. In this chapter, we develop this line of thinking a step further and show that it

[1]Gauss' principle of least constraint (Udwadia & Kalaba, 1996) is a general axiom on the mechanics of constrained motions. It states that if a mechanical system is constrained by another mechanical structure the resulting acceleration $\ddot{\mathbf{x}}$ of the system will be such that it minimizes $(\ddot{\mathbf{x}} - \mathbf{M}^{-1}\mathbf{F})^T\mathbf{M}^{-1}(\ddot{\mathbf{x}} - \mathbf{M}^{-1}\mathbf{F})$ while fulfilling the constraint.

can be used as a general way of solving robotic control problems which unifies many approaches to robot control found in the literature to date. We can demonstrate stability of the controller in task space if the system can be modeled with sufficient precision and the chosen metric is appropriate. For assuring stability in the joint space further considerations may apply. To demonstrate the feasibility of our framework, we evaluate a few derived controllers on a robot arm with a simple end-effector tracking task.

This chapter is organized as follows: firstly, a novel optimal control framework based on (Udwadia, 2003) is presented and analyzed. Secondly, we discuss different robot control problems in this framework including joint and task space tracking, force and hybrid control. We show how both established and novel controllers can be derived in a unified way. Finally, we evaluate some of these controllers on a Sarcos Master robot arm.

4.2 A Novel Methodology for the Execution of Motor Skills

A variety of robot control problems can be motivated by the desire to achieve a task perfectly while minimizing the squared motor commands. In this section, we will show how the robot dynamics and the control problem can be brought into a general form which will then allow us to compute the optimal control with respect to a desired metric. We will augment this framework so that we can assure stability both in the joint space of the robot as well as in the task space of the problem.

4.2.1 Formulating Robot Control Problems

In order to formulate our framework, we will introduce the specifics of the assumed underlying robot model and show how a task can be specified.

Robot Model: We assume the well-known rigid-body dynamics model of manipulator robot arms with n degrees of freedom given by the equation

$$\mathbf{u} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}), \tag{4.1}$$

where $\mathbf{u} \in \mathbb{R}^n$ is the vector of motor commands (i.e., torques or forces), $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^n$ are the vectors of joint position, velocities and acceleration, respectively, $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the mass or inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^n$ denotes centrifugal and Coriolis forces, and $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^n$ denotes gravity (Yoshikawa, 1990; De Wit et al., 1996). At many points we will write the dynamics equations by $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} = \mathbf{u}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$ where $\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}) = -\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{G}(\mathbf{q})$ as specified in (Udwadia & Kalaba, 1996; Udwadia, 2003). We assume that an accurate model of our robot system is available.

Task Description: A task for the robot is assumed to be described in the form of a constraint description, i.e., it is given by a function

$$\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}, t) = 0. \tag{4.2}$$

where $\mathbf{h} \in \mathbb{R}^k$ where the dimensionality is arbitrary. For example, if the robot is supposed to follow a desired trajectory $\mathbf{q}_{\text{des}}(t) \in \mathbb{R}^n$, we could formulate it by $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}, t) = \mathbf{q} - \mathbf{q}_{\text{des}}(t) = 0$; this case is analyzed in detail in Section 4.3.1. We consider only tasks wherein Equation (4.2) can be reformulated as

$$\mathbf{A}(\mathbf{q}, \dot{\mathbf{q}}, t)\ddot{\mathbf{q}} = \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}, t), \tag{4.3}$$

which can be achieved for most tasks by differentiation of Equation (4.2) with respect to time, assuming that h is sufficiently smooth. For example, our previous task, upon differentiation, becomes $\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_{\text{des}}(t)$ so that $\mathbf{A} = \mathbf{I}$ and $\mathbf{b} = \ddot{\mathbf{q}}_{\text{des}}(t)$. An advantage of this task formulation is that non-holomonic constraints can be treated in the same general way. Note that Equation (4.2) is also a general way to represent the motor primitives we have introduced in Chapter 2.

In Section 4.3, we will always give the task description first in the general form in Equation (4.2), and then derive the resulting controller using the form which is the linear in accelerations, given in Equation (4.3).

4.2.2 Optimal Control Framework

Let us assume that we are given a robot model and a constraint description of the task as described in the previous section. In this case, we can describe the desired properties of the framework as follows: first, the task has to be achieved perfectly, i.e., $h(\mathbf{q}, \dot{\mathbf{q}}, t) = 0$, or equivalently, $\mathbf{A}\ddot{\mathbf{q}} = \mathbf{b}$, holds at all times. Second, we intend to minimize the control force with respect to some given metric, i.e., $J(t) = \mathbf{u}^T\mathbf{N}(t)\mathbf{u}$, at *each* instant of time. The solution to this can be derived from a generalization of Gauss' principle as originally suggested in (Udwadia, 2003). We formalize this here in the following theorem.

Theorem 13 *The class of controllers which minimizes*

$$J(t) = \mathbf{u}^T\mathbf{N}(t)\mathbf{u}, \tag{4.4}$$

for a mechanical system $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} = \mathbf{u}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$ *while fulfilling the task constraint*

$$\mathbf{A}\ddot{\mathbf{q}} = \mathbf{b}, \tag{4.5}$$

is given by

$$\mathbf{u} = \mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}\left(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{F}\right), \tag{4.6}$$

where $\mathbf{D}^{+}$ *denotes the pseudo-inverse for a general matrix* $\mathbf{D}$*, and* $\mathbf{D}^{1/2}$ *denotes the symmetric, positive definite matrix for which* $\mathbf{D}^{1/2}\mathbf{D}^{1/2} = \mathbf{D}$.

Proof. By defining $\mathbf{z} = \mathbf{N}^{1/2}\mathbf{u} = \mathbf{N}^{1/2}(\mathbf{M}\ddot{\mathbf{q}} - \mathbf{F})$, we obtain $\ddot{\mathbf{q}} = \mathbf{M}^{-1}\mathbf{N}^{-1/2}(\mathbf{z} + \mathbf{N}^{1/2}\mathbf{F})$. Since the task constraint $\mathbf{A}\ddot{\mathbf{q}} = \mathbf{b}$ has to be fulfilled, we obtain

$$\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\mathbf{z} = \mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{F}. \tag{4.7}$$

The vector $\mathbf{z}$ which minimizes $J(t) = \mathbf{z}^T\mathbf{z}$ while fulfilling Equation (4.7), is given by $\mathbf{z} = (\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2})^{+}(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{F})$, and as the motor command is given by $\mathbf{u} = \mathbf{N}^{-1/2}\mathbf{z}$, the theorem holds. ■

The choice of the metric $\mathbf{N}$ plays a central role, because it determines the type of solution. Often, we require a solution which has a kinematic interpretation; such a solution is usually given by a metric like $\mathbf{N} = \mathbf{M}^{-2}$. In other cases, the control force $\mathbf{u}$ may be required to comply with the principle of virtual displacements by d'Alembert for which the metric $\mathbf{N} = \mathbf{M}^{-1}$ is more appropriate. In Section 4.3, we will see how the choice of $\mathbf{N}$ results in several different controllers.

Note that this framework has been suggested in general in (Udwadia & Kalaba, 1996; Udwadia, 2003), and the special case with a metric $\mathbf{N} = \mathbf{M}^{-1}$ has been presented in (Bruyninckx & Khatib, 2000) with respect to robot control.

4.2.3 Stability Analysis

Up to this point, this framework has been introduced in an idealized fashion neglecting the possibility of imperfect initial conditions and measurement noise. Therefore, we modify this framework slightly and show how we can ensure stability. This modification will be introduced in Section 4.2.3.1. Furthermore, we realize that the case of under-constrained tasks, i.e., tasks where some degrees of freedom of the robot are redundant for the given task, can cause undesired properties or even instability in joint-space; we will treat this problem in Section 4.2.3.2.

4.2.3.1 Stability in Task Space

Up to this point, we have assumed that we always have perfect initial conditions and that we know the robot model perfectly. However, we have to compensate for the fact that we might not be sitting perfectly on the trajectory from the start or that we might get disturbed out of this trajectory. (Udwadia, 2003) suggested that this can be achieved by requiring that the desired task is an attractor, e.g., it could be prescribed as a dynamical system in the form

$$\dot{\mathbf{h}}(\mathbf{q}, \dot{\mathbf{q}}, t) = \mathbf{f}_{\mathbf{h}}(\mathbf{h}, t), \tag{4.8}$$

where $\mathbf{h} = \mathbf{0}$ is a globally asymptotically stable equilibrium point – or a locally asymptotically stable equilibrium point with a sufficiently large region of attraction. Note that $\mathbf{h}$ can be a function of robot variables (as in end-effector trajectory control in Section 4.3.2) but often it suffices to choose it to be state vector (for example for joint-space trajectory control as in Section 4.3.1). In the case of holomonic tasks (such as tracking control for a robot arm), i.e. $h_i(\mathbf{q}, t) = 0$, $i = 1, 2, \ldots, k$ we can make use of a particularly simple form as suggested in (Udwadia, 2003) and turn this task into an attractor

$$\ddot{h}_i + \delta_i \dot{h}_i + \kappa_i h = 0, \tag{4.9}$$

where δ_i and κ_i are chosen appropriately. We will make use of this 'trick' in order to derive several algorithms. Obviously, different attractors with more desirable convergence properties (and/or larger basins of attraction) can be obtained by choosing $\mathbf{f_h}$ appropriately.

If we have a task-space stabilization as discussed in the paragraph above, we can assure that the control law is stable in task space at least in a region near about the desired trajectory. We show this in the following theorem.

Theorem 14 *If we can assure the attractor property of the task $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}, t) = \mathbf{0}$, or equivalently, $\mathbf{A}\ddot{\mathbf{q}} = \mathbf{b}$, and if our robot model is accurate, it is straightforward to show that the controller is stable in task space.*

Proof. When combining the robot dynamics equation with the controller, and after reordering the terms, we obtain

$$\mathbf{AM}^{-1}(\mathbf{M}\ddot{\mathbf{q}} - \mathbf{F}) = \left(\mathbf{AM}^{-1}\mathbf{N}^{-1/2}\right)^{+}(\mathbf{b} - \mathbf{AM}^{-1}\mathbf{F}). \tag{4.10}$$

If we now premultiply the equation with $\mathbf{D} = \mathbf{AM}^{-1}\mathbf{N}^{-1/2}$, and noting that $\mathbf{DD}^{+}\mathbf{D} = \mathbf{D}$, we obtain $\mathbf{A}\ddot{\mathbf{q}} = \mathbf{GG}^{+}\mathbf{b} = \mathbf{b}$. The equality follows because the original trajectory defined by $\mathbf{A}\ddot{\mathbf{q}} = \mathbf{b}$ yields a consistent set of equations. If this is an attractor, we will have perfect task achievement asymptotically. ■

An analysis of the stability properties of the derived controllers when an imperfect robot model is given will be part of future work.

4.2.3.2 Stability in Joint Space

While the stability in task space is fairly well-understood, it is not immediately clear whether the control law is stable in joint-space. It is fairly straightforward to create a counter-example. Example 15, illustrates a situation where a redundant robot arm is stable in task-space while unstable in joint-space.

Example 15 *Let us assume the simplest possible robot, a prismatic robot with two horizontal, parallel links. The mass matrix of this robot is a constant given by $\mathbf{M} = \operatorname{diag}(m_1, 0) + m_2\mathbf{1}$ where $\mathbf{1}$ denotes a matrix having only ones as entries, and the additional forces are $\mathbf{F} = \mathbf{0}$. Let us assume the task is to move the end-effector $x = q_1 + q_2$ along a desired position x_{des}, i.e., the task can be specified by $\mathbf{A} = [1, 1]$, and $b = \ddot{x}_{des} + \delta(\dot{x}_{des} - \dot{x}) + \kappa(x_{des} - x)$ after double differentiation and task stabilization. While this obviously is stable in task-space, the initial condition $q_1(t_0) = x_{des}(t_0) - q_2(t_0)$ would result into both $q_i(t)$'s diverging into opposite directions. The reason for this is obvious: the effort of stabilizing in joint space is not task relevant – any solution stabilizing this problem in joint-space would increase the cost.*

From this example, we see that the general framework does not always suffice but that it has to be modified so that we can incorporate a minimal control which in practice stabilizes the robot without affecting the task achievement. One possibility to stabilize the robot in joint-space is by having a joint-space motor command $\mathbf{u}_1$ as an additional component of the the motor command $\mathbf{u}$, i.e.,

$$\mathbf{u} = \mathbf{u}_1 + \mathbf{u}_2\left(\mathbf{u}_1\right), \tag{4.11}$$

where the first component $\mathbf{u}_1$ denotes an arbitrary joint-space motor command for stabilization, while the second component $\mathbf{u}_2\left(\mathbf{u}_1\right)$ denotes the task-space motor command generated with the previously explained equations. The task-space component depends on the joint-space component as it has to compensate for it. We can show that the fulfillment of the task $\mathbf{A}\ddot{\mathbf{q}} = \mathbf{b}$ by the controller is not affected by the choice of the joint-space control law $\mathbf{u}_1$.

Theorem 16 *For any chosen joint-stabilizing control law $\mathbf{u}_1 = f(\mathbf{q})$, the resulting task space control law $\mathbf{u}_2\left(\mathbf{u}_1\right)$ ensures that the joint-stabilizing control law acts in the null-space of the task.*

Proof. When determining $\mathbf{u}_2$, we consider $\mathbf{u}_1$ to be part of our forces, i.e., we have $\tilde{\mathbf{F}} = \mathbf{F} + \mathbf{u}_1$. We obtain $\mathbf{u}_2 = \mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}\left(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\tilde{\mathbf{F}}\right)$ using Theorem 13. By reordering the complete control law $\mathbf{u} = \mathbf{u}_1 + \mathbf{u}_2\left(\mathbf{u}_1\right)$, we obtain

$$\begin{aligned}
\mathbf{u} &= \mathbf{u}_1 + \mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}\left(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}(\mathbf{F} + \mathbf{u}_1)\right), \\
&= \mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{F}) + (\mathbf{I} - \mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}\mathbf{A}\mathbf{M}^{-1})\mathbf{u}_1, \\
&= \mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{F}) \\
&+ \mathbf{N}^{-1/2}[\mathbf{I} - \left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2})]\mathbf{N}^{1/2}\mathbf{u}_1,
\end{aligned}$$

The task space is defined by $\mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}$, and that the matrix $\mathbf{N}^{-1/2}[\mathbf{I} - \left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2})]$ makes sure that the joint-space control law and the task space control law are $\mathbf{N}$-orthogonal. ■

Despite that the task is still achieved, the optimal control problem is affected by the restructuring of our control law. While we originally minimized $J(t) = \mathbf{u}^T\mathbf{N}(t)\mathbf{u}$, we now have a modified cost function

$$\tilde{J}(t) = \mathbf{u}_2^T\mathbf{N}(t)\mathbf{u}_2 = \left(\mathbf{u} - \mathbf{u}_1\right)^T\mathbf{N}(t)\left(\mathbf{u} - \mathbf{u}_1\right), \tag{4.12}$$

which is equivalent to stating that the complete control law $\mathbf{u}$ should be as close to the joint-space control law $\mathbf{u}_1$ as possible under task achievement.

This reformulation can have significant advantages if used appropriately. For example, a variety of applications – such as using the robot as a haptic interface – a compensation of the robot's gravitational, coriolis and centrifugal forces in joint space can be useful. Such a compensation can only be derived when making use of the modified control law. In this case, we set $\mathbf{u}_1 = -\mathbf{F} = \mathbf{C} + \mathbf{G}$, which allows us to obtain

$$\mathbf{u}_2 = \mathbf{N}^{-1/2} \left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+} \mathbf{b}, \tag{4.13}$$

which does not contain these forces, and we would have a complete control law of $\mathbf{u} = \mathbf{C} + \mathbf{G} + \mathbf{N}^{-1/2} \left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+} \mathbf{b}$.

4.3 Robot Control Laws

The previously described framework offers a variety of applications in robotics – we will only be able to give the most important ones in this chapter. Most of these controllers which we will derive are known from the literature but often from very different building principles. In this section, we show how a vast variety of control laws for different situations can be derived in a simple and straightforward way by using the unifying framework that has been developed hereto. We derive control laws for joint-space trajectory control for both fully actuated and overactuated "muscle-like" robot systems from our framework. We also discuss task-space tracking control systems, and show that most well-known inverse kinematics controllers are applications of the same principle. Additionally, we will discuss how the control of constrained manipulators through impedance and hybrid control can be easily handled within our framework.

4.3.1 Joint-Space Trajectory Control

The first control problem we attempt to tackle is joint-space trajectory control. We consider two different situations: (a) We control a fully actuated robot arm in joint-space, and (b) we control an overactuated arm. The case (b) could, for example, have agonist-antagonist muscles as actuators similar to a human arm[2].

4.3.1.1 Fully Actuated Robot

The first case which we consider is the one of a robot arm which is actuated at every degree of freedom. We have the trajectory as constraint with $\mathbf{h}(\mathbf{q}, t) = \mathbf{q}(t) - \mathbf{q}_d(t) = \mathbf{0}$. We turn this constraint into an attractor constraint using the idea in Section 4.2.3.1, yielding

$$(\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_d) + \mathbf{K}_D (\dot{\mathbf{q}} - \dot{\mathbf{q}}_d) + \mathbf{K}_P (\mathbf{q} - \mathbf{q}_d) = 0, \tag{4.14}$$

[2]An open topic of interest is to handle underactuated robot arm control. This will be part of future work.

where $\mathbf{K}_D = (\delta_{i,j})$ are positive-definite damping gains, and $\mathbf{K}_P = (\kappa_{ij})$ are positive-definite proportional gains. We can bring this into the form $\mathbf{A}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}} = \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})$ with

$$\mathbf{A} = \mathbf{I}, \tag{4.15}$$

$$\mathbf{b} = \ddot{\mathbf{q}}_d + \mathbf{K}_D (\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) - \mathbf{K}_P (\mathbf{q}_d - \mathbf{q}). \tag{4.16}$$

In this case, we can use Theorem 13 and derive the controller. Using $(\mathbf{M}^{-1}\mathbf{N}^{-1/2})^+ = \mathbf{N}^{1/2}\mathbf{M}$ as both matrices are of full rank, we obtain

$$\begin{aligned}\mathbf{u} &= \mathbf{u}_1 + \mathbf{N}^{-1/2} \left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^+ \left(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}(\mathbf{F} + \mathbf{u}_1)\right), \\ &= \mathbf{M}^{1/2} \left(\mathbf{M}^{-1/2}\right)^{-1} (\ddot{\mathbf{q}}_d + \mathbf{K}_D (\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_P (\mathbf{q}_d - \mathbf{q}) - \mathbf{M}^{-1} (-\mathbf{C} - \mathbf{G})), \\ &= \mathbf{M}(\ddot{\mathbf{q}}_d + \mathbf{K}_D (\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_P (\mathbf{q}_d - \mathbf{q})) + \mathbf{C} + \mathbf{G}. \end{aligned} \tag{4.17}$$

Note that all joint-space motor commands or virtual forces $\mathbf{u}_1$ always disappear from the control law and that the chosen metric $\mathbf{N}$ is not relevant – the derived solution is unique and general. It turns out that this a well-known control law, i.e., the **Inverse Dynamics Control Law** (Yoshikawa, 1990; De Wit et al., 1996).

4.3.1.2 Overactuated Robots

Overactuated robot arms as they can be found in biological systems are inheritently different from previously discussed robot arms. For instance, these arms are actuated by several linear actuators, e.g., muscles that often act on the system in form of opposing pairs. These interactions of the opposing pairs of muscles can be modeled using the dynamics equations of

$$\mathbf{D}\mathbf{u} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}), \tag{4.18}$$

where $\mathbf{D}$ depends on our type of muscle. In the simplest model for a two degrees of freedom robot it could be given by

$$\mathbf{D} = \begin{bmatrix} -l & +l & 0 & 0 \\ 0 & 0 & -l & +l \end{bmatrix}. \tag{4.19}$$

We can bring this equation into the standard form by multiplying it with $\mathbf{D}^+$, which results in a modified system where $\tilde{\mathbf{M}}(\mathbf{q}) = \mathbf{D}^+\mathbf{M}(\mathbf{q})$, and $\tilde{\mathbf{F}}(\mathbf{q}, \dot{\mathbf{q}}) = -\mathbf{D}^+\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{D}^+\mathbf{G}(\mathbf{q})$. If we have expressed the trajectory like in previous examples, and we obtain the following controller

$$\mathbf{u} = \tilde{\mathbf{M}}^{1/2} \left(\mathbf{A}\tilde{\mathbf{M}}^{-1/2}\right)^+ \left(\mathbf{b} - \mathbf{A}\tilde{\mathbf{M}}^{-1}\tilde{\mathbf{F}}\right), \tag{4.20}$$

$$= \mathbf{D}^+\mathbf{M}(\ddot{\mathbf{q}}_d + \mathbf{K}_D (\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) - \mathbf{K}_P (\mathbf{q}_d - \mathbf{q})) + \mathbf{D}^+ (\mathbf{C} + \mathbf{G}). \tag{4.21}$$

While immidiately intuitive, it is somehow surprising that this particular controller should fall out of the presented framework. Due to a lack of hardware and realistic simulators, we cannot evaluate this approach within the scope of this chapter.

4.3.2 End-effector Trajectory Control

While joint-space control of a trajectory $\mathbf{q}(t)$ is straightforward and the presented methodology appears to simply repeat earlier results from the literature, the same cannot be said about end-effector control where the position $\mathbf{x}(t)$ of the end-effector is moved along some given trajectory. This problem is generically more difficult as the choice of the metric $\mathbf{N}$ determines the type of the solution and as the joint-space of the robot often has redundant degrees of freedom resulting in problems as already presented in Example 15. In the following context, we will show how to derive different approaches to end-effector control from the presented framework; this yields both established as well as novel control laws.

The task description is given by the end-effector trajectory as constraint with $\mathbf{h}(\mathbf{q}, t) = \mathbf{f}(\mathbf{q}(t)) - \mathbf{x}_d(t) = \mathbf{x}(t) - \mathbf{x}_d(t) = \mathbf{0}$, where $\mathbf{x} = \mathbf{f}(\mathbf{q})$ denotes the forward kinematics. We turn this constraint into an attractor constraint using the idea in Section 4.2.3.1, yielding

$$(\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_d) + \mathbf{K}_D(\dot{\mathbf{x}} - \dot{\mathbf{x}}_d) + \mathbf{K}_P(\mathbf{x} - \mathbf{x}_d) = 0, \tag{4.22}$$

where $\mathbf{K}_D = (\delta_{i,j})$ are positive-definite damping gains, and $\mathbf{K}_P = (\kappa_{ij})$ are positive-definite proportional gains. We make use of the differential forward kinematics, i.e.,

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \tag{4.23}$$

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}. \tag{4.24}$$

These allow us to formulate the problem in form of constraints, i.e., we intend to fulfill

$$\ddot{\mathbf{x}}_d + \mathbf{K}_D(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \mathbf{K}_P(\mathbf{x}_d - \mathbf{x}) = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}, \tag{4.25}$$

and we can bring this into the form $\mathbf{A}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}} = \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})$ with

$$\mathbf{A}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}, \tag{4.26}$$

$$\mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) = \ddot{\mathbf{x}}_d + \mathbf{K}_D(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \mathbf{K}_P(\mathbf{x}_d - \mathbf{x}) - \dot{\mathbf{J}}\dot{\mathbf{q}}. \tag{4.27}$$

These equations determine our task constraints. However, the resulting controller depends on the chosen metric and joint-space control law; it is not a unique, general solution as for joint-space control.

4.3.2.1 Separation of Kinematics and Dynamics

The choice of the metric $\mathbf{N}$ determines the type of the task. A metric of particular importance is $\mathbf{N} = \mathbf{M}^{-2}$ as this metric allows the decoupling of kinematics and dynamics as we will see in this section. Using this metric in Theorem 13, we obtain a control law

$$\begin{aligned}\mathbf{u} &= \mathbf{u}_1 + \mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}\left(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}(\mathbf{F} + \mathbf{u}_1)\right),\\ &= \mathbf{M}\mathbf{J}^{+}(\ddot{\mathbf{x}}_d + \mathbf{K}_D(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \mathbf{K}_P(\mathbf{x}_d - \mathbf{x}) - \dot{\mathbf{J}}\dot{\mathbf{q}}) - \mathbf{M}\mathbf{J}^{+}\mathbf{J}\mathbf{M}^{-1}\mathbf{F}\\ &+ \mathbf{M}(\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\mathbf{M}^{-1}\mathbf{u}_1.\end{aligned}$$

If we choose the joint-space control law $\mathbf{u}_1 = \mathbf{u}_0 - \mathbf{F}$, we obtain the control law

$$\begin{aligned}\mathbf{u} &= \mathbf{M}\mathbf{J}^{+}(\ddot{\mathbf{x}}_d + \mathbf{K}_D(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \mathbf{K}_P(\mathbf{x}_d - \mathbf{x}) - \dot{\mathbf{J}}\dot{\mathbf{q}}) + \mathbf{C} + \mathbf{G}\\ &+ \mathbf{M}(\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\mathbf{M}^{-1}\mathbf{u}_0.\end{aligned} \tag{4.28}$$

This control law is the combination of a **resolved-acceleration kinematic controller** (Yoshikawa, 1990; Hsu, Hauser, & Sastry, 1989) with a model-based controller and an additional null-space term. Similar controllers have been introduced in (Park, Chung, & Youm, 2002, 1995; Chung, Chung, & Y.Youm, 1993; K.C.Suh & Hollerbach, 1987). The null-space term can be eliminated by setting $\mathbf{u}_0 = \mathbf{0}$; however, this can result in instabilities if there are redundant degrees of freedom. This controller will be evaluated in Section 4.4.

4.3.2.2 Dynamically Consistent Decoupling

As noted earlier, another important metric is $\mathbf{N} = \mathbf{M}^{-1}$ as it is consistent with the principle of d'Alembert, i.e., it is dynamically consistent and therefore the resulting control force can be re-interpreted as mechanical structures (e.g., springs and dampers) attached to the end-effector. Again, we apply Theorem 13, and by defining $\tilde{\mathbf{F}} = \mathbf{F} + \mathbf{u}_1$ obtain the control law

$$\begin{aligned}\mathbf{u} &= \mathbf{u}_1 + \mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}\left(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\tilde{\mathbf{F}}\right),\\ &= \mathbf{u}_1 + \mathbf{M}^{1/2}\left(\mathbf{J}\mathbf{M}^{-1/2}\right)^{T}\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T}\right)^{-1}\left(\mathbf{b} - \mathbf{J}\mathbf{M}^{-1}\tilde{\mathbf{F}}\right),\\ &= \mathbf{u}_1 + \mathbf{J}^{T}\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T}\right)^{-1}\left(\mathbf{b} - \mathbf{J}\mathbf{M}^{-1}\tilde{\mathbf{F}}\right),\\ &= \mathbf{J}^{T}\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T}\right)^{-1}(\ddot{\mathbf{x}}_d + \mathbf{K}_D(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \mathbf{K}_P(\mathbf{x}_d - \mathbf{x}) - \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{J}\mathbf{M}^{-1}(\mathbf{C} + \mathbf{G}))\\ &+ \mathbf{M}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{J}^{T}\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T}\right)^{-1}\mathbf{J})\mathbf{M}^{-1}\mathbf{u}_1.\end{aligned}$$

It turns out that this is another well-known control law suggest in (Khatib, 1987) with an additional null-space term. This control-law is used in (Udwadia, 2003) and is especially interesting as it has a clear physical interpretation (Bruyninckx & Khatib, 2000; Udwadia & Kalaba, 1996; Udwadia, 2003): the metric used is consistent with principle of virtual work of d'Alembert. Similarly as before we can compensate for coriolis, centrifugal and gravitational forces in joint-space, i.e., setting $\mathbf{u}_1 = \mathbf{C}+\mathbf{G}+\mathbf{u}_0$. This yields a control law of

$$\begin{aligned}\mathbf{u} = {} & \mathbf{J}^T \left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)^{-1} (\ddot{\mathbf{x}}_d + \mathbf{K}_D (\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \mathbf{K}_P (\mathbf{x}_d - \mathbf{x}) - \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}) + \mathbf{C} + \mathbf{G} \\ & + \mathbf{M}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{J}^T \left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)^{-1} \mathbf{J})\mathbf{M}^{-1}\mathbf{u}_0.\end{aligned} \tag{4.29}$$

The compensation of the forces in joint-space is often desirable for this metric in order to have full control over the resolution of the redundancy as the gravity compensation in task space often results into strange postures.

4.3.2.3 Further Metrics

Using the identity matrix as metric, i.e., $\mathbf{N} = \mathbf{I}$, punishes the squared motor command without reweighting. This metric could be of interest as it distributes the "load" created by the task evenly on the actuators. This metric results in a control law

$$\begin{aligned}\mathbf{u} = {} & \left(\mathbf{J}\mathbf{M}^{-1}\right)^{+} (\ddot{\mathbf{x}}_d + \mathbf{K}_D (\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \mathbf{K}_P (\mathbf{x}_d - \mathbf{x}) - \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{J}\mathbf{M}^{-1} (\mathbf{C} + \mathbf{G})) \\ & + (\mathbf{I} - \left(\mathbf{J}\mathbf{M}^{-1}\right)^{+} \mathbf{J}\mathbf{M}^{-1})\mathbf{u}_1.\end{aligned} \tag{4.30}$$

To our knowledge, this controller has not been presented in the literature.

Another, fairly practical idea would be to weight the different joints depending on the maximal torques $\tau_{\max,i}$ of each joint resulting in a metric $\mathbf{N} = \operatorname{diag}(\tau_{\max,1}^{-1}, \ldots, \tau_{\max,n}^{-1})$.

4.3.3 Controlling Constrained Manipulators: Impedance & Hybrid Control

Contact with outside objects fundamentally alters the robot's dynamics, i.e., a generalized contact force $\mathbf{F}_C \in \mathbb{R}^6$ acting on the end-effector changes the dynamics of the robot to

$$\mathbf{u} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{J}^T\mathbf{F}_C. \tag{4.31}$$

In this case, the interaction between the robot and the environment has to be controlled. This kind of control can both be used to make the interaction with the environment safe (e.g., in a manipulation task) as well as to use the robot to simulate a behavior (e.g., in a haptic display task). We will discuss impedance control and hybrid control as examples of the application of the proposed framework; however, further control ideas such as parallel control can be treated in this framework, too.

4.3.3.1 Impedance Control

In impedance control, we want the robot to simulate the behavior of a mechanical system such as

$$\mathbf{M}_d(\ddot{\mathbf{x}}_d - \ddot{\mathbf{x}}) + \mathbf{D}_d(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \mathbf{P}_d(\mathbf{x}_d - \mathbf{x}) = \mathbf{F}_C, \tag{4.32}$$

where $\mathbf{M}_d \in \mathbb{R}^{6\times 6}$ denotes the mass matrix of the desired system, $\mathbf{F}_C \in \mathbb{R}^6$ denotes the measured external forces exerted onto the system, $\mathbf{D}_d \in \mathbb{R}^6$ denotes the desired damping, and $\mathbf{P}_d \in \mathbb{R}^6$ denotes the gains towards the desired position. Using Equation (4.24) from Section 4.3.2, we see that this can simply be brought in the standard form for tasks by

$$\mathbf{M}_d\mathbf{J}\ddot{\mathbf{q}} = \mathbf{F}_C - \mathbf{M}_d\ddot{\mathbf{x}}_d - \mathbf{D}_d(\dot{\mathbf{x}}_d - \mathbf{J}\dot{\mathbf{q}}) - \mathbf{P}_d(\mathbf{x}_d - \mathbf{f}(\mathbf{q})) - \mathbf{M}_d\dot{\mathbf{J}}\dot{\mathbf{q}},$$

after dropping all indices. From this we can infer the task description given by

$$\begin{aligned} \mathbf{A} &= \mathbf{M}_d\mathbf{J}, \\ \mathbf{b} &= \mathbf{F}_C - \mathbf{M}_d\ddot{\mathbf{x}}_d - \mathbf{D}_d(\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}_d) - \mathbf{P}_d(\mathbf{f}(\mathbf{q}) - \mathbf{x}_d) - \mathbf{M}_d\dot{\mathbf{J}}\dot{\mathbf{q}}. \end{aligned} \tag{4.33}$$

A major question in this context is the choice of the correct joint-space control law $\mathbf{u}_1(\mathbf{q}, \dot{\mathbf{q}})$, and the right metric to achieve such tasks.

Separation of both Systems through Kinematics. Similar as in end-effector control, a practical metric is $\mathbf{N} = \mathbf{M}^{-2}$ as this basically separates both dynamic systems into two separate ones as it will become apparent in this section. For simplicity, we make use of the joint-space control law $\mathbf{u}_1 = \mathbf{C} + \mathbf{G} + \mathbf{u}_0$ similar as before. This results in the control law

$$\begin{aligned} \mathbf{u} &= \mathbf{u}_1 + \mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}\left(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}(\mathbf{F} + \mathbf{u}_1)\right), \\ &= \mathbf{M}\left(\mathbf{M}_d\mathbf{J}\right)^{+}\left(\mathbf{F}_C - \mathbf{M}_d\ddot{\mathbf{x}}_d - \mathbf{D}_d(\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}_d) - \mathbf{P}_d(\mathbf{f}(\mathbf{q}) - \mathbf{x}_d) - \mathbf{M}_d\dot{\mathbf{J}}\dot{\mathbf{q}}\right) + \mathbf{C} + \mathbf{G} \\ &\quad + (\mathbf{I} - \mathbf{M}\left(\mathbf{M}_d\mathbf{J}\right)^{+}\mathbf{M}_d\mathbf{J}\mathbf{M}^{-1})\mathbf{u}_0. \end{aligned}$$

As $(\mathbf{M}_d\mathbf{J})^{+} = \mathbf{J}^T\mathbf{M}_d\left(\mathbf{M}_d\mathbf{J}\mathbf{J}^T\mathbf{M}_d\right)^{-1} = \mathbf{J}^{+}\mathbf{M}_d^{-1}$ since $\mathbf{M}_d$ is invertible, we can simplify this control law into

$$\begin{aligned} \mathbf{u} = {} & \mathbf{M}\mathbf{J}^{+}\mathbf{M}_d^{-1}(\mathbf{F}_C - \mathbf{M}_d\ddot{\mathbf{x}}_d - \mathbf{D}_d(\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}_d) - \mathbf{P}_d(\mathbf{f}(\mathbf{q}) - \mathbf{x}_d)) \\ & - \mathbf{M}\mathbf{J}^{+}\dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{C} + \mathbf{G} + \mathbf{M}(\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\mathbf{M}^{-1}\mathbf{u}_0. \end{aligned} \tag{4.34}$$

We note that $\ddot{\mathbf{x}}_d = \mathbf{M}_d^{-1}(\mathbf{F}_C - \mathbf{M}_d\ddot{\mathbf{x}}_d - \mathbf{D}_d(\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}_d) - \mathbf{P}_d(\mathbf{f}(\mathbf{q}) - \mathbf{x}_d))$ is a desired acceleration in task-space. This clarifies the previous remark: we have a first system which describes the interaction with the environment – and additionally we use a second, inverse-model type controller to execute the desired accelerations with our robot arm.

Dynamically Consistent Combination. Similar as in end-effector control, a practical metric is $\mathbf{N} = \mathbf{M}^{-1}$ which combines both dynamic systems into a big one employing Gauss' principle. For simplicity, we make use of the joint-space control law $\mathbf{u}_1 = \mathbf{C} + \mathbf{G} + \mathbf{u}_0$ similar as before. This results into the control law

$$\begin{aligned}
\mathbf{u} &= \mathbf{u}_1 + \mathbf{N}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}\left(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}(\mathbf{F} + \mathbf{u}_1)\right),\\
&= \mathbf{u}_1 + \mathbf{J}^T\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)^{-1}\left(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}(\mathbf{F} + \mathbf{u}_1)\right),\\
&= \mathbf{M}^{1/2}\left(\mathbf{M}_d\mathbf{J}\mathbf{M}^{-1/2}\right)^{+}\left(\mathbf{F}_C - \mathbf{D}_d(\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}_d) - \mathbf{P}_d(\mathbf{f}(\mathbf{q}) - \mathbf{x}_d) - \mathbf{M}_d\dot{\mathbf{J}}\dot{\mathbf{q}}\right) \qquad (4.35)\\
&+ \mathbf{C} + \mathbf{G} + (\mathbf{I} - \mathbf{M}\left(\mathbf{M}_d\mathbf{J}\right)^{+}\mathbf{M}_d\mathbf{J}\mathbf{M}^{-1})\mathbf{u}_0.
\end{aligned}$$

As $(\mathbf{M}_d\mathbf{J}\mathbf{M}^{-1/2})^{+} = \mathbf{M}^{-1/2}\mathbf{J}^T\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)^{-1}\mathbf{M}_d^{-1}$ since $\mathbf{M}_d$ is invertible, we can simplify this control law into

$$\begin{aligned}
\mathbf{u} &= \mathbf{J}^T\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)^{-1}\mathbf{M}_d^{-1}(\mathbf{F}_C - \mathbf{D}_d(\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}_d) - \mathbf{P}_d(\mathbf{f}(\mathbf{q}) - \mathbf{x}_d)) \qquad (4.36)\\
&- \mathbf{M}\mathbf{J}^{+}\dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{C} + \mathbf{G} + (\mathbf{I} - \mathbf{M}\mathbf{J}^{+}\mathbf{J}\mathbf{M}^{-1})\mathbf{u}_0.
\end{aligned}$$

We note that the main difference between the two control law is the location of the matrix $\mathbf{M}$.

4.3.3.2 Hybrid Control

In hybrid control, we intend to control the desired position of the end-effector $\mathbf{x}_d$ and the desired contact force exerted by the end-effector $\mathbf{F}_d$. Modern, common hybrid control approaches are essentially similar to our introduced framework (De Wit et al., 1996). Both are inspired by constrained motion and use this insight in order to achieve the desired task. In traditional hybrid control, a natural or artificial, idealized holomonic constraint $\boldsymbol{\phi}(\mathbf{q}, t) = \mathbf{0}$ acts on our manipulator, and subsequently the direction of the forces is determined through the virtual work principle of d'Alembert. We can make significant contributions here as our framework is a generalization of the Gauss' principle that allows us to handle even non-holomic constraints $\boldsymbol{\phi}(\mathbf{q}, \dot{\mathbf{q}}, t) = \mathbf{0}$ as long as they are given in the form

$$\mathbf{A}_{\boldsymbol{\phi}}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}} = \mathbf{b}_{\boldsymbol{\phi}}(\mathbf{q}, \dot{\mathbf{q}}). \qquad (4.37)$$

$\mathbf{A}_{\boldsymbol{\phi}}$, $\mathbf{b}_{\boldsymbol{\phi}}$ depend on the type of the constraint, e.g., for scleronomic, holomonic constraints $\boldsymbol{\phi}(\mathbf{q}) = \mathbf{0}$, we would have $\mathbf{A}_{\boldsymbol{\phi}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}_{\boldsymbol{\phi}}$ and $\mathbf{b}_{\boldsymbol{\phi}}(\mathbf{q}, \dot{\mathbf{q}}) = -\dot{\mathbf{J}}_{\boldsymbol{\phi}}\dot{\mathbf{q}}$ with $\mathbf{J}_{\boldsymbol{\phi}} = \partial\boldsymbol{\phi}/\partial\mathbf{q}$ as in (De Wit et al., 1996). Additionally, we intend to exert the contact force $\mathbf{F}_d$ in the task; this can be achieved if we choose the joint-space control law

$$\mathbf{u}_1 = \mathbf{C} + \mathbf{G} + \mathbf{J}_{\boldsymbol{\phi}}^T\mathbf{F}_d. \qquad (4.38)$$

From the previous discussion, this constraint is achieved by the control law

$$\mathbf{u} = \mathbf{u}_1 + \mathbf{N}^{-1/2} \left(\mathbf{A}_{\phi}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+} (\mathbf{b}_{\phi} - \mathbf{A}_{\phi}\mathbf{M}^{-1}(\mathbf{F} + \mathbf{u}_1)), \tag{4.39}$$

$$= \mathbf{C} + \mathbf{G} + \mathbf{N}^{-1/2} \left(\mathbf{A}_{\phi}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+} \mathbf{b}_{\phi} \tag{4.40}$$

$$+ \mathbf{N}^{-1/2}(\mathbf{I} - \left(\mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+} \mathbf{A}\mathbf{M}^{-1}\mathbf{N}^{-1/2})\mathbf{N}^{1/2}\mathbf{J}_{\phi}^{T}\mathbf{F}_d.$$

Note that the exerted forces act in the null-space of the achieved; therefore both the constraint, and therefore the force can be set independently.

4.4 Evaluations

The main contribution of this chapter is the unifying methodology for deriving robot controllers. In order to demonstrate the framework's feasibility for providing implementable controllers for real robots, we have chosen a few of the controllers derived here and evaluate them with a simple tracking task. In future work, we plan to evaluate all controllers presented in this chapter with more complex tasks.

The joint-space trajectory controller derived in this chapter is already well established in the literature, and such that further evaluation is not necessary. Of more interest to us are the end-effector controllers, since they introduce added complexity, particularly the problem of redundancy resolution. Due to a lack of force sensors on our experimental platform, we are unable to implement the impedance or hybrid controllers, but plan to do so in our future work. For this chapter, we evaluate the three end-effector controllers from Section 4.3.2: (i) the resolved-acceleration kinematic controller (with metric $\mathbf{N} = \mathbf{M}^{-2}$) in Equation (4.28), (ii) Khatib's operational space control law ($\mathbf{N} = \mathbf{M}^{-1}$) in Equation (4.29), and (iii) the identity metric control law ($\mathbf{N} = \mathbf{I}$) in Equation (4.30).

As an experimental platform, we use the Sarcos Dextrous Master Arm, a hydraulic manipulator with an anthropomorphic design shown in Figure 4.1 (b). Its seven degrees of freedom mimic the major degrees of freedom of the human arm, i.e., the three in the shoulder, one in the elbow and in the wrist.

The robot's end-effector tracks a planar "figure-eight (8)" pattern in task space at two different speeds. In order to stabilize the null-space trajectories, we choose a PD control in joint space which pulls the robot towards a fixed rest posture, $\mathbf{q}_{\text{rest}}$; this control law is given by

$$\mathbf{u}_0 = \mathbf{M}\left(\mathbf{K}_{P0}\left(\mathbf{q}_{\text{rest}} - \mathbf{q}\right) - \mathbf{K}_{D0}\dot{\mathbf{q}}\right).$$

Additionally we apply gravity, centrifugal and Coriolis force compensation, such that $\mathbf{u}_1 = \mathbf{u}_0 + \mathbf{C} + \mathbf{G}$. For consistency, all three controllers are assigned the same gains both for the task and joint space stabilization.

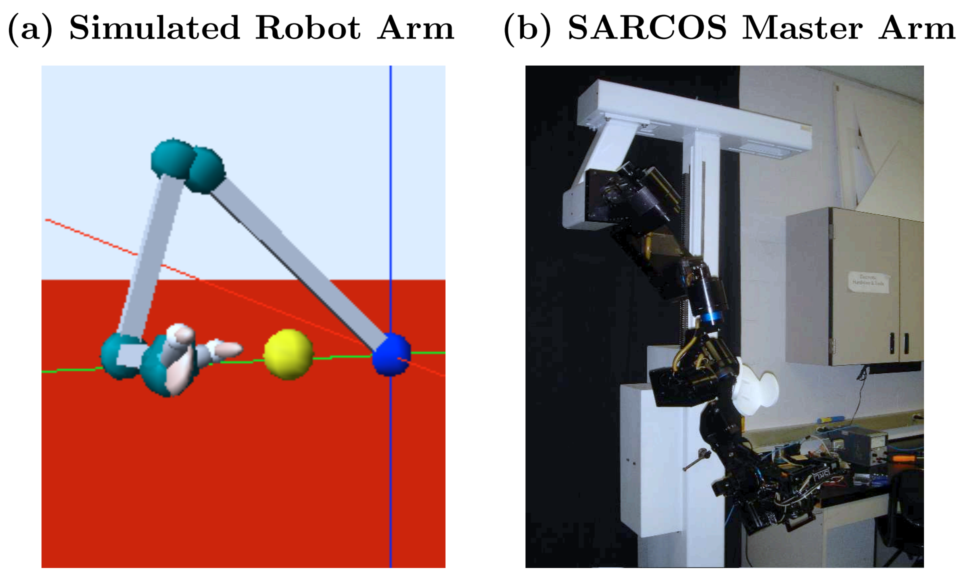


Figure 4.1: Setups in which we evaluate the designed controllers: (a) a physical simulation of the SARCOS Master Arm, (b), the robot arm.

Figure 4.2 shows the end-point trajectories of the three controllers in a slow pattern of 8 seconds per cycle "figure-eight (8)". Figure 4.3 shows a faster pace of 4 seconds per cycle. All three controllers have similar end-point trajectories and result in fairly accurate task achievement. Each one has an offset from the desired (thin black line), primarily due to the imperfect dynamics model of the robot. The root mean squared errors (RMS) between the actual and the desired trajectory in task-space for each of the controllers are shown in the Table 4.1.

As expected, the performance of the three controllers is very similar in task space. However, the resolved-acceleration kinematic controller ($\mathbf{N} = \mathbf{M}^{-2}$) appears to have a slight advantage here. The reason is most likely due to errors in the dynamics

Table 4.1: This table shows the root mean squared error results of the tracking achieved by the different control laws.

Metric	**Slow RMS error [m]**	**Fast RMS error [m]**
$\mathbf{N} = \mathbf{M}^{-2}$	0.0122	0.0130
$\mathbf{N} = \mathbf{M}^{-1}$	0.0126	0.0136
$\mathbf{N} = \mathbf{I}$	0.0130	0.0140

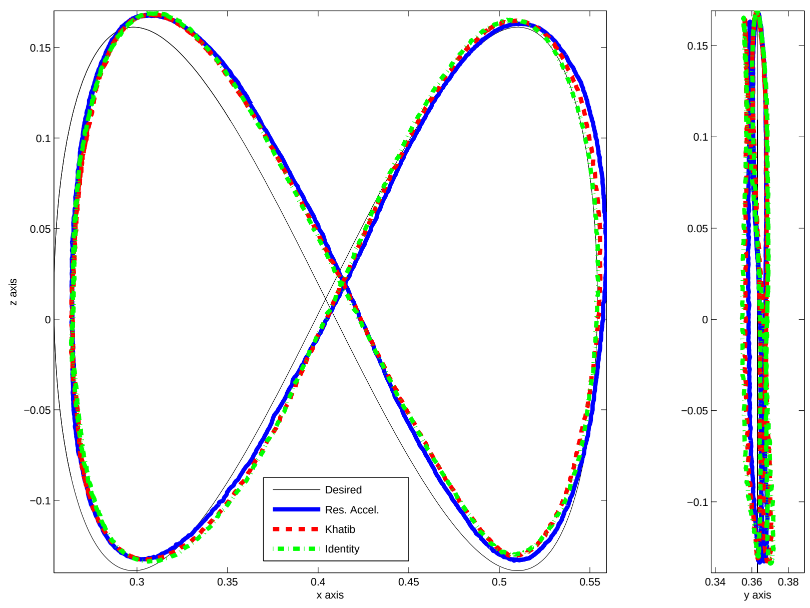


Figure 4.2: This figure shows the three end-effector trajectory controllers tracking a "figure eight (8)" pattern at 8 seconds per cycle. On the left is the x-z plane with the y-z plane on the right. All units are in meters.

model, since the effect of these is amplified by the inversion of the mass matrix in the control laws given in Equations (4.29, 4.30) while the decoupling of the dynamics and kinematics provided by the controller in Equation (4.28) can be favorable as the effect of the modeling error is not increased. Clearly, more accurate model parameters of the manipulator's rigid body dynamics would result in a reduction of the gap between these control laws as we have confirmed in simulations. Figure 4.4 shows how the joint space trajectories appear for the fast cycle. Although end-point trajectories were very similar, joint space trajectories differ significantly due to the different optimization criteria of each control law.

4.5 Conclusion and Proposed Future Work

In this section, we give a short conclusion on the current state of our motor skill execution framework and subsequently discuss the future topics which we intend to address in this area until the completion of this thesis.

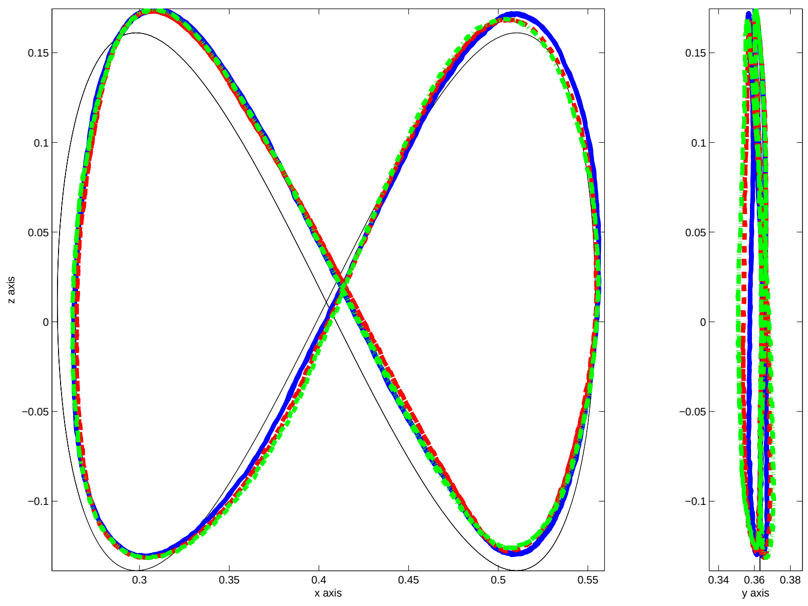


Figure 4.3: The same three controllers tracking the same "figure eight (8)" pattern at a faster pace of 4 seconds per cycle. The labels and units remain the same as in Figure 4.2.

4.5.1 Conclusion on the Current State

In this chapter we have presented a novel optimal control framework which allows the development of a unified approach for deriving robot control laws. We have shown in detail how we can make use of both the robot model and a task description in order to create the control law which is optimal with respect to the squared motor command under a metric while *perfectly* fulfilling the task *at each instant of time*. We have discussed how to realize stability both in task as well as in joint-space for this framework.

Building on that foundation, we demonstrated how a variety of control laws–which on first inspection appear rather unrelated to one another–can be derived using this straightforward framework. The covered types of tasks include joint-space trajectory control for both fully actuated and overactuated robots, end-effector trajectory control, impedance and hybrid control.

The implemention of three of the end-effector trajectory control laws resulting from our unified framework on a real-world Sarcos Master Arm robot has been carried out. As expected, the behavior in task space is very similar for all three control laws; yet,

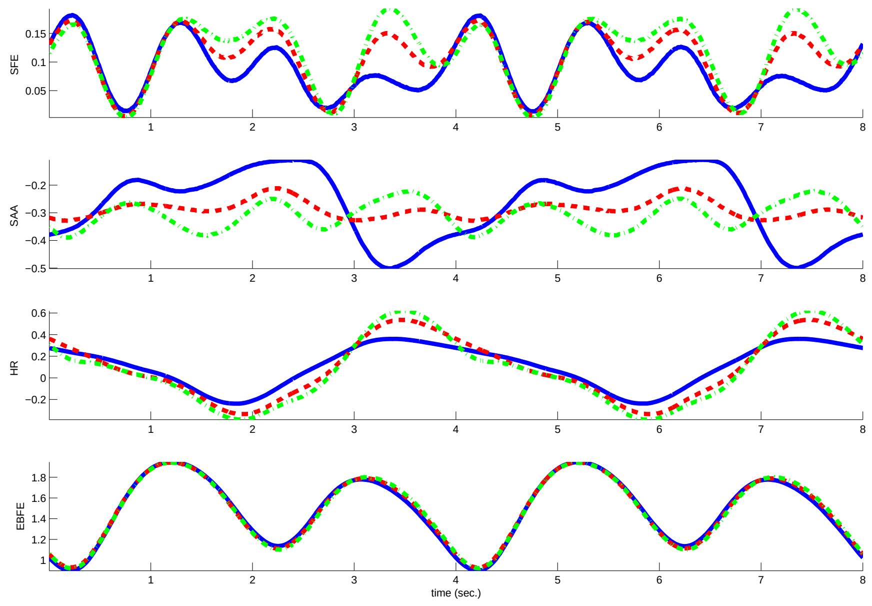


Figure 4.4: Joint space trajectories for the four major degrees of freedom, i.e., shoulder flexion-extension (SFE), shoulder adduction-abduction (SAA), humeral rotation (HR) and elbow flexion-extension (EBFE), are shown here. Joint angle units are in radians. The labels are identical to the ones in Figure 4.2.

they result in very different joint-space behaviors due to the different cost functions resulting from the different metrics of each control law.

The major contribution of this chapter is the unified framework that we have developed. It allows a derivation of a variety of previously known controllers, and promises the easy development of a host of novel ones. The particular controllers reported in this chapter were selected primarly for illustarting the applicability of this framework and showing its strength in unifying different control algorithms using a common building principle.

4.5.2 Proposed Future Work

Up to now, we have only presented the Gauss' control framework as a unifying framework for the generation tracking control laws in joint and task space as well as a framework for creating force control laws. Let us here outline the projects which we are currently working on and which we intend to complete together with this thesis:

1. The major advantage of the Gauss' control law in comparison to other frameworks for control is that can handle non-holomonic robots and tasks in exactly the same way as holomonic ones. We are therefore currently working on evaluations and comparisons for non-holomonic control using mobile robots.
2. When using an underactuated system, we can treat this in a similar fashion where part of the system becomes a constraint.
3. Particularly important and obvious is the application of this framework towards motor primitives, i.e., the motor primitives become attractor constraints in the Gauss' framework.
4. As we have seen in the evaluations, the rigid body model does not completely capture the dynamics of our SARCOS robot arm and the control law for this reason has a significant offset from the trajectory. For this reason, we need to bring model learning into this framework. It is fairly likely that this point will result into a complete new chapter of this thesis.

Points 1-3 are straightforward and will most likely be achieved by Fall 2005. Point 4 is of particular importance and has to be done with care. It is clear that here a pseudo-inverse needs to be learned which for example can be achieved using the forward-inverse model approach (Wolpert & Kawato, 1998) and that the chosen joint-space control law determines the type of model learned.

Chapter 5

Application to Robotics

In this chapter, we intend to give a brief overview on the intended application to robotics. This overview can at this point only demonstrate what foundations exist and where we want to take these. Therefore, this chapter is rather short in comparison with the previous ones.

5.1 T-ball Swing

As already outlined in Chapter 3, we are working on an implementation of T-ball on a real robot as shown in Figure 3.8. For this, we are using both policy search techniques presented in Chapter 3 as well as the motor primitive framework outlined in Chapter 2. As we have made clear in previous sections this work has already started and preliminary results are available. This piece of work has to be finished up until the completion.

5.2 Learning of Locomotion

The project of learning locomotion using the motor primitive framework is slightly more complicated as it involves a variety of different types of problems, i.e., the generations of gaits patterns, foot placement problems (which can even involve perceptual problems) and the stabilization of the robot so that it does not fall over. From this perspective, learning of locomotion is a prime example of motor skill aquisition. To date, there has only been work on the aquisition of gait patterns through supervised learning (Schaal et al., 2003).

We have a variety of different platforms for testing learning biped locomotion systems. These include the simulations of the robot DB-2 in Figure 5.1 (a), the small biped robot DB-chan in Figure 5.1 (b) as well as the real robot DB-2 in Figure 5.1 (c). Both physical robots are located with our collaborators at ATR in Japan. There has been work on DB chan with gait patterns through supervised learning (Schaal et al., 2003) as mentioned before.

We intend to tackle the following problems which together will allow us to learn basic forms of locomotion:

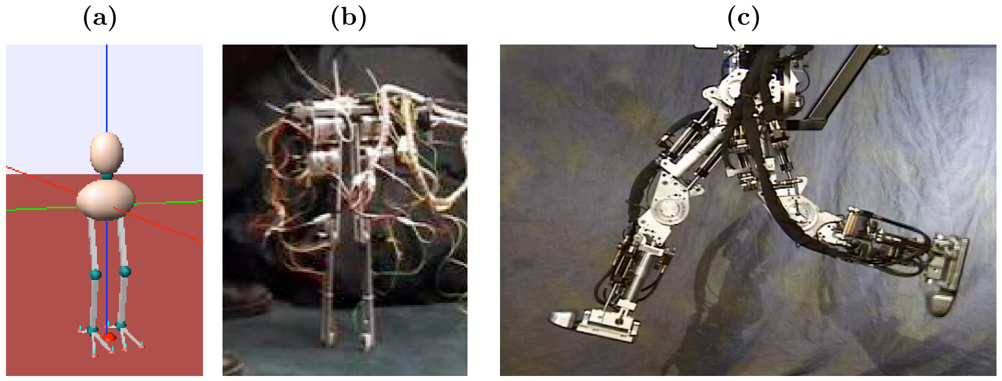


Figure 5.1: This figure shows the different simulations and robots of which we can make use for accomplishing the tasks outlined in this thesis proposal. In (a) our simulation platform is shown, in (b) the robot DB chan, and in (c) the legs of DB 2.

1. We intend to learn Gauss control laws for the stabilization of the biped using model-based learning.
2. We intend to improved gaits based on rythmic motor primitives and learned with supervised learning in (Schaal et al., 2003) using reinforcement learning.

Further work on foot placement with discrete primitives on a nonuniform terrain might become useful.

5.3 Complex Movements

For complex skill learning tasks, we propose to work on problems similar as in Figure 5.1, i.e., box tumbling and motorized travelling salesman problems. In here, we can make use of a variety of further plattforms such as out robot arm as before in Figure 5.1.

Chapter 6

Conclusion

In conclusion, we have accomplished the following steps

1. We have discussed what hierachical frameworks exist and how they could be employed for learning motor skills.

2. We have presented a hierarchical framework for the representation and learning of motor skills.

3. The natural actor-critic methods have yielded a variety of theoretical insights into previous reinforcement learning problems and has been successfully applied to motor primitives for simple and complex motor tasks including the T-ball swing on an antropomorphic robot arm.

4. We have shown theoretically and in experiments that the generalized Gauss' control framework with a squared metric is suitable when accurate robot dynamics models exist and connects to previous control approaches.

Until the completion of this thesis in Fall 2006, we propose the following projects. Each of these projects has a approximate date of completion.

1. Complete project on policy search based T-ball learning with comparisons. *Approximate Date of Completion*: May, 2005.

2. Extend the Gauss' control framework to more complicated metrics and apply it to at least one nonholomonic system. *Approximate Date of Completion*: June-July, 2005.

3. Aquire Gauss' controllers using model learning. This step is independent from Step 2. *Approximate Date of Completion*: August, 2005.

4. Move from the policy-gradient based method for learning motor primitives towards a probabilistic policy search method which is applied both on the motor primitive as well as motor task level. *Approximate Date of Completion*: October, 2005.

5. Use the currently best methods for policy search and the model-learning based Gauss' control to learn locomotion for legged robots. The success of this step depends on steps 1–4. *Approximate Date of Completion*: January-February, 2006.

6. Learning complex motor skills such as box tumbling and the motorized traveling salesman problem which employ both sequencing and superposition of motor primitives, and require primitives for forces as well. *Approximate Date of Completion*: July, 2006.

If time permits, we would like to insert a project on how probabilistic policy search techniques can be extended to infer cost functions for motor policies.

References

Albu-Schaefer, A. (2002). *Regelung von robotern mit elastischen gelenken am beispiel der dlr-leichtbauarme.* Unpublished doctoral dissertation, Munich University of Technology, Munich, Germany.

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*.

Amit, R., & Mataric, M. (2002). Learning movement sequences from demonstration. In *International conference on development and leanring (icdl-2002).* Cambridge, MA: June 12-15.

Arbib, M. A. (1981). Perceptual structures and distributed motor control. In V. B. Brooks (Ed.), *Handbook of physiology, section 2: The nervous system vol. ii, motor control, part 1* (pp. 1449–1480). Bethesda, MD: American Physiological Society.

Atkeson, C. G., & Schaal, S. (1997). Robot learning from demonstration. In D. H. Fisher Jr. (Ed.), *Machine learning: Proceedings of the fourteenth international conference (icml '97)* (pp. 12–20). Nashville, TN, July 8-12, 1997: Morgan Kaufmann.

A.W. Salatian, Y. Z., K.Y. Yi. (1997). Reinforcement learning for a biped robot to climb sloping surfaces. *J. Robot. Syst.*, *14*, 283–296.

Bagnell, J. A., Kakade, S., Ng, A. Y., & Schneider, J. (2004). Policy search by dynamic programming. In S. Thrun, L. Saul, & B. Schölkopf (Eds.), *Advances in neural information processing systems 16.* Cambridge, MA: MIT Press.

Baird, L. (1993). *Advantage updating.* Wright-Patterson Air Force Base, OH: Wright Laboratory.

Bartlett, P. (2002). An introduction to reinforcement learning theory: Value function methods. 184-202.

Barto, A., & Mahadevan, S. (2003). Recent advances in hierachical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, *13*, 41–77.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *13*, 834–846.

Baxter, J., & Bartlett, P. (1999). Direct gradient-based reinforcement learning. *Journal of Artificial Intelligence Research.*

Baxter, J., Bartlett, P., & Weaver, L. (2001). Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research*, *15.*

Bellman, R. (1957). *Dynamic programming.* Princeton, NJ: Princeton University Press.

Bellman, R. E. (1967). *Introduction to the mathematical theory of control processes* (Vol. 40-I). New York, NY: Academic Press.

Bellman, R. E. (1971). *Introduction to the mathematical theory of control processes* (Vol. 40-II). New York, NY: Academic Press.

Bellman, R. E., & Kalaba, R. E. (1965). *Dynamic programming and modern control theory.* Academic Press.

Benbrahim, H. (1996). *Biped dynamic walking using reinforcement learning.* (Ph.D. Thesis at University of New Hampshire)

Benbrahim, H., & Franklin, J. (1997). Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems.*

Berny, A. (2000). Statistical machine learning and combinatorial optimization. *In L. Kallel, B. Naudts, and A. Rogers, editors, Theoretical Aspects of Evolutionary Computing, Lecture Notes in Natural Computing*, *0*(33).

Bertsekas, D., & Tsitsiklis, J. (1996). *Neuro-dynamic programming.* Belmont, MA: Athena Scientific.

Billard, A., & Mataric, M. (2001). Learning human arm movements by imitation: Evaluation of a biologically-inspired architecture. *Robotics and Autonomous Systems*, *941*, 1–16.

Billard, A., & Schaal, S. (2002). Computational elements of robot learning by imitation. In *American mathematical society central section meeting.* Madison, Oct.12-13,2002: Providence, RI: American Mathematical Society.

Bishop, C. M. (1995). *Neural networks for pattern recognition.* New York: Oxford University Press.

Boyan, J. (1999). Least-squares temporal difference learning. 49–56.

Bradtke, S., Ydstie, E., & Barto, A. (1994). *Adaptive linear quadratic control using policy iteration.* Amherst, MA: University of Massachusetts.

Branicky, M., & Mitter, S. (1995). Algorithms for optimal hybrid control. In *Proceedings of the ieee conference on decision and control* (pp. 2661–6). .

Branicky, M. S. (1998). Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Trans. on Automatic Control*, *43*(4), 475–482.

Bruyninckx, H., & Khatib, O. (2000). Gauss' principle and the dynamics of redundant and constrained manipulators. *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, 2563–2569.

Bryson, A. E. (1981). *Applied optimal control: Optimization, estimation, and control.* Hemisphere Pub. Corp.

Burridge, R. R., Rizzi, A. A., & Koditschek, D. E. (1999). Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, *18*(6), 534-555.

Buss, M., Stryk von, & O., G., Schmidt. (2000). Towards hybrid optimal control. *at-Automatisierungstechnik*, *48*(09), 448-459.

Caines, P. E., & Wei, Y. jun. (1998). Hierarchical hybrid control systems: A lattice theoretic formulation. *IEEE AC*, *43*(4), 501-508.

Chung, W., Chung, W., & Y.Youm. (1993). Null torque based dynamic control for kinematically redundant manipulators. *Journal of Robotic Systems*, *10*(6), 811–834.

C. Zhou, Q. M. (2000). Reinforcement learning with fuzzy evaluative feedback for a biped robot. In *Proceedings of the ieee international conference on robotics and automation* (pp. 3829–3834). .

Dautenhahn, K., & Nehaniv, C. L. (Eds.). (2002). *Imitation in animals and artifacts.* Cambridge, MA: MIT Press.

Dayan, P. (1990). Reinforcement comparison. In D. Touretzky, J. Elman, T. Sejnowski, & G. Hinton (Eds.), *Proceedings of the 1990 connectionist models summer school* (p. 45-51). San Mateo, CA: Morgan Kaufmann.

Dayan, P., & Hinton, G. E. (1997). Using expectation-maximization for reinforcement learning. *Neural Computation*, *9*(2), 271-278.

De Wit, C. A. C., Siciliano, B., & Bastin, G. (1996). *Theory of robot control.* Springer-Verlag Telos.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, *13*, 227-303.

Doya, K., Samejima, K., Katagiri, K., & Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural Comput*, *14*(6), 1347–69.

Duflo, M. (1997). *Random iterative models.* Springer Verlag.

Dyer, P., & McReynolds, S. R. (1970). *The computation and theory of optimal control.* New York: Academic Press.

Edstroem, K. (1999). *Switched bond graphs: Simulation and analysis.* Unpublished doctoral dissertation, Linkoeping University, Linkoeping, Sweden.

Febbraro, A. D., Giua, A., & Menga, G. (Eds.). (2001, January). *Special issue on "hybrid petri nets"* (No. 1/2). Kluwer Academic.

Fidelman, P., & Stone, P. (2004). Learning ball acquisition on a physical robot. In *2004 international symposium on robotics and automation (isra).* Queretaro. Mexico.

Fuerverger, A., McLeish, D., Kreimer, J., & Rubinstein, R. (1989). Sensitivity analysis and the "what if" problem in simulation analysis. *Math. Comput. Modelling*, *12*, 193–219.

Glynn, P. (1987). Likelihood ratio gradient estimation: an overview. In *Proceedings of the 1987 winter simulation conference* (p. 366-375). Atlanta, GA.

Glynn, P. (1989). Optimization of stochastic systems via simulation. In *Proceedings of the 1989 winter simulation conference* (pp. 90–105). Atlanta, GA.

Glynn, P. (1990). Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, *33*(10), 75–84.

Greensmith, E., Bartlett, P., & Baxter, J. (2001). Variance reduction techniques for gradient estimates in reinforcement learning. *Advances in Neural Information Processing Systems*, *14*(34).

Greensmith, E., Bartlett, P. L., & Baxter, J. (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, *5*, 1471–1530.

Gullapalli, V. (1993a). Learning control under extreme uncertainty. 327–334.

Gullapalli, V. (1993b). Learning control under extreme uncertainty. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems* (Vol. 5, pp. 327–334). Morgan Kaufmann, San Mateo, CA.

Gullapalli, V. (1995). Skillful control under uncertainty via direct reinforcement learning. *Robotics and Autnomous Systems*, *15*, 237–246.

Gullapalli, V., Franklin, J., & Benbrahim, H. (1994). Aquiring robot skills via reinforcement learning. *IEEE Control Systems*, -(39).

Hardt, M., & Stryk, O. von. (2000). Towards optimal hybrid control solutions for gait patterns of a quadruped. In M. Armada & P. Gonzalez de Santos (Eds.), *Proc. clawar 2000 – 3rd international conference on climbing and walking robots* (p. 385-392). Bury St. Edmunds and London, UK: Professional Engineering Publishing.

Harville, D. A. (2000). *Matrix algebra from a statistician's perspective.* Springer Verlag.

Hasdorff, L. (1976). *Gradient optimization and nonlinear control.* John Wiley & Sons.

Henzinger, T. A. (1996). The theory of hybrid automata. In *Proceedings of the 11th annual symposium on logic in computer science (lics)* (p. 278-292). IEEE Computer Society Press.

Hirzinger, G., Sporer, N., Albu-Schäffer, A., Hähnle, M., Krenn, R., Pascucci, A., & Schedl, M. (2002). Dlr's torque-controlled light weight robot iii - are we reaching the technological limits now? In *Icra* (p. 1710-1716).

Hsu, P., Hauser, J., & Sastry, S. (1989). Dynamic control of redundant manipulators. *Journal of Robotic Systems*, *6*(2), 133–148.

Ijspeert, A., Nakanishi, J., & Schaal, S. (2001). Trajectory formation for imitation with nonlinear dynamical systems [conference]. In *Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS2001)* (p. 752-757). Maui, Hawai.

Ijspeert, A., Nakanishi, J., & Schaal, S. (2002a). Learning attractor landscapes for learning motor primitives [conference]. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in neural information processing systems 15 (nips2002)* (pp. 1547–1554).

Ijspeert, A., Nakanishi, J., & Schaal, S. (2002b). Learning rhythmic movements by demonstration using nonlinear oscillators [conference]. In *Proceedings of the ieee/rsj int. conference on intelligent robots and systems (iros2002)* (pp. 958–963).

Ijspeert, A., Nakanishi, J., & Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In *Advances in neural information processing systems 15.* Cambridge, MA: MIT Press.

Inamura, T., Iwaki, T., Tanie, H., & Nakamura, Y. (2004). Embodied symbol emergence based on mimesis theory. *International Journal of Robotics Research*, *23*(4-5), 363–377.

Inamura, T., Toshima, I., & Nakamura, Y. (2002). Acquisition and embodiment of motion elements in closed mimesis loop. In *International conference on robotics and automation (icra2002)* (pp. 1539–1544). Washinton, May 11-15 2002.

Jaakkola, T., Singh, S. P., & Jordan, M. I. (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems* (Vol. 7, pp. 345–352). The MIT Press.

Jacobson, D. H., & Mayne, D. Q. (1970). *Differential dynamic programming.* New York: American Elsevier Publishing Company, Inc.

Johansson, K. H. (2000, Spring). *Eecs291e hybrid systems.* (UC Berkeley Lecture)

Johansson, M., & Rantzer, A. (1998). Computation of piecewise quadratic lyapunov functions for hybrid systems. *IEEE Trans. on Automatic Control*, *43*(4), 555–559.

Jun Morimoto, K. D. (2000). Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. In *Icml* (p. 623-630).

Kakade, S. (2001). A natural policy gradient. In *Advances in neural information processing systems* (Vol. 14).

Kakade, S. A. (2002). Natural policy gradient. *Advances in Neural Information Processing Systems 14*.

Kawato, M. (1999). Internal models for motor control and trajectory planning. *Curr Opin Neurobiol*, *9*(6), 718–727.

Kawato, M., Gandolfo, F., Gomi, H., & Wada, Y. (1994). Teaching by showing in kendama based on optimization principle. In *Proceedings of the international conference on artificial neural networks (icann'94)* (Vol. 1, pp. 601–606).

K.C.Suh, & Hollerbach, J. M. (1987). Local versus global torque optimization of redundant manipulators. *Proc. IEEE Int. Conference on Robotics and Automation*, 619–624.

Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, *3*(1), 43–53.

Kimura, H., & Kobayashi, S. (1998). An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value function. *15th International Conference on Machine Learning*, 278–286.

Kohl, N., & Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE international conference on robotics and automation.* New Orleans, LA.

Konda, V. (2002). Actor-critic algorithms. *Ph.D. Thesis (MIT)*, *3*(36).

Konda, V., & Tsitsiklis, J. (2000). Actor-critic algorithms. *Advances in Neural Information Processing Systems 12*.

Konda, V., & Tsitsiklis, J. (2001). Actor-critic algorithms. *Submitted to SIAM Journal on Control and Optimisation*(38).

Lawrence, G., Cowan, N., & Russell, S. (2003). Efficient gradient estimation for motor control learning. In *Proc. uai-03.* Acapulco, Mexico.

Leithead, D. J. (2000). Survey of gain-scheduling analysis and design. *International Journal of Control*, *73*(11), 1001-1025.

Lemmon, M., He, K., & Markovsky, I. (1999). Supervisory hybrid systems. *IEEE Control Systems*, *19*(4), 42-55.

Li, W., & Todorov, E. (2004). Iterative linear-quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the 1st international conference on informatics in control, automation and robotics* (Vol. 1, pp. 222–229). INSTICC Press.

Lozano-Perez, T., Mason, M. T., & Taylor, R. H. (1984). Automatic synthesis of fine-motion strategies for robots. *Int. J. Rob. Res.*, *3*(1), 3–23.

Lygeros, J., Tomlin, C., & Sastry, S. S. (2005). *Hybrid systems and control.* In preperation.

Lynch, N., Segala, R., Vaandrager, F., & Weinberg, H. (1996). Hybrid i/o automata. In R. Alur, T. Henzinger, & E. Sontag (Eds.), *Hybrid systems iii: Verification and control* (Vol. 1066, p. 496-510). Springer-Verlag.

Marbach, P., & Tsitsiklis, J. (1999). Simulation-based optimization of markov reward processes: implementation issues.

Marbach, P., & Tsitsiklis, J. N. (2003). Approximate gradient methods in policy-space optimization of markov reward processes. *Journal of Discrete Event Dynamical Systems*, *13*, 111-148.

McGovern, A., & Barto, A. G. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 2001 international conference on machine learning.*

Miyamoto, H., Gandolfo, F., Gomi, H., Schaal, S., Koike, Y., Osu, R., Nakano, E., & Kawato, M. (1995). A kendama learning robot based on a dynamic optimization theory. In *Preceedings of the 4th ieee international workshop on robot and human communication (ro-man'95)* (pp. 327–332). Tokyo.

Miyamoto, H., Gandolfo, F., Gomi, H., Schaal, S., Koike, Y., Rieka, O., Nakano, E., Wada, Y., & Kawato, M. (1996). A kendama learning robot based on a dynamic optimiation principle. In *Preceedings of the international conference on neural information processing* (pp. 938–942). Hong Kong.

Miyamoto, H., & Kawato, M. (1998). A tennis serve and upswing learning robot based on bi-directional theory. *Neural Networks*, *11*, 1331–1344.

Miyamoto, H., Schaal, S., Gandolfo, F., Koike, Y., Osu, R., Nakano, E., Wada, Y., & Kawato, M. (1996). A kendama learning robot based on bi-directional theory. *Neural Networks*, *9*(8), 1281–1302.

Moon, T., & Stirling, W. (2000). *Mathematical methods and algorithms for signal processing.* Prentice Hall.

Morimoto, J. (2002). Robust low torque biped walking using differential dynamic programming with a minimax criterion. *submitted to CLAWAR.*

Morimoto, J., & Atkeson, C. G. (2003). Minimax differential dynamic programming: An application to robust biped walking. In S. T. S. Becker & K. Obermayer (Eds.), *Advances in neural information processing systems 15* (pp. 1539–1546). Cambridge, MA: MIT Press.

Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., & Kawato, M. (2004). Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, *47*(2-3), 79–91.

Neumann, K., & Morlock, M. (2002). *Operations research.* Hanser Verlag.

Paine, R. W., & Tani, J. (2004). Motor primitive and sequence self-organization in a hierarchical recurrent neural network. *Neural Netw*, *17*(8-9), 1291–309.

Park, J., Chung, W.-K., & Youm, Y. (1995). Specification and control of motion for kinematically redundant manipulators. *Proc. Internationational Conference of Robotics Systems.*

Park, J., Chung, W.-K., & Youm, Y. (2002). Characterization of instability of dynamic control for kinematically redundant manipulators. *Proc. IEEE Int. Conference on Robotics and Automation.*

Parr, R., & Russell, S. (1998). Reinforcement learning with hierarchies of machines. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), *Advances in neural information processing systems* (Vol. 10). The MIT Press.

Perkins, T., & Barto, A. (2001). Lyapunov-constrained action sets for reinforcement learning. In *Proceedings of the eighteenth international conference on machine learning* (pp. 409–416).

Peters, J., Mistry, M., & Udwadia, F. E. (2005). A novel methodology for the control of robotic systems. In *Submitted to the international conference on robot systems (iros).*

Peters, J., Vijayakumar, S., & Schaal, S. (2003a). Reinforcement learning for humanoid robotics. In *Humanoids2003, third ieee-ras international conference on humanoid robots.* Karlsruhe, Germany, Sept.29-30.

Peters, J., Vijayakumar, S., & Schaal, S. (2003b). Scaling reinforcement learning paradigms for motor learning. In *Proceedings of the 10th joint symposium on neural computation (jsnc 2003).* Irvine, CA, May 2003.

Peters, J., Vijayakumar, S., & Schaal, S. (2004). *Linear quadratic regulation as benchmark for policy gradient methods.* Los Angeles, CA: USC Technical Report.

Peters, J., Vijaykumar, S., & Schaal, S. (2003). Reinforcement learning for humanoid robotics. *IEEE International article on Humandoid Robots (HUMANOIDS).*

Pollard, N. (2004). Closure and quality equivalence for efficient synthesis of grasps from examples. *International Journal of Robotics Research*, *6*(23), 595–614.

Pollard, N. S., & Hodgins, J. K. (2002). Generalizing demonstrated manipulation tasks. In *Workshop on the algorithmic foundations of robotics (wafr '02).*

Pook, P. K., & Ballard, D. H. (1993). Recognizing teleoperated manipulations. In *Proceedings ieee international conference on robotics and automation* (Vol. 3, pp. 913–918). Atlanta, GA, May 1993: Piscataway, NJ: IEEE.

Ramadge, P. J., & Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM J. Contr. Optimiz.*, *25*, 206-230.

Sabes, P. N., & Jordan, M. I. (1996). Reinforcement learning by probability matching. *Advances in Neural Information Processing Systems*, *8*, 1080–1086.

Samejima, K., Doya, K., & Kawato, M. (2003). Inter-module credit assignment in modular reinforcement learning. *Neural Netw*, *16*(7), 985–94.

Schaal, S. (1997). Learning from demonstration. In M. C. Mozer, M. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems 9* (pp. 1040–1046). Cambridge, MA: MIT Press.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, *3*(6), 233–242.

Schaal, S., Ijspeert, A., & Billard, A. (2004). Computational approaches to motor learning by imitation. In C. D. Frith & D. Wolpert (Eds.), *The neuroscience of social interaction* (pp. 199–218). Oxford: Oxford University Press.

Schaal, S., Peters, J., Nakanishi, J., & Ijspeert, A. (2003). Control, planning, learning, and imitation with dynamic movement primitives. In *Workshop on bilateral paradigms on humans and humanoids, ieee international conference on intelligent robots and systems (iros 2003).* Las Vegas, NV, Oct. 27-31.

Schaal, S., Peters, J., Nakanishi, J., & Ijspeert, A. (2004). Learning movement primitives. In *International symposium on robotics research (isrr2003).* Ciena, Italy: Springer.

Schoknecht, R. (2003). Optimality of reinforcement learning algorithms with linear function approximation. In S. T. S. Becker & K. Obermayer (Eds.), *Advances in neural information processing systems 15* (pp. 1555–1562). Cambridge, MA: MIT Press.

Sentis, L., & Khatib, O. (2004). Task-oriented control of humanoid robots through prioriization. In *Ieee-ras/rsj international conference on humanoid robots.* Santa Monica, CA, November 2004.

Spall, J. C. (2003). *Introduction to stochastic search and optimization: Estimation, simulation, and control.* Hoboken, NJ: Wiley.

Sternad, D., & Schaal, S. (1998). Segmentation of endpoint trajectories does not imply segmented control. In *Abstracts of the eigth annual meeting of neural control of movement (ncm)* (p. F-6). Key West, Florida, April 14-19.

Stryk, M. G. O. von. (2000). Decomposition of mixed-integer optimal control problems using branch and bound and sparse direct collocation. In *Adpm.* Dortmund, Germany.

Stryk, O. von. (1999). *User's guide for dircol version 2.1: A direct collocation method for the numerical solution of optimal control problems* (Report). Lehrstuhl M2 Hoehere Mathematik und Numerische Mathematik, Technische Universit ?at M ?unchen.

Sutton, R. (2000). Policy gradient methods for reinforcement learning with function approximation. *Presentation at NIPS*, *12*(22).

Sutton, R., & Barto, A. (1998a). *Reinforcement learning.* Boston, MA: MIT Press.

Sutton, R., & Barto, A. (1998b). *Reinforcement Learning.* MIT PRESS.

Sutton, R., McAllester, D., Singh, S., & Mansour, Y. (2000a). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, *12*(22).

Sutton, R., McAllester, D., Singh, S., & Mansour, Y. (2000b). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems 12.* Cambridge, MA: MIT Press.

Sutton, R., McAllester, D., Singh, S., & Mansour, Y. (2001). *Comparing policy gradient methods.* (Unfinished paper)

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction.* Cambridge: MIT Press.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*, 181–211.

T.Henzinger, & Toi, H. (1996). Linear phase-portrait approximations for nonlinear hybrid systems. In *Proceedings of the 1995 dimacs workshop on verification and control of hybrid systems.*

Tricomi, F. G. (1985). *Integral equations.* Dover Publications.

Tsitsiklis, J. N., & Roy, B. V. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, *42*(5), 674–690.

Udwadia, F. E. (2003). A new perspective on tracking control of nonlinear structural and mechanical systems. *Proc. R. Soc. Lond. A*, *2003*, 1783–1800.

Udwadia, F. E., & Kalaba, R. E. (1996). *Analytical dynamics: A new approach.* Cambridge University Press.

Wada, Y., & Kawato, M. (1994). Trajectory formation of arm movement by a neural network with forward and inverse dynamics models. *Systems and Computers in Japan*, *24*, 37–50.

Wada, Y., & Kawato, M. (1995). A theory for cursive handwriting based on the minimization principle. *Biological Cybernetics*, *73*(1), 3–13.

Weaver, L., & Tao, N. (2001a). The optimal reward baseline for gradient-based reinforcement learning. *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference*, *17*(29).

Weaver, L., & Tao, N. (2001b). The variance minimizing constant reward baseline for gradient-based reinforcement learning. *Technical Report ANU*, -(30).

Williams, R. (1986). *Reinforcement learning in connectionist networks: A mathematical analysis* (Technical Report No. ICS-8605). San Diego, CA: University of California, Institute for Cognitive Science.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(23).

Wolpert, D. M., & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, *11*(7-8), 1317–1329.

Xu, X., & Antsaklis, P. (2002). An approach to optimal control of switched systems with internally forced switchings. In *Proceedings of the american control conference* (pp. 148 –153). Anchorage, USA.

Yang, Z. (2001). *Introduction to hybrid systems - modeling and control* (Tech. Rep.). Aalborg University.

Yoshikawa, T. (1990). *Foundations of robotics: Analysis and control.* MIT Press.

Appendix A

Additional Derivations

As the rather long derivation of theorems required for the proofs in this thesis proposal would interrupt the flow in this appendix, we have collected them here. The theorems and discussions include (i) the solution of partitioned regression problem, and (ii) the Fisher information property of the All-Action matrix.

A.1 Partitioned Regression Problems

In this section, we derive a Theorem needed for the derivation of estimators which simplifies the estimation of the compatible function approximation.

Theorem 17 *A regression problem of the form*

$$\boldsymbol{\beta}^* = \begin{bmatrix}\boldsymbol{\beta}_1 & \boldsymbol{\beta}_2\end{bmatrix}^T = \operatorname*{argmin}_{\boldsymbol{\beta}} \left(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\right)^T \left(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\right), \tag{A.1}$$

with basis function $\mathbf{X} = \begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}$ *has the unique solution*

$$\boldsymbol{\beta}_1 = \left(\mathbf{X}_1^T\mathbf{X}_1\right)^{-1}\mathbf{X}_1^T\left(\mathbf{Y} - \mathbf{X}_2\mathbf{b}\right), \tag{A.2}$$

$$\boldsymbol{\beta}_2 = \mathbf{Q}^{-1}\mathbf{X}_2^T\left(\mathbf{Y} - \mathbf{X}_1\left(\mathbf{X}_1^T\mathbf{X}_1\right)^{-1}\mathbf{X}_1^T\mathbf{Y}\right), \tag{A.3}$$

with

$$\begin{aligned}\mathbf{Q}^{-1} &= \left(\mathbf{X}_2^T\mathbf{X}_2\right)^{-1} \\ &+ \left(\mathbf{X}_2^T\mathbf{X}_2\right)^{-1}\mathbf{X}_2^T\mathbf{X}_1\left(\mathbf{X}_1^T\mathbf{X}_1 - \mathbf{X}_1^T\mathbf{X}_2\left(\mathbf{X}_2^T\mathbf{X}_2\right)^{-1}\mathbf{X}_2^T\mathbf{X}_1\right)^{-1}\mathbf{X}_1^T\mathbf{X}_2\left(\mathbf{X}_2^T\mathbf{X}_2\right)^{-1}.\end{aligned} \tag{A.4}$$

Proof. The solution of the regression problem in Equation (A.1) is given by ■

$$\boldsymbol{\beta}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}, \tag{A.5}$$

see (Harville, 2000). By defining $\mathbf{T} = \mathbf{X}_1^T\mathbf{X}_1$, $\mathbf{U} = \mathbf{X}_1^T\mathbf{X}_2$, $\mathbf{W} = \mathbf{X}_2^T\mathbf{X}_2$, and subsequently applying the Matrix Inversion Theorem (see (Harville, 2000), pages 98–101), we obtain

$$\left(\mathbf{X}^T\mathbf{X}\right)^{-1} = \begin{bmatrix} \mathbf{T} & \mathbf{U} \\ \mathbf{U}^T & \mathbf{W} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{T}^{-1} + \mathbf{T}^{-1}\mathbf{U}\mathbf{Q}^{-1}\mathbf{U}^T\mathbf{T}^{-1} & -\mathbf{T}^{-1}\mathbf{U}\mathbf{Q}^{-1} \\ -\mathbf{Q}^{-1}\mathbf{U}^T\mathbf{T}^{-1} & \mathbf{Q}^{-1} \end{bmatrix}, \tag{A.6}$$

with $\mathbf{Q} = \mathbf{W} - \mathbf{U}^T\mathbf{T}^{-1}\mathbf{U}$. We can simplify $\mathbf{Q}^{-1}$ using the Sherman-Morrison Theorem (see (Moon & Stirling, 2000), pages 258–259) which yields

$$\mathbf{Q}^{-1} = \mathbf{W}^{-1} + \mathbf{W}^{-1}\mathbf{U}^T\left(\mathbf{T} - \mathbf{U}\mathbf{W}^{-1}\mathbf{U}^T\right)^{-1}\mathbf{U}\mathbf{W}^{-1}. \tag{A.7}$$

When multiplying $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ by $\mathbf{X}^T\mathbf{Y} = \left[\mathbf{X}_1^T\mathbf{Y} \quad \mathbf{X}_2^T\mathbf{Y}\right]^T$, we obtain

$$\begin{aligned} \boldsymbol{\beta}_1 &= \left(\mathbf{T}^{-1} + \mathbf{T}^{-1}\mathbf{U}\mathbf{Q}^{-1}\mathbf{U}^T\mathbf{T}^{-1}\right)\mathbf{X}_1^T\mathbf{Y} - \mathbf{T}^{-1}\mathbf{U}\mathbf{Q}^{-1}\mathbf{X}_2^T\mathbf{Y}, && \text{(A.8)} \\ &= \mathbf{T}^{-1}\left(\mathbf{X}_1^T\mathbf{Y} - \mathbf{U}\boldsymbol{\beta}_2\right), && \text{(A.9)} \\ \boldsymbol{\beta}_2 &= \mathbf{Q}^{-1}\left(\mathbf{X}_2^T\mathbf{Y} - \mathbf{U}^T\mathbf{T}^{-1}\mathbf{X}_1^T\mathbf{Y}\right). && \text{(A.10)} \end{aligned}$$

After inserting the definitions for $\mathbf{T}$, $\mathbf{U}$, and $\mathbf{W}$, we obtain Equations (A.2, A.3, A.2).

A.2 Fisher Information Property

In Section 3.98, we explained that the all-action matrix $F(\boldsymbol{\theta})$ equals in general the Fisher information matrix $G(\boldsymbol{\theta})$. In (Moon & Stirling, 2000), we can find the well-known lemma that by differentiating $\int_{\mathbb{R}^n} p(\boldsymbol{\xi})d\boldsymbol{\xi} = 1$ twice with respect to the parameters $\boldsymbol{\theta}$, we can obtain

$$\int_{\mathbb{R}^n} p(\boldsymbol{\xi})\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\xi})d\boldsymbol{\xi} = -\int_{\mathbb{R}^n} p(\boldsymbol{\xi})\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\xi})\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\xi})^T d\boldsymbol{\xi} \tag{A.11}$$

for any probability density function $p(\boldsymbol{\xi})$. Furthermore, we can rewrite the probability $p(\boldsymbol{\xi}_{0:n})$ of a history as

$$p\left(\boldsymbol{\xi}_{0:n}\right) = p\left(\mathbf{x}_0\right)\prod_{t=0}^{n} p\left(\mathbf{x}_{t+1}\left|\mathbf{x}_t, \mathbf{u}_t\right.\right)\pi\left(\mathbf{u}_t\left|\mathbf{x}_t\right.\right),$$

which implies that the log-policy derivatives are given by

$$\nabla^2_{\boldsymbol{\theta}} \log p\left(\boldsymbol{\tau}_{0:n}\right) = \sum_{t=0}^{n} \nabla^2_{\boldsymbol{\theta}} \log \pi\left(\mathbf{u}_t\left|\mathbf{x}_t\right.\right). \tag{A.12}$$

Using Equations (A.11), and the definition of the Fisher information matrix (Amari, 1998), we can determine Fisher information matrix for the average reward case in sample notation, i.e,

$$
\begin{aligned}
G(\boldsymbol{\theta}) &= \lim_{n\to\infty} \frac{1}{n} E_{\boldsymbol{\xi}_{0:n}} \left\{ \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p(\boldsymbol{\xi}_{0:n}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p(\boldsymbol{\xi}_{0:n})^T \right\}, \qquad \text{(A.13)}\\
&= -\lim_{n\to\infty} \frac{1}{n} E_{\boldsymbol{\xi}_{0:n}} \left\{ \boldsymbol{\nabla}^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\xi}_{0:n}) \right\},\\
&= -\lim_{n\to\infty} \frac{1}{n} E_{\boldsymbol{\xi}_{0:n}} \left\{ \sum_{t=0}^{n} \boldsymbol{\nabla}^2_{\boldsymbol{\theta}} \log \pi\left(\mathbf{u}_t \left| \mathbf{x}_t \right.\right) \right\},\\
&= -\int_{\mathbb{X}} d^{\pi}(\mathbf{x}) \int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}) \boldsymbol{\nabla}^2_{\boldsymbol{\theta}} \log \pi(\mathbf{u}|\mathbf{x}) d\mathbf{u} d\mathbf{x},\\
&= \int_{\mathbb{X}} d^{\pi}(\mathbf{x}) \int_{\mathbb{U}} \pi(\mathbf{u}|\mathbf{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\mathbf{u}|\mathbf{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\mathbf{u}|\mathbf{x})^T d\mathbf{u} d\mathbf{x},\\
&= F(\boldsymbol{\theta})
\end{aligned}
$$

This proves that the all-action matrix is indeed the Fisher information matrix for the average reward case. For the discounted case, with a discount factor γ we realize that we can rewrite the problem where the probability of rollout is given by $p_\gamma(\boldsymbol{\xi}_{0:n}) = p(\boldsymbol{\xi}_{0:n})(\sum_{i=0}^{n} \gamma^i \mathbb{I}_{x_i,u_i})$. It is straightforward to show that $\boldsymbol{\nabla}^2_{\boldsymbol{\theta}} \log p\left(\boldsymbol{\xi}_{0:n}\right) = \boldsymbol{\nabla}^2_{\boldsymbol{\theta}} \log p_\gamma(\boldsymbol{\xi}_{0:n})$, and derive that the all-action matrix equals the Fisher information matrix by the same kind of reasoning as in Eq.(A.13). Therefore, we can conclude that in general, i.e., $G(\boldsymbol{\theta}) = F(\boldsymbol{\theta})$.