Risk-Sensitive Policy Search with Applications to Robot-Badminton

Erlernen risikosensitiver Strategien mit Anwendungen an Robot-Badminton
Master-Thesis von David Nass aus Lahnstein
Dezember 2018

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Risk-Sensitive Policy Search with Applications to Robot-Badminton
Erlernen risikosensitiver Strategien mit Anwendungen an Robot-Badminton

Vorgelegte Master-Thesis von David Nass aus Lahnstein

1. Gutachten: Prof. Jan Peters, Boris Belousov
2. Gutachten: Prof. Dr.-Ing. Ulrich Konigorski

Tag der Einreichung:

Erklärung zur Abschlussarbeit gemäß § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, David Nass, die vorliegende Master-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Datum / Date:	Unterschrift / Signature:
______________________	____________________________________

Abstract

With the increasing pace of automation, modern robotic systems need to act in stochastic, non-stationary, partially observable environments. A range of algorithms for finding parameterized policies have been proposed in the past that optimize for long-term average performance. However, the majority of the proposed approaches does not explicitly take into account the variability of the performance metric, which may lead to finding policies that although performing well on average, can perform spectacularly bad in a particular run or period of time. To address this shortcoming, we study an approach to policy optimization that explicitly takes into account higher order statistics of the reward function. In this thesis, we extend policy gradient methods to include the entropic risk measure in the objective function and evaluate their performance on experiments from the field of finance and robot control.

Furthermore, we learn a robot badminton task. Badminton requires the robot to execute time critical, precise movements that need to be adapted constantly to new situations. We implement a ball prediction that can estimate a desired hitting point. The hitting movements are learned by kinesthetic teaching of a human instructor. The learned skills can generalize to new situations by adapting a small set of parameters without the need to relearn the complete movement. We apply the proposed method in a simulation environment and a real Barrett WAM.

Zusammenfassung

Mit der zunehmenden Automatisierung müssen moderne Robotersysteme in stochastischen, nicht stationären, teilweise beobachtbaren Umgebungen agieren. In der Vergangenheit wurde eine Reihe von Algorithmen zum erlernen parametrisierter Policies eingeführt, die den Durchschnittsnutzen maximieren. Die Mehrheit dieser Ansätze berücksichtigt jedoch nicht explizit die Variabilität des Nutzens, was dazu führen kann, dass Policies gefunden werden, die, obwohl sie im Durchschnitt gut sind, in einer bestimmten Sequenz oder in einem bestimmten Zeitraum außerordentlich schlecht abschneiden können. Um dieses Problem zu beheben, untersuchen wir einen Ansatz zur Optimierung von Policies, der explizit statistische Parameter höherer Ordnung in der Nutzenfunktion berücksichtigt. Innerhalb dieser Masterthesis erweitern wir Policy-Gradientenalgorithmen, um das entropische Risikomaß in die Optimierung einzubeziehen und bewerten diese an Experimenten aus dem Finanzbereich und der Robotersteuerung.

Außerdem bringen wir einem Roboterarm das Badminton spielen bei. Im Badminton muss der Roboter zeitkritische, präzise Bewegungen ausführen, die ständig an neue Situationen angepasst werden müssen. Wir implementieren eine Ballprädiktion, die den Punkt des Zusammentreffens von Ball und Schläger schätzen kann. Die Schlagbewegungen werden durch kinästhetische Ausführung eines menschlichen Trainers erlernt. Die erlernten Bewegungs-Primitive können sich an neue Situationen anpassen, indem eine kleine Menge von Parametern angepasst wird, ohne die gesamte Bewegung neu erlernen zu müssen. Wir wenden die vorgeschlagene Methode sowohl in einer Simulationsumgebung, als auch an einem echten Barrett WAM an.

Acknowledgments

First and foremost, I would like to thank my advisor Boris Belousov for the proposal of this topic and his excellent support. I highly appreciate his guidance throughout my work. With patience he steered me towards the right direction when I needed it and encouraged me with helpful comments and insights. I am also grateful to Prof. Dr. Jan Peters and Prof. Dr.-Ing. Ulrich Konigorski who provided me with the opportunity to work in such an inspiring research environment. Especially, the access to the laboratory and robots was appreciated.
My very profound gratitude goes to my parents, family and to my girlfriend for providing me with unfailing support throughout the process of researching and writing this thesis. This accomplishment would not have been possible without them.
Thank you.

Contents

Figures and Tables

List of Figures

List of Tables

Abbreviations, Symbols and Operators

List of Abbreviations

Notation	Description
Adam	adaptive moment estimation
CrKR	cost-regularized kernel regression
DMP	dynamic movement primitive
DOF	degree of freedom
EKF	extended Kalman filter
EM	expectation maximization
i.i.d.	independently and identically distributed
KF	Kalman filter
KL	Kullback-Leibler
MAE	mean absolute error
MDP	Markov decision process
MP	movement primitive
MSE	mean squared error
NPG	natural policy gradient
PG	policy gradient
PPO	proximal policy optimization
ProMP	probabilistic movement primitive
REPS	relative entropy policy search
RFF	random fourier features
RL	reinforcement learning
TRPO	trust region policy optimization

List of Symbols

Notation	Description
$\mathbf{a}$	Action
C	Cost
$\boldsymbol{\phi}(\mathbf{s})$	Transformation of states $\mathbf{s}$ to feature space
D_{KL}	Kullback-Leibler divergence
η	Lagrangian multiplier
μ	Mean of a Gaussian distribution
$\boldsymbol{\rho}$	Meta-parameters that adapt movement primitives to new situations
$\boldsymbol{\omega}$	Parameter vector of upper-level policy
$\boldsymbol{\theta}$	Parameter vector of lower-level policy
$\pi_{\boldsymbol{\omega}}$	Parametric policy corresponding to parameter vector $\boldsymbol{\omega}$
J	Performance measure
$\pi(\mathbf{a}\vert\mathbf{s})$	Probability of taking action $\mathbf{a}$ in state $\mathbf{s}$ under policy π
R	Reward
γ	Risk factor
$\mathbf{s}$	State
σ	Standard deviation of a Gaussian distribution

List of Operators

Notation	Description	Operator
$\vert\ \vert$	Absolute value	$\vert \bullet \vert$
$\mathrm{argmax}_{\boldsymbol{\omega}}$	Parameter vector $\boldsymbol{\omega}$ at which the function values are maximized	$\mathrm{argmax}_{\boldsymbol{\omega}}(\bullet)$
clip	Values outside the interval $[a, b]$ are clipped to the interval edges	$\mathrm{clip}(\bullet, a, b)$
$\nabla_{\boldsymbol{\omega}}$	Derivative with respect to parameter vector $\boldsymbol{\omega}$	$\nabla_{\boldsymbol{\omega}}(\bullet)$
$\mathbb{E}$	Expectation of samples	$\mathbb{E}(\bullet)$

log	the natural logarithm	$\log(\bullet)$
Var	Variance of samples	$\mathrm{Var}(\bullet)$

1 Introduction

The word robot was only introduced in 1920, by the Czech Karel Čapek [1]. However, the concept of artificial humans and automated machinery can be dated back to classical times and practical applications of robots did not grow until the 20th century. Nowadays, robots find their usage in many different fields, e.g. in industry, agriculture, medical science, military, etc. As early as 1965, Simon [2] claimed “machines will be capable, within twenty years, of doing any work a man can do”. Nevertheless, most of the robotic platforms are designed and limited to a specific task in well-defined environments. Many robots may outperform humans in specialized but limited applications, but they are still far away from having the diversity and complexity of skills as humans.

While the automation of processes in the industry, as well as in every day life, advances constantly, controllers for the robots are still mostly hand-crafted. Therefore, they are very limited to adapt to new situations. For optimal robot control the exact model of both the robot and its interacting environment are assumed to be known. Hence, performing actions in unknown or uncertain environments is challenging. An alternative to the hand-tuned controller could be a self-learning approach by using machine-learning methods. The focus of this thesis is going to be on reinforcement learning (RL) [3], where the considered task is learned by trial and error, while receiving rewards based on the performance. In recent years RL gained more interest. For example, the company DeepMind developed a self-learning system that uses a convolutional neural network combined with RL methods. Their systems are capable of playing games such as Go, Atari 2600 video games and chess with superhuman performance [4]. In contrast to the aforementioned examples, robotic control problems have to deal with high-dimensional continuous state and action spaces, making a lot of common RL-methods unfeasible. Policy Search proposes parameterized policies to reduce the search space of possible solutions. This is the preferred RL approach within this thesis.

It is easy to imagine that the safety of the robot and its (perhaps even human) surrounding must be the top priority. Thus, risk sensitivity might play a role to some extend in the learning of a new task. Risk can be interpreted as the predictability of an outcome, where a certain outcome is associated with low risk, while in contrast an uncertain outcome has a higher risk. Imagine a robot working hand in hand, together with a human. When the robot is near the human, it should limit itself to performing low risk movements that are not going to harm the human in any way. On the contrary, it is possible to think of situations, where a risky behavior is requested. For example during a badminton task.

Badminton is a highly challenging racket game played on an indoor court divided by a net. The players need to posses excellent motor coordination abilities, precision and speed, to outplay the opponent and to return any incoming shuttlecock. These requirements are challenging for robots. The robot needs to predict the location and timing of where and when to hit and return the shuttlecock to the opponents half of the court and plan a movement towards that point. Additionally, badminton players must make split second decisions depending on their own and the opponents position on the field to decide how and where to return the shuttlecock to. These decisions can be motivated with risk-sensitivity. A clear towards the back of the court is considered to be a save option, while the powerful smash to score a point is risky.

1.0.1 Contribution

This thesis contributes to the fields of reinforcement learning in robotics. Robotic systems need to act in stochastic, non-stationary, partially observable environments. A range of algorithms for finding parameterized policies have been proposed in the past that maximize the expected reward. However, these approaches do not take into account the variability or uncertainty of the performance measure. For stochastic robot control problems the objective should include, besides the maximization of the expected reward, some variance related criteria. Within this work, we pursue to extend state-of-the-art policy search algorithms to accommodate a risk-sensitive objective. The algorithms are evaluated on experiments motivated by the field of finance and a task closely related to badminton. This should offer insight if autonomous systems can be optimized not only for average performance, but also considering variance related criteria. During the thesis the exploration-exploitation trade-off will be closely examined.

Further, we develop an approach to teach a Barrett WAM to solve a badminton task. A ball prediction is implemented that can estimate the hitting point. We select a concept, where we can learn the hitting movement by imitation learning. This results in a movement primitive that can be adapted by a small number of meta-parameters to fit new situations. During learning, only the meta-parameters need to be learned to hit an incoming ball successfully. The learning is done in a simulation environment under near perfect conditions. Nevertheless, the learned task is transfered to a real Barrett WAM.

1.0.2 Outline

The thesis is structured as follows:

- Chapter 2 gives an introduction to the mathematical framework of RL, using the Markov decision process (MDP). To deal with the curse of dimensionality in a robot learning problem, popular policy search methods are introduced. Afterwards, we discuss two movement primitive frameworks that enable generalization to new situations and can be learned by demonstrations. We conclude the chapter with an explanation of the extended Kalman filter (EKF), that will later be used to handle the ball prediction in the robot-badminton task.

- Chapter 3 introduces the reader to risk-measures and risk-sensitive optimization in policy search. We adapt policy gradient methods to find both optimistic and pessimistic policies and discuss what effects a variance related criterion has on the exploration of the action space. Further, we investigate the risk-sensitivity of relative entropy policy search (REPS).

- In Chapter 4 the algorithms, which are discussed and introduced in Chapter 3, are evaluated and verified by experiments. The first experiment is motivated by the field of finance, where the risk of investment strategies is a constant companion. The second experiment is aimed to bring the goal of learning a badminton task closer. It is related to problems usually encountered in robot learning and control and thus it can be investigated, whether a risk-sensitive objective is beneficial in similar problems.

- Chapter 5 presents our approach to learn a robot-badminton task with policy search methods. The learning from demonstration and how the robot skills can be generalized to new situations is explained. Moreover, a ball prediction with an EKF is implemented.

- Chapter 6 concludes the thesis. The results obtained from this work are summarized and discussed. We suggest interesting research direction for possible future thesis.

2 Foundations

Within this chapter, we lay the mathematical foundations for the thesis. First, a brief definition and formulation of reinforcement learning problems with Markov decision processes is given. Then, the reader will be introduced to policy search methods. Following, is a description of two movement primitive frameworks and a summary about the extended Kalman filter.

2.1 Reinforcement Learning

Reinforcement learning (RL) is a subfield of machine learning that focuses on learning optimal behavior within a dynamical system while maximizing a certain reward criterion [3]. The learner is most often called the agent and interacts with its environment via actions. Unlike in a supervised learning scenario, the best case solutions are not presented to the agent. The actions are rather chosen based on the observed rewards and iteratively improved until convergence. Thus, the agent can learn optimal behavior through trial and error without the need to have a perfect knowledge over its environment. A RL problem can be formally modeled as a Markov decision process (MDP) [3], which consists of:

- a set of environment and/or agent states $\mathbf{x} \in S$
- a set of actions $\mathbf{a} \in A$ of the agent
- the transition dynamics $\mathscr{P}(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t)$, satisfying the Markov property that defines the probability of transition from state $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$
- the immediate reward $r(\mathbf{x}, \mathbf{a})$
- the discount factor $\gamma \in [0, 1]$

The Markov property defines that the probability of transition from state $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$ must always satisfy $\mathscr{P}(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t, \mathbf{x}_{t-1}, \mathbf{a}_{t-1}, ...) = \mathscr{P}(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t)$, where the next step depends only on the current state and action but not on previous states or actions. The agent can choose actions that are dependent on the state based on some policy $\pi(\mathbf{a}|\mathbf{x})$. This policy defines the probability of taking action $\mathbf{a}$ given state $\mathbf{x}$. The goal is to find a policy that will maximize the expected cumulative reward [5]:

$$J_\pi = \mathbb{E}_{p_\pi}\left[r_T(\mathbf{x}_t) + \sum_{t=0}^{T-1} r(\mathbf{x}_t, \mathbf{a}_t) \right] = \mathbb{E}_{p_\pi}\left[R(\boldsymbol{\tau})\right] = \int R(\boldsymbol{\tau}) p_\pi(\boldsymbol{\tau}) d\boldsymbol{\tau}$$

The variable T is the number of time-steps in one episode and $\boldsymbol{\tau} = (\mathbf{x}_0, \mathbf{a}_0, \mathbf{x}_1, \mathbf{a}_1, ...)$ represents the trajectory of actions and states. In some cases an infinite-horizon case is considered, where the discount factor γ regularizes the trade-off between long-term and short-term reward:

$$J_\pi = \mathbb{E}_{p_\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{x}_t, \mathbf{a}_t) \right] = \mathbb{E}_{p_\pi}\left[R(\boldsymbol{\tau})\right] = \int R(\boldsymbol{\tau}) p_\pi(\boldsymbol{\tau}) d\boldsymbol{\tau}$$

Here, the discount factor needs to be $\gamma \in [0, 1)$ to guarantee convergence. In both cases the trajectory distribution $p_\pi(\boldsymbol{\tau})$ of the policy π is given by:

$$p_\pi(\boldsymbol{\tau}) = \mathscr{P}(\mathbf{x}_0) \prod_{t=0}^{T-1} \mathscr{P}(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t) \pi(\mathbf{a}_t|\mathbf{x}_t)$$

Having shown how a general RL problem can be formalized, the next section will contain an introduction on how to obtain optimal policies using policy search methods. It should be noted that other approaches exist to tackle RL problems, namely value function methods like policy iteration or value iteration [3]. These approaches however require the exploration of the entire state-action space and are therefore impractical for continuous states and actions.

2.2 Policy Search Methods

When working in robotics, one major problem arises that is known as the "curse of dimensionality"[6]. When working in continuous state and action spaces, the number of data to fill up the action-state-space becomes unfeasibly high. One possible solution to this problem is to use parameterized policies $\pi_{\boldsymbol{\theta}}$, which reduces the search space of possible solutions to the parameter space. These policies can be learned by Policy Search Methods [5].
The focus of this thesis is on model-free policy search, where the policy is updated solely on the sampled trajectories and observed rewards, without any knowledge of system parameters or dynamics. A policy search method is divided into three parts [5]: the policy exploration step, the policy evaluation step, and the policy update step.
During the exploration step, multiple trajectories using the policy $\pi_{\boldsymbol{\theta}}$, given the parameters $\boldsymbol{\theta}$, are generated. The exploration strategy can be step-based, that explore new actions at each time step, or episode-based, where the exploration is only added at the start of each episode. Within the scope of this thesis only episode-based exploration strategies are considered.
Subsequently, a policy evaluation step is performed, in which the quality of the sampled trajectories are assessed by some reward measure. It is possible to distinguish between step and episode based evaluation, where either a reward is computed for each time-step or at the end of the episode.
Based on the results of the evaluation-step, a new set of parameters, and thus a new policy, can be calculated during the policy update step. Different strategies rely on gradient methods, expectation maximization and information theoretic approaches [5]. All the strategies aim to maximize the expected return given the parameters of the policy:

$$J_{\boldsymbol{\theta}} = \mathbb{E}_{p_{\boldsymbol{\theta}}}[R(\tau)] = \int p_{\boldsymbol{\theta}}(\tau) R(\tau) d\tau$$

The exploration step presupposes that the policy is stochastic and has some variance. Thus, $\pi_{\boldsymbol{\theta}} = \pi(\mathbf{a}|\mathbf{x}, \boldsymbol{\theta})$ defines the probability of taking action $\mathbf{a}$ given the states $\mathbf{x}$ and parameters $\boldsymbol{\theta}$. Policy search methods also allow for using and learning of movement primitives as policies. See Section 2.3 for the definition and further insight.
As already mentioned, the thesis is mainly going to discuss an episodic scenario. In this case, it is useful to define a stochastic upper level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ that selects the parameters $\boldsymbol{\theta}$ of a deterministic lower control policy $\pi_{\boldsymbol{\theta}}(\mathbf{x}, t)$. This approach leads to the following optimization problem:

$$J_{\boldsymbol{\omega}} = \int \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) \int p_{\boldsymbol{\theta}}(\tau) R(\tau) d\tau d\boldsymbol{\theta} = \int \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta} = \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[R(\boldsymbol{\theta})] \tag{2.1}$$

Furthermore, the above formulation can be extended to learn solutions for multiple situations. A situation is defined by a context vector $\mathbf{s}$. The context vector $\mathbf{s}$ contains all the environment and task specific variables that change from one situation to another. To generalize the policies, the upper-level policy can be defined as $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s})$, i.e. the probability of parameters $\boldsymbol{\theta}$ of the lower-level policy given the context $\mathbf{s}$. The optimization problem is extended to maximizing the expected return over all contexts

$$J_{\boldsymbol{\omega}} = \int \mu(\mathbf{s}) \int \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s}) R(\boldsymbol{\theta}, \mathbf{s}) d\boldsymbol{\theta} d\mathbf{s}, \tag{2.2}$$

where $\mu(\mathbf{s})$ is the distribution over the context $\mathbf{s}$ and $R(\boldsymbol{\theta}, \mathbf{s})$ is the expected return of using the policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s})$ [5].
In the next sections commonly used policies and multiple algorithms to perform the policy update step are introduced.

2.2.1 Policies

A suitable choice of parameterized upper-level policy is a linear Gaussian policy:

$$\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) = \pi(\boldsymbol{\theta}|\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}) \tag{2.3}$$

Both $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ are learnable parameters and part of $\boldsymbol{\omega} = \{\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\}$ and define the mean and variance of the distribution. The Gaussian policy can be extended to multiple contexts by including a mapping of the context $\mathbf{s}$ to a feature space $\boldsymbol{\phi}(\mathbf{s})$:

$$\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s}) = \pi(\boldsymbol{\theta}|\mathbf{s}, \boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{M}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{s}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}) \tag{2.4}$$

Here, the learnable parameters consist of $\boldsymbol{\omega} = \{\mathbf{M}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\}$ and the mean depends linearly on $\mathbf{M}$.

Random Fourier Features

Within the framework of this thesis random fourier features (RFF) are used exclusively for feature parameterization. RFF are motivated by constructing low-dimensional representations to approximate a kernel and were introduced in [7]. These approximations are very useful to accelerate calculations done with the kernel matrix. Further insight and the exact derivation can be found in the cited literature.
In the scope of this work, a slightly different formulation is used that was inspired by Rajeswaran et al. [8]. The feature vector $\boldsymbol{\phi} \in \mathbb{R}^{\mathrm{N}}$, with N being the number of features, is defined as

$$\phi_i(\mathbf{s}) = \cos\left(\frac{\sum_j^d W_{i,j} s_j}{\nu} + \varphi_i\right), \tag{2.5}$$

where $\mathbf{W}$ is a matrix with each element drawn from a normal distribution $W_{i,j} \sim \mathcal{N}(0,1)$, d corresponds to the dimension of states and $i = 1, ..., N$. The variable ν can be interpreted as a bandwidth parameter and changes the frequency of the cosine function. The phase shift can be influenced by φ and is uniformly distributed in the interval $[0, 2\pi)$. It proved to be useful to have the first feature $\phi_0 = 1$, by setting $\mathbf{W}_0 = \mathbf{0}$ and $\varphi_0 = 0$, to include a bias. The only parameter that is open for tuning is the bandwidth ν. It is suggested to set ν to the average pairwise distances between state vectors [8]. In contrast to radial basis function features [3], that often need strenuous manual tuning, RFFs often led to satisfying results, without the need to excessively tune multiple parameters.

2.2.2 Policy Gradient

As the name policy gradient (PG) suggests, these methods aim to maximize the expected return $J_{\boldsymbol{\omega}}$ with a gradient update step. As the gradient $\nabla_{\boldsymbol{\omega}}$ points in the direction of steepest ascent, the update of the parameters can be performed as follows [5]

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \alpha \nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}},$$

where α is the step-size that needs to be defined.
To compute $\nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}}$ the “likelihood ratio trick”, $\nabla_x f(x) = f(x)\nabla_x \log f(x)$, can be applied:

$$\begin{aligned}\nabla_{\boldsymbol{\omega}} \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[R(\boldsymbol{\theta})] &= \nabla_{\boldsymbol{\omega}} \int \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta} = \int \nabla_{\boldsymbol{\omega}} \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &= \int \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta} = \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) R(\boldsymbol{\theta})]\end{aligned} \tag{2.6}$$

For Gaussian policies (see Section 2.2.1) the gradient w.r.t $\boldsymbol{\omega}$ is given in the Appendix A.1. The expectation in Equation (2.6) can be estimated by sampling N parameters $\boldsymbol{\theta}$ from the upper-level policy and evaluating their return, thus leading to

$$\nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}} = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}_i) R(\boldsymbol{\theta}_i). \tag{2.7}$$

Equation (2.7) may be used to generate an estimator for the gradient by sampling from the current policy. The gradient approach can be further modified by replacing the gradient with the natural gradient [3], which will later in this section be discussed in greater detail.

Adaptive Step-size

In practice it is often challenging to tune the step-size α to receive optimal performance. A step-size that is set too large can overshoot the optimum, lead to fluctuations around the minimum or even diverge, while a step-size that is too small causes a slow convergence rate. Another problem arises because the learning rate is set equally for all parameters and could hinder performance for sparse gradients. Lastly, gradient ascent is know to behave poorly at saddle-points as the gradients are close to zero.
Adaptive moment estimation (Adam) is a stochastic optimization algorithm that tackles the above issues [9]. It calculates adaptive step-sizes for all parameters and stores an exponential decaying moving average of the first and second moment - the mean and uncentered variance - of the gradients. These moving averages are calculated as follows:

$$\begin{aligned}\mathbf{m}_k &= \beta_1 \mathbf{m}_{k-1} + (1-\beta_1) \nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}} \\ \mathbf{v}_k &= \beta_2 \mathbf{v}_{k-1} + (1-\beta_2) (\nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}})^2 .\end{aligned} \tag{2.8}$$

The parameters β_1 and β_2 tune the decay rate, $\mathbf{m}_k$ is the estimate of the first moment and $\mathbf{v}_k$ the estimate of the second moment at iteration k. In practice the estimates are initialized with $\mathbf{0}$ and are therefore biased towards zeros. Thus, bias-corrected $\hat{\mathbf{m}}_k$ and $\hat{\mathbf{v}}_k$ are calculated:

$$\hat{\mathbf{m}}_k = \frac{\mathbf{m}_k}{1-\beta_1^k} \tag{2.9}$$

$$\hat{\mathbf{v}}_k = \frac{\mathbf{v}_k}{1-\beta_2^k} \tag{2.10}$$

The parameter update step is given by

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_{k+1} + \frac{\alpha}{\sqrt{\hat{\mathbf{v}}_k}+\epsilon}\hat{\mathbf{m}}_k, \tag{2.11}$$

where α is a new step-size parameter and ϵ ensures the denominator is never zero. Typical values, as suggested by the author of [9] are $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$. The step-size may vary from one problem to another but a value of $\alpha = 0.001$ proved to be a good reference [9]. Adam was in use for all the carried out experiments that update the policy with a stochastic gradient ascent.

Baselines

The collected rewards $R(\boldsymbol{\theta})$ are sampled from a stochastic policy. Hence, the evaluated rewards often have a high variance resulting in gradients that are affected by noise. To reduce the variance in the gradient estimates, it is suggested to subtract a baseline form the rewards [3].

$$\nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}} = \mathbb{E}_{\pi_{\boldsymbol{\omega}}}\left[\nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})(R(\boldsymbol{\theta}) - b)\right] \tag{2.12}$$

It is straight forward to show that a baseline does not include a bias into the gradient:

$$\mathbb{E}_{\pi_{\boldsymbol{\omega}}}\left[-\nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) b\right] = -b\nabla_{\boldsymbol{\omega}} \int \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} = 0 \tag{2.13}$$

A good first choice for a baseline is the average reward but it is possible to calculate optimal baselines that minimize the variance of the gradient [5]. Optimal baselines for the algorithms can be found in the corresponding literature.

Natural Policy Gradient

The natural policy gradient (NPG) [10] is based on the intuition that small changes in $\boldsymbol{\omega}$ could lead to big changes in the policy $\pi_{\boldsymbol{\omega}}$, which consequently could lead to a worse performance of the learning process or in the worst case to unstable behavior. To ensure stable learning, the distance between $\pi_{\boldsymbol{\omega}_k}$ and the new updated distribution $\pi_{\boldsymbol{\omega}_{k+1}}$ is limited. A suitable measure for the distance of probability distributions is the Kullback-Leibler (KL) divergence [5]:

$$D_{\mathrm{KL}}(p||q) = \int p(x) \log\left(\frac{p(x)}{q(x)}\right)\mathrm{d}x,$$

with the probability densities p and q. For sufficiently small $\partial\boldsymbol{\omega}$, the KL divergence of $\pi_{\boldsymbol{\omega}_k}$ and $\pi_{\boldsymbol{\omega}_{k+1}}$ can be approximated with the Fisher information matrix $\mathbf{F}_{\boldsymbol{\omega}}$ [10]:

$$\begin{aligned} &D_{\mathrm{KL}}(\pi_{\boldsymbol{\omega}}||\pi_{\boldsymbol{\omega}+\partial\boldsymbol{\omega}}) \approx \partial\boldsymbol{\omega}^{\mathrm{T}}\mathbf{F}_{\boldsymbol{\omega}}\partial\boldsymbol{\omega}, \\ \text{with}\quad &\mathbf{F}_{\boldsymbol{\omega}} = \mathbb{E}_{\pi_{\boldsymbol{\omega}}}\left[\nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})^{\mathrm{T}}\right]. \end{aligned} \tag{2.14}$$

The following optimization problem can be established to find the natural gradient [11]:

$$\begin{aligned} \nabla_{\boldsymbol{\omega}}^{\mathrm{NG}} J_{\boldsymbol{\omega}} &= \underset{\boldsymbol{\omega}}{\operatorname{argmax}} \partial\boldsymbol{\omega}^{\mathrm{T}}\nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}} \\ \text{s.t.}:\quad & D_{\mathrm{KL}}(\pi_{\boldsymbol{\omega}}||\pi_{\boldsymbol{\omega}+\partial\boldsymbol{\omega}}) \approx \partial\boldsymbol{\omega}^{\mathrm{T}} F_{\boldsymbol{\omega}} \partial\boldsymbol{\omega} < \epsilon \end{aligned} \tag{2.15}$$

which gives the solution:

$$\nabla_{\boldsymbol{\omega}}^{\mathrm{NG}} J_{\boldsymbol{\omega}} \propto F_{\boldsymbol{\omega}}^{-1}\nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}} \tag{2.16}$$

As the multiplication of the gradient with the inverse of the Fisher information matrix is a linear transformation, the natural gradient is invariant to linear transformations of the parameters of the distribution [5].

2.2.3 Relative Entropy Policy Search

Relative entropy policy search (REPS) is an information-theoretic approach. This class of methods follow the idea of restricting how far a new policy distribution can differ from the old. Like in the NPG, the KL divergence is used as a difference measure between policies. REPS however is an expectation maximization (EM) method, where the parameter update is performed with a weighted maximum likelihood estimate [12]. Closed form solutions can be obtained in the Appendix A.2, for the Gaussian policies used in this thesis.
The following optimization problem for REPS can be established:

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & J(\pi(\boldsymbol{\theta})) = \int \pi(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ \text{subject to} \quad & \epsilon \geq D_{\mathrm{KL}}(\pi(\boldsymbol{\theta}) || \mathbf{q}(\boldsymbol{\theta})) \\ & 1 = \int \pi(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \end{aligned} \tag{2.17}$$

The above states a constrained optimization problem, which can be solved with the Lagrangian multiplier method. For a detailed derivation the reader is referred to [12] and [5]. This leads to the closed form solution of the new policy update

$$\pi(\boldsymbol{\theta}) \propto \mathbf{q}(\boldsymbol{\theta}) \exp\left(\frac{R(\boldsymbol{\theta})}{\eta}\right) \tag{2.18}$$

where η is the Lagrangian multiplier corresponding to the KL-bound. The dual-function $g(\eta)$ must be minimized to obtain the parameter η.

$$\begin{aligned} \underset{\eta}{\text{minimize}} \quad & g(\eta) = \eta\epsilon + \eta \log \sum_i \frac{1}{N} \exp\left(\frac{R(\boldsymbol{\theta}_i)}{\eta}\right) \\ \text{subject to} \quad & \eta > 0. \end{aligned} \tag{2.19}$$

The dual function defines a lower bound on the optimal solution to (2.17) [13, p.:223]. Optimizing the (2.19) can under specific conditions be equivalent to solving the original problem. The dual function is convex in η and can be solved with any optimizer for constrained optimization problems [5].
The closed solution of $\pi(\boldsymbol{\theta})$ in (2.18) is only known from samples, because the explicit function $R(\boldsymbol{\theta})$ is usually unknown. In practice a parametric policy $\pi_{\boldsymbol{\omega}}$ is fitted to the samples obtained from (2.18). The parameters are calculated with a weighted maximum likelihood estimate, where the weights correspond to $d_i = \exp(R(\boldsymbol{\theta}_i/\eta))$ (see Appendix A.2).

2.2.4 Proximal Policy Optimization

Proximal policy optimization (PPO) is an algorithm that distinguishes itself from comparable algorithms such as trust region policy optimization (TRPO) (see [14] for insights), through its straightforward and efficient implementation and overall better performance. It uses an objective called the “surrogate” objective to update its parameters [15]. Unlike TRPO and REPS, the constraint on the policy update, bounded by the KL divergence, is formulated as a penalty in the objective function [15].

$$\underset{\boldsymbol{\omega}}{\text{maximize}} \quad J_{\boldsymbol{\omega}}^{\text{pen}} = \mathbb{E}\left[\frac{\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})}{\pi_{\boldsymbol{\omega}_{\text{old}}}(\boldsymbol{\theta})} \hat{A}(\boldsymbol{\theta})\right] - \beta\, \mathbb{E}\left[D_{\mathrm{KL}}(\pi_{\boldsymbol{\omega}_{\text{old}}}(\boldsymbol{\theta}) || \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}))\right], \tag{2.20}$$

where $\hat{A}(\boldsymbol{\theta})$ is the estimated advantage, the expected rewards minus some baseline. The parameter β controls the trust region of the parameter update and can be adjusted dynamically:

$$\begin{aligned} \text{if } \mathbb{E}\left[D_{\mathrm{KL}}(\pi_{\boldsymbol{\omega}_{\text{old}}}(\boldsymbol{\theta}) || \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}))\right] \geq 1.5\delta \rightarrow \beta_{k+1} = 2\beta_k \\ \text{else if } \mathbb{E}\left[D_{\mathrm{KL}}(\pi_{\boldsymbol{\omega}_{\text{old}}}(\boldsymbol{\theta}) || \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}))\right] \leq \frac{\delta}{1.5} \rightarrow \beta_{k+1} = \frac{\beta_k}{2} \end{aligned} \tag{2.21}$$

Another suggested objective by the authors of PPO is the clipped surrogate objective that manages without computing the KL [15]:

$$\underset{\boldsymbol{\omega}}{\text{maximize}} \quad J_{\boldsymbol{\omega}}^{\text{clip}} = \mathbb{E}\left[\min\left(\frac{\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})}{\pi_{\boldsymbol{\omega}_{\text{old}}}(\boldsymbol{\theta})} \hat{A}(\boldsymbol{\theta}), \text{clip}\left[\frac{\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})}{\pi_{\boldsymbol{\omega}_{\text{old}}}(\boldsymbol{\theta})}, 1-\epsilon, 1+\epsilon\right] \hat{A}(\boldsymbol{\theta})\right)\right] \tag{2.22}$$

where ϵ is a hyperparameter with a suggested value of 0.2. The clipping term behaves similar to a penalty on the KL divergence between policy updates. It clips the objective to prevent it from growing once the new policy steps too far from the old, thus preventing an update that is too greedy.
No matter what objective is used, the resulting parameter update is performed with stochastic gradient descent:

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \alpha \nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}}^{\text{clip|pen}} \tag{2.23}$$

The presented objectives allow multiple gradient update steps per data sample, because overly greedy updates of parameters are prohibited. This makes the algorithm a lot more sample efficient compared to the PG and NPG [15].

2.3 Movement Primitives

When teaching a robot a new skill, it is often useful to represent the task as a set of movements. These movements should be able to be combined and blended in order to form more complex tasks, to generalize to new situations and to be temporal scalable [16]. The goal is to find a task specific control policy for every possible state that transfers the robot to the next state [17], $\mathbf{u} = \pi(\mathbf{x}, t)$.
As it is infeasible to visit every space in a continuous setting, function approximations become crucial to lower the dimensionality of the movement primitives. One possible solution is to use basis functions. These approximations generalize neighboring states, to reduce the complexity of the problem to make learning of the policy π computational tractable.
The movement primitives (MPs) should possess some specific properties that are summarized in the following [16].

- Concise representation:
  The number of parameters in a MP representation should be kept to a minimum. Fewer parameters facilitate learning from demonstration and RL.

- Adaptation and time modulation:
  To guarantee generalization to new situations, the MP framework must be able to adapt its final desired positions and velocities. Some MPs-frameworks also offer to set via points (see Section 2.3.2). In certain tasks the correct timing is crucial and the duration of the movement has to be modified by implementing time modulation.

- Combination and sequencing:
  Combination of multiple MPs can enhance the diversity of applications by forming more complex tasks. Another approach to generate more complicated movements, is to play multiple MPs in a row. For efficient sequencing, smooth transitions between trajectories are obligatory.

- Coupling the degrees of freedom (DOFs):
  The multiple DOFs, if present, need to be synchronized to reproduce the demonstration correctly. In many approaches time is used to perform the synchronization. In Section 2.3.2 a framework is introduced that captures the correlation between multiple robot joints and thus creating an additional synchronization method.

- Optimal Behavior:
  Preferably the trajectories given by a MP is the optimal solution to a task. Especially when the system is stochastic, a single trajectory cannot guarantee optimality.

- Stability:
  It is of high priority to generate stable movements from MPs. Under no circumstances should a policy lead to unstable behavior.

There are a variety of different movement primitive frameworks [16][18][19][20], all of which have different advantages and best use-case scenarios but very rarely offer all of the desired properties. In this work two different approaches to represent robot movements - dynamic movement primitives (DMPs) and probabilistic movement primitives (ProMPs) - will be described.

2.3.1 Dynamic Movement Primitives

DMPs were first introduced by Stefan Schaal [20] and further improved by Ijspeert and Kober in [17] and [21].

The idea is to use a second order dynamical system that is well understood and for which stability can be proven easily. With the help of an additional, nonlinear term the desired behavior can be forced upon the system. In the original paper the system was formulated as a first order differential equation:

$$\begin{aligned}\frac{1}{\tau}\dot{x} &= \alpha(\beta(g-y)-x)+f \\ \frac{1}{\tau}\dot{y} &= x \end{aligned} \tag{2.24}$$

For simplicity and clarity the second order system will be used throughout the thesis:

$$\ddot{y} = \tau^2\left(\alpha\left(\beta(g-y)-\frac{\dot{y}}{\tau}\right)+af\right) \tag{2.25}$$

The state y describes the state variable of one DOF, for example the joint angle q of a robot. The parameters α and β have to be chosen to critically damp the system in order to converge toward the goal g. The system can be temporally scaled with the parameter τ. The term f is the nonlinear forcing function that together with a scaling parameter a alters the output of the system to a desired shape.
In [17] a phase variable z as a replacement of time was introduced that has the following first-order linear dynamics:

$$\dot{z} = -\frac{1}{\tau}\alpha_z z$$

This system starts at $z(0) = 1$, uses the same time constant τ and the parameter α_z is set such that $z(T) \approx 0$, where T is the length of the movement.
The forcing function consists of N weighted basis functions and is given by

$$f(z) = \frac{\sum_{i=1}^{N}\psi_i(z)\omega_i}{\sum_{i=1}^{N}\psi_i(z)}z.$$

The learnable parameters are the weights $\boldsymbol{\omega} = [\omega_1 \dots \omega_N]^{\mathrm{T}}$ and $\boldsymbol{\psi}(z) = [\psi(z)_1 \dots \psi(i)_N]^{\mathrm{T}}$ is the vector consisting of all N radial basis function that are defined as follows

$$\psi_i(z) = \exp\left(-\frac{1}{2\sigma_i^2}(z-c_i)^2\right) = \exp\left(-h_i(z-c_i)^2\right),$$

where h_i is the bandwidth of the basis function and c_i its center. In order to guarantee stability of the dynamical system, f has to vanish for $t \rightarrow T$ which is the case if $z \rightarrow 0$ for $t = T$.
The factor a is often set to $a = (g - y_0)$ and scales the forcing function to maintain the overall shape of the trajectory while reaching different goals [17].
When using Dynamic Movement Primitives to learn from demonstration, Equation (2.25) can be rearranged to form a function approximation problem:

$$f_{\text{des}} = \frac{1}{a}\left(\frac{\ddot{y}_{\text{des}}}{\tau^2} - \alpha(\beta(g-y_{\text{des}}) - \frac{\dot{y}_{\text{des}}}{\tau})\right) \tag{2.26}$$

Equation (2.26) lets us compute a desired forcing function for a given trajectory. If we want to fit a DMP to the desired trajectory, we need to calculate f_{des} for every time-step and find the weight vector $\boldsymbol{\omega}$ that results in the forcing function $\mathbf{f} = \boldsymbol{\Psi}\boldsymbol{\omega}$, which is as close as possible to $\mathbf{f}_{\text{des}}$. We will define the following:

$$\begin{aligned}\boldsymbol{\psi}(z) &= \begin{bmatrix}\psi_1 & \psi_2 & \dots & \psi_N\end{bmatrix}^T \\ \boldsymbol{\Psi} &= \begin{bmatrix}\frac{\psi^{\mathrm{T}}(z(0))}{\sum_{i=1}^{N}\psi_i(z(0))}z(0) & \dots & \frac{\psi^{\mathrm{T}}(z(T))}{\sum_{i=1}^{N}\psi_i(z(T))}z(T)\end{bmatrix}^{\mathrm{T}} \\ \mathbf{f}_{\text{des}} &= \begin{bmatrix}f_{\text{des}}(0) & \dots & f_{\text{des}}(T)\end{bmatrix}^{\mathrm{T}}\end{aligned}$$

Thus, solving the following linear least squares problem gives us the desired solution:

$$\begin{aligned}\boldsymbol{\omega} &= \underset{\boldsymbol{\omega}}{\operatorname{argmin}}\frac{1}{2}\left(\mathbf{f}_{\text{des}}(z)-\boldsymbol{\Psi}\boldsymbol{\omega}\right)^{\mathrm{T}}\left(\mathbf{f}_{\text{des}}(z)-\boldsymbol{\Psi}\boldsymbol{\omega}\right) \\ \boldsymbol{\omega} &= (\boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^{\mathrm{T}}\mathbf{f}_{\mathbf{des}}\end{aligned} \tag{2.27}$$

The current formulation of DMP has the drawback of only being able to set a goal position. For hitting movement it is necessary to set the velocity at the goal position as well. Kober et al. introduced a modification to the original DMP in [21].
The dynamical system in Equation (2.25) is adapted to set a desired final velocity $\dot{g}$ and includes a moving goal g_m:

$$\begin{aligned} \ddot{y} =& (1-z)\tau^2 \left(\alpha \left(\beta (g_m - y) - \frac{\dot{g}-\dot{y}}{\tau} \right) + af \right) \\ g_m =& g_m^0 - \dot{g}\frac{\ln(z)}{\tau \alpha_h} \\ g_m^0 =& g - \dot{g}\tau \end{aligned} \tag{2.28}$$

The parameter α_h needs to ensure that $g_m(T) = g$.
The extension to multiple (n) degrees of freedom is straight forward as each DOF is modeled independently. Equation (2.28) becomes

$$\begin{aligned} \ddot{\mathbf{y}} =& (1-z)\tau^2 \left(\alpha \left(\beta (\mathbf{g}_m - \mathbf{y}) - \frac{\dot{\mathbf{g}}-\dot{\mathbf{y}}}{\tau} \right) + \mathbf{Af} \right) \\ \mathbf{g}_m =& \mathbf{g}_m^0 - \dot{\mathbf{g}}\frac{\ln(z)}{\tau \alpha_h} \\ \mathbf{g}_m^0 =& \mathbf{g} - \dot{\mathbf{g}}\tau, \end{aligned} \tag{2.29}$$

where $\mathbf{A} = \mathrm{diag}(a_1, a_2, ..., a_n)$ and the vectors $\mathbf{g} = [g_1, ..., g_n]^\mathrm{T}$ and $\mathbf{y} = [y_1, ..., y_n]^\mathrm{T}$ include the goals and states of all DOFs. Learning from demonstration can be done in a similar manner as in the one dimensional case.
DMPs therefore offer a concise representation of movements, are able to be time modulated, offer stability and are able to be learned by one demonstration. However, combination of multiple primitives is not intended and not possible in the original formulation. In addition, the coupling of multiple DOFs only comes from the time synchronization of the phase variable and in a stochastic system optimal behavior cannot be guaranteed.

2.3.2 Probabilistic Movement Primitives

Recently a novel framework for representing movements has been developed by Paraschos et al. [22]. This frameworks attempts to tackle the disadvantages of DMPs while still incorporating its main advantages such as phase variables and the use of basis functions.
Firstly considered is the case of a single DOF of one joint angle q. A single movement is written as a trajectory $\tau = \{y_t\}_{t=0...T}$. As the MP of this framework describes different possibilities to execute the movement, a distribution over the trajectories is defined. The distribution is dependent on the weighting vector $\boldsymbol{\omega}$, which is used to represent the trajectory in a condensed form [22]:

$$p(\boldsymbol{\tau}|\boldsymbol{\omega}) = \prod_t \mathcal{N}(\mathbf{y}_t|\boldsymbol{\Phi}_t\boldsymbol{\omega}, \boldsymbol{\Sigma}_y), \tag{2.30}$$

with

$$\mathbf{y}_t = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = \boldsymbol{\Phi}_t\boldsymbol{\omega} + \boldsymbol{\epsilon}_y,$$

where q_t and $\dot{q}_t$ are the joint positions and velocities at time t and $\boldsymbol{\Phi}_t = \left[\boldsymbol{\phi}_\mathbf{t}\ \dot{\boldsymbol{\phi}}_\mathbf{t}\right]^\mathrm{T}$ are the time dependent basis functions with the corresponding derivatives. The $\epsilon_y \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_y)$ is observation noise and represents zero-mean independently and identically distributed (i.i.d.) Gaussian noise.
As the variance of the trajectories should be captured as well, the weight vector $\boldsymbol{\omega}$ will be represented by the distribution $p(\boldsymbol{\omega}; \boldsymbol{\theta})$. For simplicity this will be a Gaussian distribution with the parameter vector $\boldsymbol{\theta} = \{\boldsymbol{\mu_\omega}, \boldsymbol{\Sigma_\omega}\}$.
To gain a variable execution speed of the movement, a phase variable z_t is introduced. At the beginning, $t = 0$, $z_0 = 0$ and at the end $z_T = 1$. The velocity of the phase can be chosen as a constant, $\dot{z}_t = 1/T$. The basis functions $\boldsymbol{\phi}_t$ are no longer time dependent but instead depend on z_t, which leads to:

$$\begin{aligned} \boldsymbol{\phi}_t &= \boldsymbol{\phi}(z_t) \\ \dot{\boldsymbol{\phi}}_t &= \dot{\boldsymbol{\phi}}(z_t)\dot{z}_t \end{aligned}$$

Similarly to the DMPs, Gaussian basis functions are used for stroke-like movements. As rhythmic movements are not used in this thesis, the reader is referred to [16]. The basis functions are defined as follows:

$$
\begin{aligned}
b_i &= \exp\left(-\frac{1}{2\sigma_i^2}(z-c_i)^2\right) = \exp\left(-\frac{(z-c_i)^2}{2h_i}\right) \\
\phi_i &= \frac{b_i(z)}{\sum_{j=1}^{n} b_j(z)}
\end{aligned}
$$

The centers c_i of the ith basis function are uniformly spaced in the phase domain on the interval $[-2h, (1+2h)]$, where h is the width of each function [22].

Up until now, each degree of freedom was modeled separately and like in the DMP case no correlation between the joints can be captured. The above model can be easily extended to eliminate this disadvantage.

The weight vectors $\boldsymbol{\omega}_n$ of each degree of freedom n are combined to a weight vector $\boldsymbol{\omega} = [\boldsymbol{\omega}_1^{\mathrm{T}}, ..., \boldsymbol{\omega}_d^{\mathrm{T}}]^{\mathrm{T}}$. Furthermore, the basis function matrix $\boldsymbol{\Phi}_t$ is extended to a block matrix $\boldsymbol{\Psi}_t$. Thus, we can represent a trajectory for multiple degrees of freedom as

$$
p(\tau|\boldsymbol{\omega}) = \prod_t \mathcal{N}(\mathbf{Y}_t|\boldsymbol{\Psi}_t\boldsymbol{\omega}, \boldsymbol{\Sigma}_y), \tag{2.31}
$$

where

$$
\mathcal{N}(\mathbf{Y}_t|\boldsymbol{\Psi}_t\boldsymbol{\omega}, \boldsymbol{\Sigma}_y) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}_{1,t} \\ \vdots \\ \mathbf{y}_{d,t}\end{bmatrix} \middle| \begin{bmatrix}\boldsymbol{\Phi}_t & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{\Phi}_t\end{bmatrix} \boldsymbol{\omega}, \boldsymbol{\Sigma}_y\right).
$$

The vector $\mathbf{Y}_t$ contains the joint angles and velocities of all degrees of freedom. The distribution $p(\boldsymbol{\omega}|\boldsymbol{\theta})$ is over the combined parameter vectors and captures the correlation between multiple DOFs as well [22].

Other than DMPs described in the previous section, ProMPs need multiple demonstrations to learn the distribution $p(\boldsymbol{\omega};\boldsymbol{\theta})$. The distribution of the state given the parameters $\boldsymbol{\theta} = \boldsymbol{\mu}_{\boldsymbol{\omega}}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}}$ is

$$
\begin{aligned}
p(\mathbf{y}_t;\boldsymbol{\theta}) &= \int \mathcal{N}\left(\mathbf{y}_t|\boldsymbol{\Psi}_t\boldsymbol{\omega}, \boldsymbol{\Sigma}_y\right)\mathcal{N}\left(\boldsymbol{\omega}|\boldsymbol{\mu}_{\boldsymbol{\omega}}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}}\right)\mathrm{d}\boldsymbol{\omega} \\
&= \mathcal{N}\left(\mathbf{y}_t|\boldsymbol{\Psi}_t\boldsymbol{\mu}_{\boldsymbol{\omega}}, \boldsymbol{\Psi}_t\boldsymbol{\Sigma}_{\boldsymbol{\omega}}\boldsymbol{\Psi}_t^{\mathrm{T}} + \boldsymbol{\Sigma}_y\right).
\end{aligned} \tag{2.32}
$$

The learning of the weight parameters $\boldsymbol{\omega}_i$ is done separately for each trajectory i with linear ridge regression. The parameters $\boldsymbol{\mu}_{\boldsymbol{\omega}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\omega}}$ can be estimated with a maximum likelihood algorithm [16]. Thus

$$
\begin{aligned}
\boldsymbol{\omega}_i &= \left(\boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{\Psi} + \lambda\mathbf{I}\right)^{-1}\boldsymbol{\Psi}^{\mathrm{T}}\mathbf{Y}_i \\
\boldsymbol{\mu}_{\boldsymbol{\omega}} &= \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\omega}_i \\
\boldsymbol{\Sigma}_{\boldsymbol{\omega}} &= \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{\omega}_i - \boldsymbol{\mu}_{\boldsymbol{\omega}})(\boldsymbol{\omega}_i - \boldsymbol{\mu}_{\boldsymbol{\omega}})^{\mathrm{T}},
\end{aligned} \tag{2.33}
$$

where N is the number of demonstrations and $\boldsymbol{\Psi}$ the basis function matrix for all time steps. The demonstration data with the position and velocity of all joints and time steps is enclosed in $\boldsymbol{Y}_i$. The parameter λ is a ridge factor [22].

One big advantage of ProMPs is their ability to adapt to new situations. Both via-points/-velocities and final positions/velocities are possible by conditioning the MP on a new desired observation that is added to the probabilistic model. By applying Bayes theorem, new ProMP parameters can be calculated that are conditioned on the new observation $\mathbf{x}_t^* = \{\mathbf{y}_t^*, \boldsymbol{\Sigma}_{\mathrm{y}}^*\}$ at time t [16] with

$$
p(\boldsymbol{\omega}|\mathbf{x}_t^*) \propto \mathcal{N}\left(\mathbf{y}_t^*|\boldsymbol{\Psi}_t\boldsymbol{\omega}, \boldsymbol{\Sigma}_{\mathrm{y}}^*\right)p(\boldsymbol{\omega}) \tag{2.34}
$$

For a Gaussian distribution over the trajectory the new parameters are given by

$$
\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{\omega}}^{\mathrm{new}} &= \boldsymbol{\mu}_{\boldsymbol{\omega}} + \mathbf{L}\left(\mathbf{y}_t^* - \boldsymbol{\Psi}_t^{\mathrm{T}}\boldsymbol{\mu}_{\boldsymbol{\omega}}\right) \\
\boldsymbol{\Sigma}_{\boldsymbol{\omega}}^{\mathrm{new}} &= \boldsymbol{\Sigma}_{\boldsymbol{\omega}} - \mathbf{L}\boldsymbol{\Psi}_t^{\mathrm{T}}\boldsymbol{\Sigma}_{\boldsymbol{\omega}} \\
\text{with}\quad \mathbf{L} &= \boldsymbol{\Sigma}_{\boldsymbol{\omega}}\boldsymbol{\Psi}_t\left(\boldsymbol{\Sigma}_{\mathrm{y}}^* + \boldsymbol{\Psi}_t^{\mathrm{T}}\boldsymbol{\Sigma}_{\boldsymbol{\omega}}\boldsymbol{\Psi}_t\right)^{-1}
\end{aligned} \tag{2.35}
$$

The basis function matrix $\mathbf{\Psi}_t$ contains the basis functions for the desired time-step t of the via-point.
ProMPs are suitable to combine multiple movements and blend between different task. The reader is referred to [16] for further insight. Overall probabilistic movement primitives offer a powerful framework to represent movements. The ability to generalize and adapt to new situations outmatches DMPs. In [22] a stochastic feedback controller is derived to reproduce the trajectory distribution and to guarantee stability as long as the demonstrated trajectories were generated by a stable control law [16].

2.4 Extended Kalman Filter

The Kalman filter (KF) is an algorithm named after Rudolf E. Kálmán, who developed the discrete-time Kalman Filter in 1960 [23]. Its main application is the estimation of unknown states based on a series of previous measurements, that are corrupted by noise. It even grants good results, when the exact system model is not known. The algorithm can be derived as a mean squared error minimizer of the process states. The estimator is considered optimal (for linear systems with Gaussian noise), because it minimizes the estimated error covariance. As a result of its recursive solution, the KF is computationally very efficient and can run in real time, which explains its wide use in a variety of different fields, including control theory, navigation, signal processing and economics. The original formulation assumes a linear system and has been studied extensively in literature. A detailed derivation can be found in [24].
In reality no system is linear. The extended Kalman filter (EKF) provides a state estimation technique for nonlinear systems. The idea is to linearize the system around the KF estimates. The prediction and update equations are also based on the linearized system.
The following nonlinear system is given

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_k \tag{2.36}$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k, \tag{2.37}$$

where $\mathbf{x}$ is the state of the system and $\mathbf{y}$ the measurement. The random variables $\mathbf{w}$ and $\mathbf{v}$ are process and observation noise that are normally distributed with zero mean and the covariance matrices $\mathbf{Q}$ and $\mathbf{R}$, $p(\mathbf{w}) \sim \mathcal{N}(0, \mathbf{Q})$ and $p(\mathbf{v}) \sim \mathcal{N}(0, \mathbf{R})$. The process as well as the observation noise is assumed to be constant. The state estimation will be written as $\hat{\mathbf{x}}$ and the error of the state estimation is consequently $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$. The estimation error covariance matrix is

$$\mathbf{P} = \mathbb{E}(\mathbf{e}\mathbf{e}^\mathrm{T})$$

The estimation is divided into two steps the *a priori* and the *a posteriori*. The *a priori* estimate at step k, $\hat{\mathbf{x}}_k^-$ and $\mathbf{P}_k^-$, captures the knowledge about the system before the measurement $\mathbf{y}_k$ has been processed. The *a posteriori* gives the estimate after the measurement has been accounted for, $\hat{\mathbf{x}}_k^+$ and $\mathbf{P}_k^+$ [24].
The system in Equations (2.36) and (2.37) is linearized around the current state estimation $\hat{\mathbf{x}}$, resulting in the partial derivative (Jacobian) matrices:

$$\mathbf{A} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}}$$

$$\mathbf{C} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}}$$

The prediction step of the EKF consists of the state estimation and of the estimation of the covariance of the estimation error:

$$\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) \tag{2.38}$$

$$\mathbf{P}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^\mathrm{T} + \mathbf{Q} \tag{2.39}$$

Once a new measurement occurs the estimates are corrected in the update step:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{C}_k^\mathrm{T} \left(\mathbf{C}_k \mathbf{P}_k^- \mathbf{C}_k^\mathrm{T} + \mathbf{R}\right)^{-1} \tag{2.40}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_k^-\right) \tag{2.41}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \mathbf{P}_k^- \tag{2.42}$$

The Equations (2.38) - (2.42) summarize the discrete-time EKF [24]. It is pointed out that the prediction step can be calculated at each time step, meaning multiple times between new measurements. That is the reason why the superscripted "+" was neglected in (2.38). The correction however is certainly solely performed once a new measurement occurred.
The difficulty in implementing the EKF lies in the initialization of the values for $\mathbf{x}_0$ and $\mathbf{P}_0$ as well as the covariance matrices $\mathbf{Q}$ and $\mathbf{R}$. The system and measurement noise matrices can be used to tune the un-/certainty of the system and the measurements. If one is certain about the system, small values for $\mathbf{Q}$ can be set or if the measurements are known to be noisy, larger values for $\mathbf{R}$ are appropriate. Often the search for the initial parameters results in strenuous trial and error. A method on how to facilitate the initialization is given in [25].

3 Risk-Sensitive Policy Search

This chapter introduces the topic of optimizing risk-sensitive objectives with policy search. The first Section 3.1 aims to generate an intuition for risk-sensitive decision processes and will help to familiarize oneself with risk-measures such as the exponential reward transformation. Further in Section 3.2, we will explore the ability to adapt the algorithms, known from Section 2.2, to risk-sensitivity.

3.1 Risk-Sensitive Objectives and Measures

In Section 2.2 multiple algorithms for solving RL problems were presented. All of them are motivated by the objective to maximize J as the (cumulative) expected reward. In some situations it might be of advantage to include a measure of risk in the decisions of finding an optimal policy. For example, a policy that performs over-exceptionally well 99% of the time but in the remaining 1% has disastrous results, leading to an overall high expected reward, might be a poorer choice than a policy that performs just well but with consistent outcomes. One might think the first policy is being optimistic, because it assumes the uncertain outcome will play out in its favor. The latter decision on the other hand is considered pessimistic, because it assumes the uncertainty will turn out to its disadvantage and therefore a saver policy is chosen [26].
Thus, risk can be related to the certainty of return. For a stochastic system the reward R is a random variable and its uncertainty is given by its variance. Therefore, it becomes obvious to include the variance in the objective, which leads to the so-called mean-variance risk measure [27]:

$$\text{maximize} \quad J_{risk} = \mathbb{E}[R] - \frac{1}{2}\gamma \mathrm{Var}[R] \tag{3.1}$$

The parameter γ measures the risk-sensitivity and the factor $1/2$ will become apparent as the thesis develops. Depending on the sign of γ the variance will either be added or subtracted from the expected reward, resulting in an optimistic ($\gamma < 0$), risk-seeking, or a pessimistic ($\gamma > 0$), risk-averse, objective.
Equation (3.1) can be formulated with cost as well:

$$\text{minimize} \quad J_{risk} = \mathbb{E}[C] + \frac{1}{2}\gamma \mathrm{Var}[C] \tag{3.2}$$

To keep the definition of $\gamma < 0$ being risk-seeking and $\gamma > 0$ being risk-averse, the sign of the additive variance is switched.
Figure 3.1 shows an example of a shortest path problem. For every path a normally distributed cost is collected. The goal is to find the path form node 1 to node 6, while minimizing the collected costs. Using the mean-variance risk measure from Equation (3.2), one receives different optimal paths for various risk-factors γ:

$$\begin{aligned}
\text{red path: } \gamma &= 0, & J_{risk} &= 30, & \text{with } \mathbb{E}[C] &= 30 & \mathrm{Var}[C] &= 90\\
\gamma &= 1, & J_{risk} &= 75, & \text{with } \mathbb{E}[C] &= 30 & \mathrm{Var}[C] &= 90\\
\text{blue path: } \gamma &= 0, & J_{risk} &= 35, & \text{with } \mathbb{E}[C] &= 35 & \mathrm{Var}[C] &= 28\\
\gamma &= 1, & J_{risk} &= 49, & \text{with } \mathbb{E}[C] &= 35 & \mathrm{Var}[C] &= 28
\end{aligned}$$

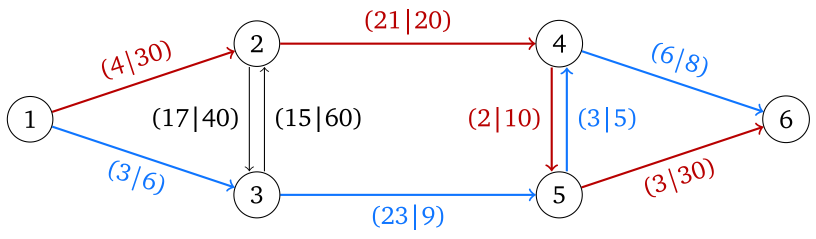


Figure 3.1.: Stochastic shortest path example. Traveling a path results in a cost, which is normally distributed with $(\mu|\sigma^2)$. The goal is to find the path that minimizes the mean variance objective (3.2). The red path is the result for the risk-neutral case $\gamma = 0$ (—) and the blue path for the risk-averse case with $\gamma = 1$ (—).

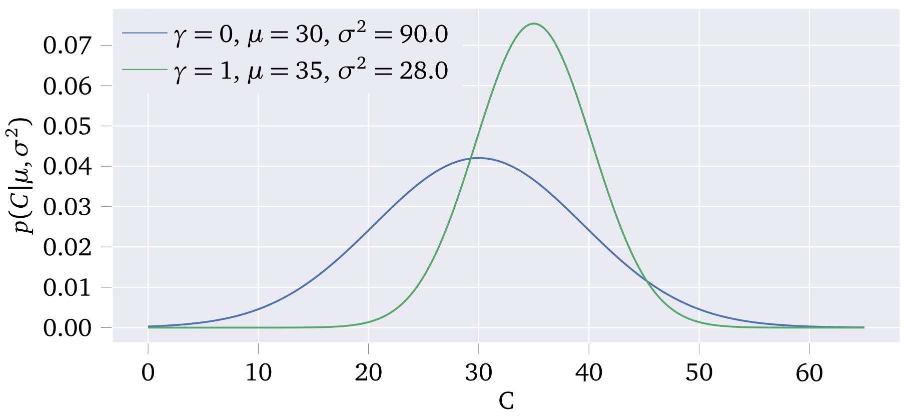


Figure 3.2.: This figure shows the distribution of the cumulative total cost for the optimal solution for $\gamma = 0$ and $\gamma = 1$. The risk-neutral objective results in a smaller but also wider distributed cost, making the results less certain. A higher risk factor penalizes the variance more and leads to a policy of higher total cost, but in return to a lower variance.

It becomes clear that a higher risk-factor gamma penalizes variance more. Hence, an optimal path with a higher expected cost but lower variance (blue) is favored over a path with lower expected cost but in exchange higher variance (red). As shown in Figure 3.2, the resulting cost distributions are widely spread for a risk-seeking solution and narrow with a higher mean-cost for risk-averse setting.
There exists a variety of risk measures with the mean-variance being only one of them [28]. In general risk-sensitivity can be described by an utility function. That means that the value of the reward r is not a proportional mapping of itself but rather a transformation by a function $U(r)$. The objective now changes to optimize the expected utility function $\mathbb{E}[U(r)]$. The expectation is over the uncertain random variable r.
One utility function proposed by Jacobson [29] is the exponential given by

$$U(r) = \mathbb{E}[\exp(-\gamma r)], \tag{3.3}$$

where γ can be interpreted as a risk-factor. The utility transformed reward can be seen in Figure 3.3.

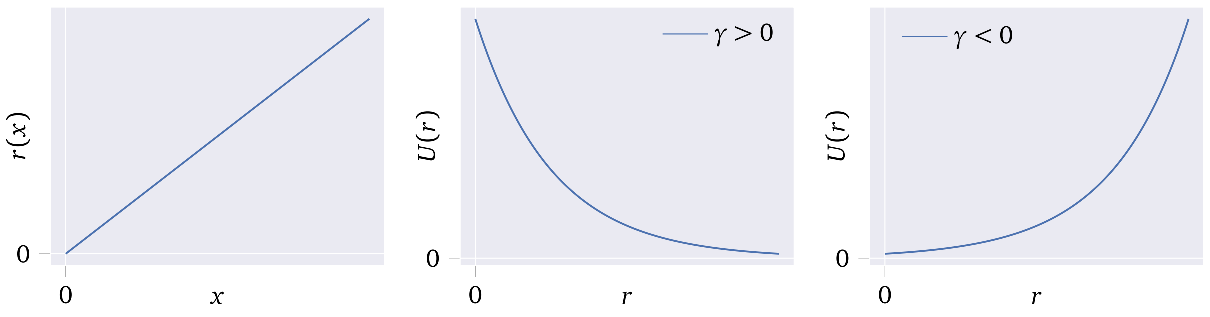


Figure 3.3.: Linear reward $r(x) = x$ and its transformation with exponential utility function $U(r) = \mathbb{E}[\exp(-\gamma r)]$ with $\gamma > 0$ and $\gamma < 0$. For $\gamma > 0$ the utility is a convex decreasing function that is minimized. Because $\mathbb{E}[U(r)] > U(\mathbb{E}[r])$ uncertain reward results in a higher utility value, thus certain reward is favored. For $\gamma < 0$ the utility is a convex increasing function that is maximized. Uncertain reward is favored.

For positive γ the expectation of $U(r)$ is minimized, which is a convex decreasing function. In this case $\mathbb{E}[U(r)] > U(\mathbb{E}[r])$, meaning an uncertain return results in a higher utility value. Thus, a certain reward with low variance is favored. The utility function is known to be risk-averse. On the other hand for $\gamma < 0$, the objective changes to be risk-seeking while maximizing a convex increasing function. Again $\mathbb{E}[U(r)] > U(\mathbb{E}[r])$, but as one is now maximizing the value of the utility, rewards with high variance are favored [30].
To make the objective easier to handle in terms of when to minimize or maximize, it is helpful to invert the utility transformation:

$$\begin{aligned} &\text{maximize} \quad J_{R,risk}(R) = U^{-1}\,\mathbb{E}[U(R)] = -\frac{1}{\gamma}\log\mathbb{E}[\exp(-\gamma R)] \\ &\text{minimize} \quad J_{C,risk}(C) = L^{-1}\,\mathbb{E}[L(C)] = \frac{1}{\gamma}\log\mathbb{E}[\exp(\gamma C)] \end{aligned} \tag{3.4}$$

Thus, the objective is returned to a reward or cost scale. Here $L(C)$ is called the disutility of a cost which is related to the utility of a reward by $L(C) = -U(-C)$. The disutility has the same risk-aware properties as the utility, meaning that $\gamma > 0$ still result in a risk-averse and $\gamma < 0$ in risk-seeking behavior [26].
One interesting property of the exponential risk measure is its relationship to the mean-variance measure. A Taylor series expansion of Equation (3.4) (derivation in Appendix A.3) leads to

$$\frac{1}{\gamma}\log \mathbb{E}[\exp(\gamma C)] = \mathbb{E}[C] + \frac{\gamma}{2}\text{Var}[C] + O(\gamma), \tag{3.5}$$

where $O(\gamma)$ are terms of order greater two that disappear for small γ. Hence, when comparing Equation (3.5) to (3.2) the exponential risk is approximately the mean-variance for small risk factors.

3.2 Risk-Sensitive Policy Search

In Section 3.1 we have learned how variance can be interpreted as a measure of risk and how to include a risk-sensitivity into an objective. The challenge in the following section is to explore how and if the algorithms from Section 2.2 can be used to solve a risk-sensitive RL problem. First, we will take a look at policy gradient with a risk criterion in Section 3.2.1. Later in Section 3.2.2, REPS is closely examined.

3.2.1 Risk-Sensitive Policy Gradient

Within this section the policy gradient method is extended to solve objectives given by Equation (3.1) and (3.4). Related work in this area has been done in [31], where a policy gradient algorithm that optimizes risk-sensitive criteria has been proposed. The criteria include the sharp ratio [32] and objectives that maximize the return with an upper bound on the variance or minimizing the variance while keeping a lower bound on the reward. The focus of this thesis will be on the exponential utility.
Considering episodic scenarios, the upper-level policy is optimized with a risk-sensitive objective. The objective functions for the mean-variance and exponential risk measure for an upper-level policy are given as:

(a) Mean-Variance

$$J^{\text{m/v}}_{\boldsymbol{\omega}} = \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[R(\boldsymbol{\theta})] - \frac{\gamma}{2}\text{Var}_{\pi_{\boldsymbol{\omega}}}[R(\boldsymbol{\theta})] \tag{3.6}$$

(b) Exponential

$$J^{\exp}_{\boldsymbol{\omega}} = -\frac{1}{\gamma}\log \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\exp(-\gamma R(\boldsymbol{\theta}))] \tag{3.7}$$

The gradients of the risk objectives (a) and (b) can be estimated using the likelihood-ration method analogously to the standard policy gradient in Section 2.2.1.

(a) Mean-Variance

$$\nabla_{\boldsymbol{\omega}} J^{\text{m/v}}_{\boldsymbol{\omega}} = \nabla_{\boldsymbol{\omega}} \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[R(\boldsymbol{\theta})] - \frac{\gamma}{2}\nabla_{\boldsymbol{\omega}}\text{Var}_{\pi_{\boldsymbol{\omega}}}[R(\boldsymbol{\theta})] \tag{3.8}$$

with

$$\nabla_{\boldsymbol{\omega}} \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[R(\boldsymbol{\theta})] = \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) R(\boldsymbol{\theta})]$$
$$\nabla_{\boldsymbol{\omega}}\text{Var}_{\pi_{\boldsymbol{\omega}}}[R(\boldsymbol{\theta})] = \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) R^2(\boldsymbol{\theta})] - 2\,\mathbb{E}_{\pi_{\boldsymbol{\omega}}}[R]\,\mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) R(\boldsymbol{\theta})]$$

(b) Exponential

$$\begin{aligned}\nabla_{\boldsymbol{\omega}} J^{\exp}_{\boldsymbol{\omega}} &= -\frac{1}{\gamma}\frac{\mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) \exp(-\gamma R(\boldsymbol{\theta}))]}{\mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\exp(-\gamma R(\boldsymbol{\theta}))]} \\ &= -\frac{1}{\gamma}\mathbb{E}_{\pi_{\boldsymbol{\omega}}}\left[\nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) \exp\left(-\gamma R(\boldsymbol{\theta}) - \log \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\exp(-\gamma R(\boldsymbol{\theta}))]\right)\right]\end{aligned} \tag{3.9}$$

The term $\log \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\exp(-\gamma R(\boldsymbol{\theta}))]$ can be dropped from Equation (3.9) as it does not influence the direction of the gradient. Also only the sign of $1/\gamma$ needs to be considered as it will decide whether to maximize or minimize the transformed reward. That results in a gradient of:

$$\nabla_{\boldsymbol{\omega}} J^{\exp}_{\boldsymbol{\omega}} \propto -\text{sgn}(\gamma)\,\mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) \exp(-\gamma R(\boldsymbol{\theta}))] \tag{3.10}$$

Equation (3.10) is similar to the standard policy gradient and only differs in the exponential transformation of the reward and the risk factor.

Equations (3.8) and (3.9) give estimates of the gradients and can be determined by samples. The advantage of using the exponential risk measures is evidently that we can use the standard policy gradient formulation from Equation (2.6). The only modifications is the replacement of the reward by the exponentially transformed reward. The exponential utility is from now on the risk measure of choice for all the executed experiments.
Furthermore, a baseline can be subtracted from the transformed reward analogous to Section 2.2.2. When using the average reward as a baseline, the advantage function will be estimated by:

$$\hat{A}(\boldsymbol{\theta}) = \exp(-\gamma R(\boldsymbol{\theta})) - \mathbb{E}\left[\exp(-\gamma R(\boldsymbol{\theta}))\right] \tag{3.11}$$

The solutions obtained from Equation (3.10) and (3.11) can be used in the NPG and PPO.
It is far from obvious how a PG method with a risk-sensitive objective will behave in an stochastic environment. For the gradient estimation the policy itself needs to be stochastic (see Sec. 2.2.1 for suitable policies). Thus, if the system is of stochastic nature, the system variance as well as the stochastic policy will introduce variability in the returns. In Chapter 4 we will conduct experiments to show the effects of a risk-sensitive objective on stochastic systems.
Furthermore, it could be of interest to use a risk-sensitive objectives in a deterministic environment, when considering the exploration-exploitation trade-off [3]. In policy search the exploration is usually accomplished by stochastic Gaussian polices. Since the stochastic policy introduces a variance in the deterministic system, the policy update of an risk-sensitive PG method should depend on the exploration. A simple one dimensional deterministic linear example

$$\begin{aligned} \text{maximize } -\frac{1}{\gamma}\log \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\exp(-\gamma R(u))] =&\text{minimize } \frac{1}{\gamma}\log \mathbb{E}_{\pi_{\boldsymbol{\omega}}}[\exp(\gamma C(u))] \\ \text{with } C =&|x| = -R \\ x =& au + b \\ u \sim& \mathcal{N}(\theta, \sigma_\theta^2), \end{aligned} \tag{3.12}$$

where the learnable parameters are θ and exploration noise σ_θ, is considered. The parameter for the example are $a = 1$ and $b = 0$. For the system in Equation (3.12) a single gradient update step is performed for multiple combinations of θ and σ_θ. In addition the risk-neutral, -averse and -seeking case is considered. The resulting gradient directions are displayed in Figure 3.4. The plots (a) to (c) show the vanilla policy gradient, while (d) to (f) the natural gradient. For both algorithms a risk-averse policy updates into the direction of smaller exploration noise, while a risk-seeking objective favors exploration. Interestingly, the effect is much stronger in the natural gradient. The conclusion to be drawn from this is that a pessimistic objective might converge prematurely as it minimizes the exploration noise too fast. Being optimistic on the other hand might be of advantage. Higher exploration is favored leading to more information about the system which in return could result in better solutions.

3.2.2 Risk-Sensitive Relative Entropy Policy Search

REPS in contrast to PG-methods does not perform the parameter update with gradient ascent. The new parameter distribution is fitted with a weighted maximum likelihood estimate and therefore waives the need for defining a step-size. As in the previous section, we aim to introduce REPS to a risk-sensitive objective. However, it has been noted in literature [5] that REPS already results in a risk-seeking policy. The closed form solution for the new policy arising from the optimization problem in Equation (2.17) is [12]

$$\pi(\boldsymbol{\theta}) = \mathbf{q}(\boldsymbol{\theta})\exp\left(\frac{R(\boldsymbol{\theta})}{\eta} - \log \mathbb{E}[\exp(R(\boldsymbol{\theta})/\eta)]\right). \tag{3.13}$$

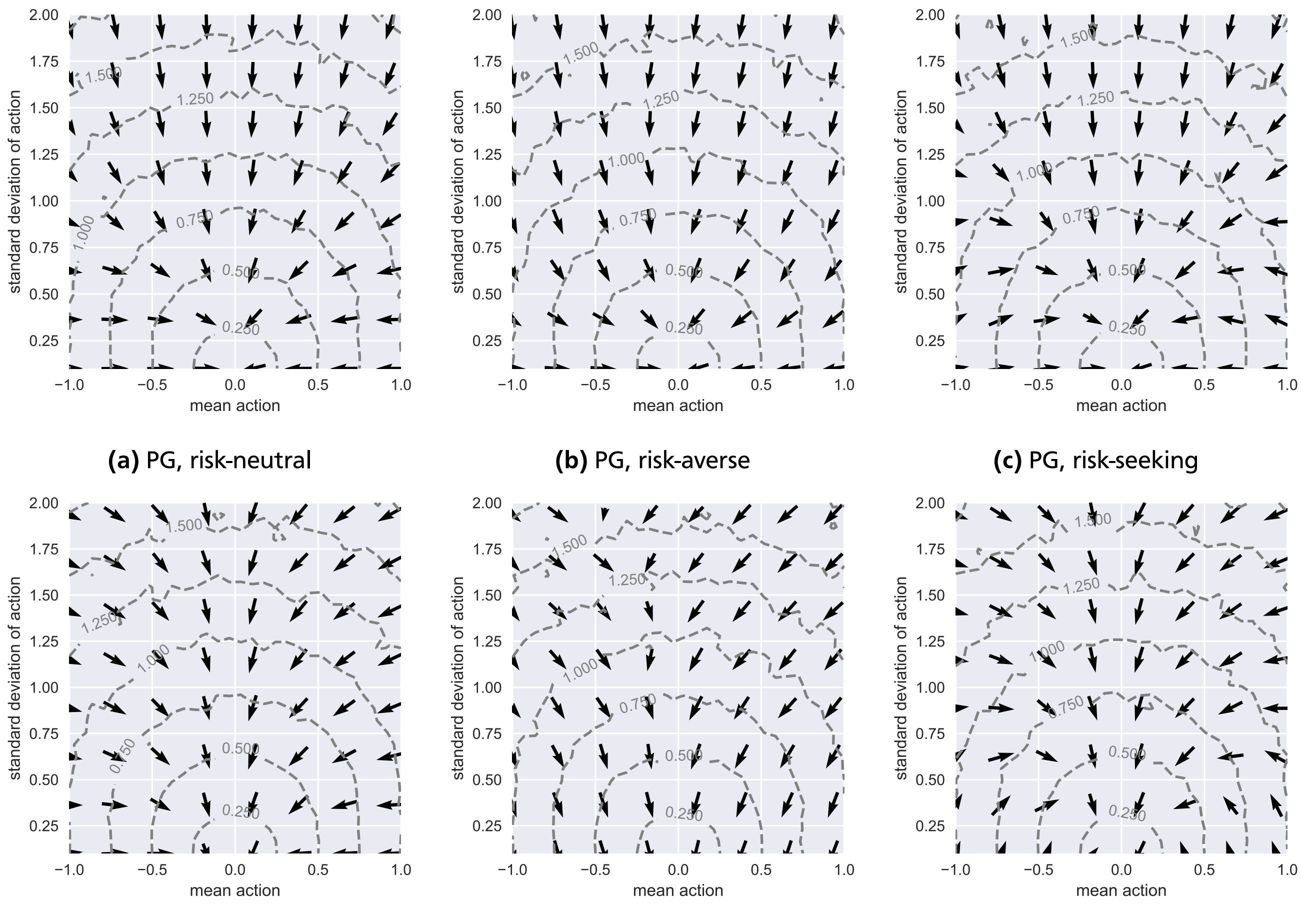
The optimal solution of Equation (3.13) is only known from samples, because the explicit function of $R(\boldsymbol{\theta})$ is usually unknown. In practice, a parametric policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ is fitted to the samples obtained from (3.13). This leads to the following optimization problem

$$\underset{\boldsymbol{\omega}}{\text{minimize}}\ D_{\mathrm{KL}}(\pi(\boldsymbol{\theta})||\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})) \propto \underset{\boldsymbol{\omega}}{\text{maximize}}\ \mathbb{E}\left[\log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})\exp\left(\frac{R(\boldsymbol{\theta})}{\eta} - \log \mathbb{E}[\exp(R(\boldsymbol{\theta})/\eta)]\right)\right]. \tag{3.14}$$

If we were to solve Equation (3.14) with a gradient method, the resulting gradient with respect to $\boldsymbol{\omega}$ is

$$\mathbb{E}\left[\nabla_{\boldsymbol{\omega}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})\exp\left(\frac{R(\boldsymbol{\theta})}{\eta} - \log \mathbb{E}[\exp(R(\boldsymbol{\theta})/\eta)]\right)\right], \tag{3.15}$$

which corresponds exactly, up to a scaling of $-1/\gamma$, to the gradient of the risk-sensitive policy gradient in Equation (3.9), with the Lagrangian multiplier η being anti-proportional to the risk factor γ. REPS solves the optimization in



(a) PG, risk-neutral **(b)** PG, risk-averse **(c)** PG, risk-seeking

(d) NPG, risk-neutral **(e)** NPG, risk-averse **(f)** NPG, risk-seeking

Figure 3.4.: Comparison of gradient directions between PG and NPG on a simple linear system with risk-neutral, risk-averse and risk-seeking objective. The risk-averse objective updates the policy towards lower variance and hence lower exploration. As a result the optimization problem might converge prematurely before finding an optimal solution. A risk-seeking objective can lead to an increase in the exploration. Interestingly, the effects are much stronger for the NPG.

Equation (3.14) with a weighted maximum likelihood (EM) estimate. The resulting weightings are $d^{[i]} = \exp(R(\boldsymbol{\theta}^{[i]})/\eta)$. Because of the convexity $\mathbb{E}[\exp(R)] > \exp(\mathbb{E}[R])$, rewards with higher variances are weighted more than those with lower variance, making the parameter update risk-seeking. The risk-sensitivity of REPS is dependent on the Lagrangian multiplier η. A small η results in a more optimistic objective than a larger η. In the formulation of REPS from Equation (2.17), the inequality constraint on the KL between new and old policy requires η to take on only positive numbers, $\eta > 0$. Thus, the solution will always have a risk-seeking component and never be risk-averse, which would require $\eta < 0$. In Section 4.1 we will validate on an experiment that REPS in fact results in risk-seeking policy.
To continue the thought that using a convex reward transformation as the weights of a maximum likelihood estimate, leads to risk-seeking behavior, the idea comes to mind to use a concave transformation, e.g. $d_i = -\exp(-\gamma R(\boldsymbol{\theta}_i))$, with some $\gamma > 0$. In theory, lower variance rewards should have a higher influence and result in risk-aversion. An arising problem with that approach is that the concave transformation, in contrast to the convex exponential transformation, does not guarantee positive weights. A 'dirty' fix could be to subtract the minimum weight from all the others and therefore shifting them to positive values. This idea could be used in the future to develop a risk-averse REPS alternative, in the Reward Weighted Regression (RWR) [33] or in Policy learning by Weighting Exploration with the Returns (PoWER)[34].

4 Experiments

Within this chapter, the risk-sensitive objectives and model free policy search algorithms of Chapter 2 are applied to different examples. The complexity of the model is kept to a minimum to enable thorough understanding and allow for clear performance comparison between algorithms. First, a portfolio management problem is discussed that naturally includes stochasticity in the returns. Generally, risk-sensitive optimization makes sense in finance problems, as an investor is always looking to find the perfect trade-off between performance and risk of his invested assets. Usually, the models are known beforehand, but in our example the goal is to find the best investment strategy, depending on the chosen riskiness, while only knowing the returns. Second, an example related to badminton will be examined that is close to problems usually considered in robot-control and -learning. Thereby we will investigate whether a risk-sensitive objective can be motivated in similar robot-learning problems. It is examined if autonomous system can be optimized not only for average performance, but also considering variance related criteria. Using risk-sensitive objectives in policy search is far from intuitive. The systems regarded in the following are of stochastic nature. Additionally, policy search algorithms rely on stochastic policies to estimate the parameter updates. Thus, with the system noise and exploration noise of the policy, two different noise sources can be identified that act upon the returns. Ultimately, the exploration noise should converge to a small value, once a solution to the problem is found. The question arises, whether a risk-averse objective will converge prematurely because the exploration is reduced too rapidly or, vice versa, whether the exploration noise still converges to small values in a risk-seeking case. Overall, the goal is to get an insight of how a risk-sensitive objective behaves in stochastic environments.

4.1 Portfolio Management

Portfolio management is determined by the constant decision of when and where private capital is invested. The capital is going to be allocated between multiple investment options. Naturally, the individual looks to maximize his returns, while keeping the risk to a minimum.
In the following example, the capital can be spread between N different assets. The complete portfolio is defined as $\mathbf{x}$, where x_i specifies the amount of investment i held. The total capital is normalized, thus $\sum_{i=1}^{N} x_i = 1$ [35].
The return of each asset is a random variable $\mathbf{r}$, distributed as a Gaussian with known mean return and variance:

$$\begin{aligned}\mathbb{E}[\mathbf{r}] &= \boldsymbol{\mu}_\mathbf{r} \\ \mathrm{Var}[\mathbf{r}] &= \boldsymbol{\Sigma}_\mathbf{r}\end{aligned} \tag{4.1}$$

The return of the complete portfolio is the cumulative reward of all the investments, times the amount of asset held, $R = \mathbf{r}^\mathrm{T}\mathbf{x}$. The cumulative reward R is again a random variable with mean and variance:

$$\begin{aligned}\mathbb{E}[R] &= \boldsymbol{\mu}_\mathbf{r}^\mathrm{T}\mathbf{x} \\ \mathrm{Var}[R] &= \mathbf{x}^\mathrm{T}\boldsymbol{\Sigma}_\mathbf{r}\mathbf{x}\end{aligned} \tag{4.2}$$

The goal is to find the optimal investment strategy that maximizes the exponential risk measure from Equation (3.4). With the risk factor γ, the risk-sensitivity can be tuned to receive different portfolio investment strategies. Thus follows the objective:

$$\text{maximize} \quad J_\text{risk} = -\frac{1}{\gamma}\log\mathbb{E}\left[\exp(-\gamma R)\right] \tag{4.3}$$

To optimize the objective above with model free policy search, a suitable policy must be defined. We are going to use the concept of upper and lower level policy. The lower level policy sets the amount of asset i held with a softmax function. This formulations guarantees the total investment cannot exceed 1:

$$x_i = \frac{\exp(\theta_i)}{\sum_{j=1}^{N}\exp(\theta_j)} \tag{4.4}$$

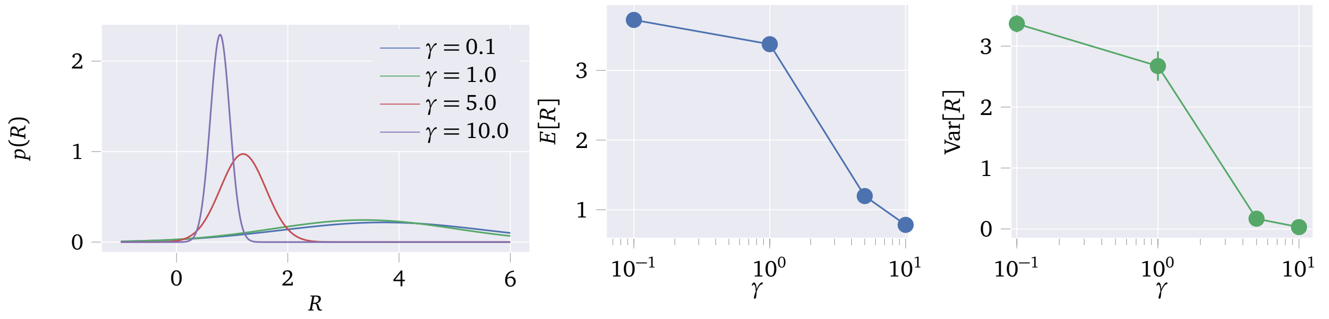
where $\boldsymbol{\theta}$ are the parameters obtained from the linear Gaussian upper level policy:

$$\pi(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}) \tag{4.5}$$

Thus, the learnable parameters are $\boldsymbol{\omega} = \{\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\}$.
For the simulation, a total amount of different assets is set to $N = 30$. The parameters of the random asset returns $\mathbf{r}$ are set as evenly spread on an interval $\boldsymbol{\mu}_\mathrm{r} \in [4, 0.5]$, $\boldsymbol{\sigma}_\mathrm{r} \in [2, 0.01]$ and $\boldsymbol{\Sigma}_\mathrm{r} = \mathrm{diag}(\boldsymbol{\sigma}_\mathrm{r}^2)$. This distribution of parameters can be interpreted as follows. Returns with a high expected value are accompanied with higher risks, whereas lower risk returns yield to lower mean reward. When comparing two policies π_1, π_2, chosen from an objective with risk factors $\gamma_1 > \gamma_2$, the first policy, π_1, should prefer lower risk assets and produce less return in average than π_2.

4.1.1 Policy Gradient with Risk-Sensitive Objective

In the following section, PG is tested on the system with the risk-sensitive objective of Equation (4.3). The parameters of the upper level policy are updated after each roll-out, which consists of 1000 samples. The algorithm is run for 2000 iterations. Additionally, the experiments are executed over 10 random seeds to ensure generalization.

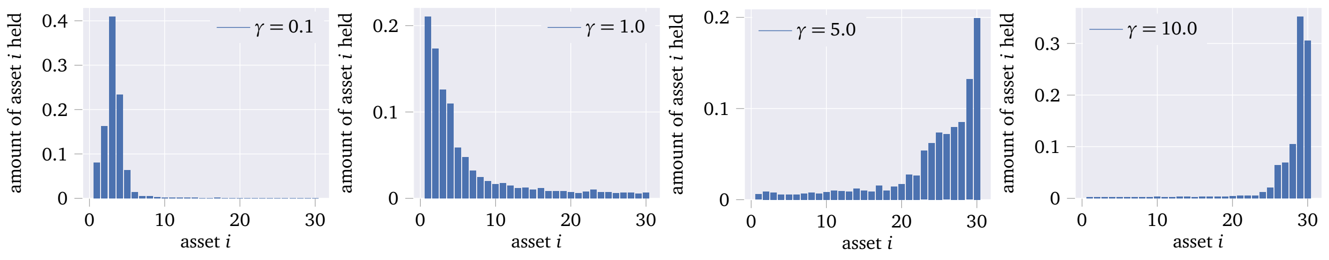


(a) Accumulated reward distribution

(b) Comparison of the expected accumulated reward $\mathbb{E}[R]$ and variance of the reward $\mathrm{Var}[R]$ over multiple γ.

Figure 4.1.: Resulting cumulated reward distribution for $\gamma \in \{0.1, 1, 5, 10\}$. Optimization is done with PG. Simulation is run over 10 random seeds. Each roll-out consists of 1000 samples. For a higher risk factor (higher γ results in more risk-averse objective) the total expected reward decreases while its certainty rises/variance decreases.

Figure 4.1 shows the distribution of the reward after the PG algorithm reached convergence for $\gamma \in \{0.1, 1, 5, 10\}$. The results coincide with the predictions, as a more pessimistic objective (more pessimistic equals higher γ) leads to a narrow reward distribution with lower mean, implying a smaller but more consistent reward. A small γ and therefore almost risk-neutral objective on the other hand, leads to an asset distribution that almost entirely aims to maximize the expected reward. The produced asset distributions can be seen in Figure 4.2. The assets were distributed to decrease in risk from $i = 1$ to $i = 30$. Thus the more risk-averse γ, the policy chose to invest in assets on the right, with lower variance.



(a) $\gamma = 0.1$ **(b)** $\gamma = 1$ **(c)** $\gamma = 5.0$ **(d)** $\gamma = 10$

Figure 4.2.: The resulting asset distributions for different $\gamma \in \{0.1, 1, 5, 10\}$ after convergence of PG on the portfolio management problem. For higher γ the resulting policy invests in assets which give a lower reward, but also have a lower variance. The policy becomes more risk-averse as γ increases.

In Figure 4.3 the convergence rate of the algorithm can be examined. PG attained convergence for all tested γ. It emerges, that the final values of J_risk depend on γ. When recalling the Taylor series transformation of the exponential risk measure (3.5), the variance acts as an ancillary additive term. As γ grows, the optimal value of J_risk goes down. For $\gamma = 5.0$ and $\gamma = 10.0$ it is apparent that the minimum variance term now outweighs the expected reward.

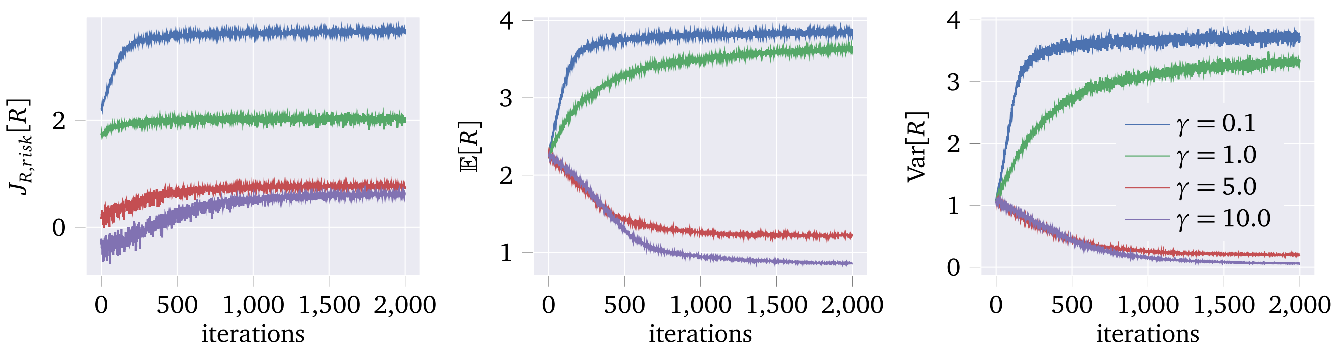


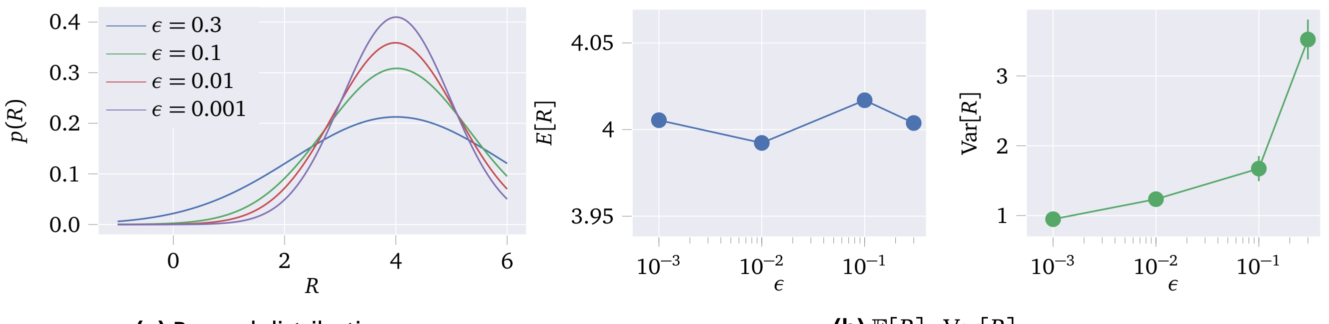
Figure 4.3.: The figure shows the convergence of the PG, optimizing the portfolio management problem for different $\gamma \in \{0.1, 1, 5, 10\}$. From left to right the exponential risk objective, $J_{R,\text{risk}}$, the expected reward, $\mathbb{E}[R]$, and variance of the reward, $\mathrm{Var}[R]$, are displayed. For $\gamma \in \{0.1, 1\}$, the expected reward outweighs the variance in the objective. As the variance gains more impact in the objective for $\gamma \in \{5, 10\}$, the resulting final variance decreases.

4.1.2 Risky Sensitivity of Relative Entropy Policy Search with Risk-Neutral Objective

The system in Equation (4.1) and (4.2) is now evaluated with REPS to examine the risk-sensitivity of the algorithm. The goal is to maximize the average return:

$$\text{maximize} \quad J_{\text{avg}} = \mathbb{E}[R] \tag{4.6}$$

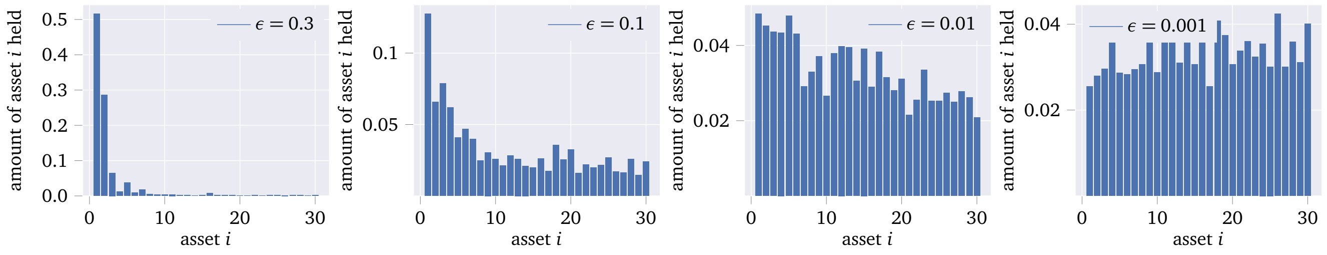
Thus, the variance is not included in the objective and any risk-sensitive behavior must come from the algorithm itself. As described in Section 3.2, the closed form solution for the policy update (Equation (2.18)) contains an exponential transformation of the reward, which makes the policy risk-seeking. The intensity of optimism depends on the Lagrangian multiplier η, which can be interpreted as a risk factor. The multiplier η is directly related to the KL-bound constraint ϵ. Hence, by running the experiments with different ϵ, the resulting policies should represent multiple levels of optimism. The asset returns were chosen to have the same mean reward of $\boldsymbol{\mu}_{\text{r}} = [4, ..., 4]^{\text{T}}$, the assets variance were evenly spread on the interval $\boldsymbol{\sigma}_{\text{r}} \in [2, 0.01]$, $\boldsymbol{\Sigma}_{\text{r}} = \text{diag}(\boldsymbol{\sigma}_{\text{r}}^2)$. The simulation was initialized with the capital equally invested in all the assets i. Thus, a risk-neutral policy should already be at its optimum, as it cannot gain on average reward, whereas an optimistic objective would invest most of the capital in assets with high variances. The results were collected over 10 random seeds, with each iteration consisting of 1000 roll-outs and run for a total of 800 iterations. For the convergence plots see Figure A.4.



(a) Reward distribution **(b)** $\mathbb{E}[R]$, $\mathrm{Var}[R]$

Figure 4.4.: Resulting cumulated reward distribution for $\epsilon \in \{0.3, 0.1, 0.01, 0.001\}$. Optimization is done with REPS on a risk-neutral objective. Simulation is run over 10 random seeds. Each roll-out consists of 1000 samples. This figure displays clearly that the resulting variance of the return is dependent on the KL-bound ϵ. As discussed in Section 3.2.2, REPS is naturally risk-seeking. Figure 4.5 shows, that for larger ϵ the capital is invested in assets with higher variance.

The experiment confirms that the KL-bound ϵ does have an effect on the risk-sensitivity of the resulting policy. In Figure 4.4 the reward distributions for $\epsilon = \{0.3, 0.1, 0.01, 0.001\}$ can be examined. Naturally, the expected reward is equal for all resulting policies. However, higher variances can be observed for a larger KL-bound. The change in variance of returns for different values of ϵ in Figure 4.4 confirms that the KL-bound has a significant effect on the risk-sensitivity of resulting policies. Larger ϵ lead to smaller Lagrangian multiplier η, thus increasing the risk-seeking behavior of REPS.



(a) $\epsilon = 0.3$ **(b)** $\epsilon = 0.1$ **(c)** $\epsilon = 0.01$ **(d)** $\epsilon = 0.001$

Figure 4.5.: Resulting asset distributions for $\epsilon \in \{0.3, 0.1, 0.01, 0.001\}$. Optimization is done with REPS on a risk-neutral objective. This figure shows, that REPS does in fact have a risk-seeking behavior, that is dependent on the KL-bound ϵ. Each asset has the same expected return but differs in the variance. For larger ϵ the policy chooses to invest in assets with a higher variance. For a smaller KL-bound the risk-seeking effect becomes less obvious.

Figure 4.5 shows the asset distributions for the produced policies. It can clearly be seen that from left to right, the distribution shifts from being very optimistic and investing in the risky asset with the highest variance, to investing more evenly into all assets. The results can further be compared to the PG with risk-neutral/-seeking objective. The plots can be found in the Appendix A.4, Figure A.3. The risk-neutral case returns similar results to REPS with small ϵ, whereas a risk-seeking PG also chooses to invest in assets with high variance.

The portfolio management example is perfectly suitable to show how and if the risk-sensitive objective works on the algorithms presented in Chapter 2. The reward distributions and convergence diagrams of NPG and PPO can be found in the Appendix A.4, figures A.1 and A.2. Since the results, with the exception of the required number of iterations, do not differ much from the PG, a more detailed explanation is omitted.

4.1.3 Conclusion

We demonstrated that PG-methods work with a risk-sensitive objective on an example motivated by portfolio management. The system was optimized with PG, NPG as well as PPO. Based on whether the risk factor is chosen to penalize or favor variance, the resulting policy chose to invest in assets with smaller or higher risk. It could not be observed, that risk-averse objectives lead to fast decrease of the exploration or that risk-seeking objectives cause an increase in exploration, thus hinder the convergence. Furthermore, REPS was used to find a solution to the problem, while maximizing the expected reward, without any variance measure. Even though the objective itself is free of any risk-sensitive criteria, it could be shown that REPS is risk-seeking and produces a policy that favors solutions with higher variance.

4.2 Badminton Toy System

This section studies the effects of risk-sensitive optimization applied to a problem of robot motor learning in the game of badminton. By understanding the following system, we aim to transfer our knowledge to similar problems encountered in control and robot reinforcement learning. Further, the example moves closer to the ultimate goal of learning a badminton task. The robot should eventually be able to return an incoming shuttlecock towards a desired target. In the following, a simplified version of such task is presented and analyzed in a risk-sensitive setting. Intuitively, it is obvious that there does not exist one unique solution of returning a shuttlecock to a target. For example, a clear will let the shuttle travel high up before descending down, while a smash will force the projectile to follow more or less a straight line. By introducing a measure of risk into the optimization problem, we hope that, depending on the degrees of risk-aversion or risk-seeking, policies representing different shuttlecock trajectories can be found. This idea is motivated by the intuition that a powerful forehand-smash is associated with a high risk, while a clear is considered to be a saver action.

4.2.1 System Dynamics and Optimization Problem

Firstly, the shuttlecock is simplified to a spherical ball to avoid dealing with higher order aerodynamic effects. Furthermore, by neglecting all friction a parabolic flight trajectory can be assumed. In a real badminton scenario one forces a velocity and resulting direction on the shuttlecock, by giving it a hard whack. As the velocity depends on multiple physical properties of both the racket and shuttlecock, such as the coefficient of restitution, mass of ball and racket, etc., it is assumed that the initial velocity right after the collision can be controlled directly. Thus, we will introduce the control variable $\mathbf{u} = [v_{x,u} \;\; v_{y,u}]$.

The system consists of two kinematic equations with constant velocity in x and constant acceleration in y direction.

$$\begin{aligned} x(t, x_0, v_{x,0}) &= v_{x,0} t + x_0 \\ y(t, y_0, v_{y,0}) &= \frac{1}{2} g t^2 + v_{y,0} t + y_0 \end{aligned} \tag{4.7}$$

where g is the gravitational constant, x_0 and y_0 are the initial position of the ball and $v_{x,0}$, $v_{y,0}$ are the ball's initial velocities. The desired target at the final time t_1 is going to be placed at $y(t_1) = 0$. Solving for $x(t_1)$ yields

$$\begin{aligned} t_1 &= \frac{v_{y,0}}{g} + \sqrt{\frac{v_{y,0}^2}{g^2} + \frac{2y_0}{g}} \\ x(t_1) &= x_0 + v_{x,0} t_1, \end{aligned} \tag{4.8}$$

where $y_0 \geq 0$ to prevent multiple or negative solutions for t_1. From (4.8) and as illustrated in Figure 4.6, it becomes clear that for a given target there exist multiple (infinitely many) solution trajectories, respectively infinitesimally many combinations of $v_{x,0}$ and $v_{y,0}$ to reach the desired target.

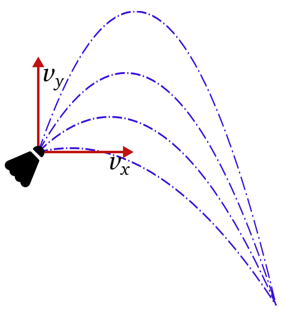


Figure 4.6.: Two dimensional badminton toy-system. The ball has a parabolic trajectory. The control variables are the initial velocities v_x and v_y of the ball. To reach a desired target multiple solutions are possible.

The actions that select the initial ball velocities are chosen from a parameterized stochastic linear Gaussian policy $u \sim \pi_{\boldsymbol{\omega}}(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\boldsymbol{\mu}_{\mathbf{u}}, \boldsymbol{\Sigma}_{\mathbf{u}})$ (see Sec. 2.2.1, Eq. (2.3)), where $\boldsymbol{\omega}$ are the learnable parameters, mean of initial velocity $\boldsymbol{\mu}_{\mathbf{u}}$ and the exploration noise $\boldsymbol{\Sigma}_{\mathbf{u}} = \mathrm{diag}(\sigma^2_{v_{x,0}}, \sigma^2_{v_{y,0}})$. Up until now, the only source of noise comes from the policy, but the system itself is purely deterministic. To investigate the influence of risk-sensitive objectives, an artificial noise κ is introduced into the system in the variables $\mathbf{v}_0 = [v_{x,0}, v_{y,0}]^T$. The noise is normally distributed. Hence, the initial velocities are composited of

$$\mathbf{v}_0 = \boldsymbol{u} + \boldsymbol{\kappa} \quad \boldsymbol{\kappa}(\mathbf{u}) \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{\kappa}}),$$

where $\mathbf{u}$ is the chosen action and $\boldsymbol{\kappa}$ the system noise with $\Sigma_{\boldsymbol{\kappa}} = \mathrm{diag}(\sigma^2_{\kappa,x}, \sigma^2_{\kappa,y})$. The optimization problem is formulated as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{\gamma} \log \mathbb{E}[\exp(\gamma|\tau_{\mathrm{des}} - x(t_1)|)] \\ \text{subject to} \quad & x(t_1) = x_0 + v_{x,0} \left(\frac{v_{y,0}}{g} + \sqrt{\frac{v_{y,0}^2}{g^2} + \frac{2y_0}{g}} \right) \\ & y_0 \geq 0. \end{aligned} \tag{4.9}$$

The desired target is τ_{des}. Only the x-component needs to be considered, as the y-component is by definition of (4.8) always 0.

Mean Absolute Error vs. Mean Squared Error

The measure of distance used in Equation (4.9) is discussed briefly in the following. Often in control problems, as in the linear-quadratic-gaussian control [23], a quadratic cost is used, but for our problem we will resort to using the absolute

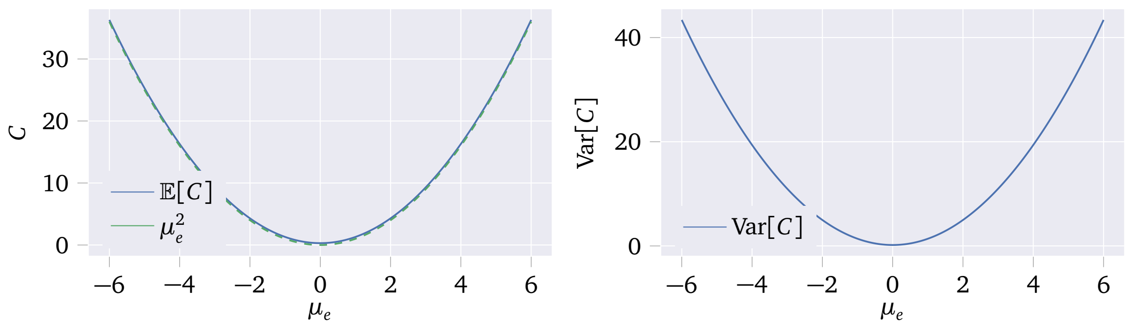
error. Why the absolute error is a better choice in our stochastic system than a quadratic error, is explained in the following.
A cost formulation is used, where the error from the desired target, $e = \tau_{\text{des}} - x(t_1)$, is transformed to a non-negative cost by the mean absolute error (MAE). The MAE is defined as

$$\frac{1}{N}\sum_{i=1}^{N}|e_i|, \tag{4.10}$$

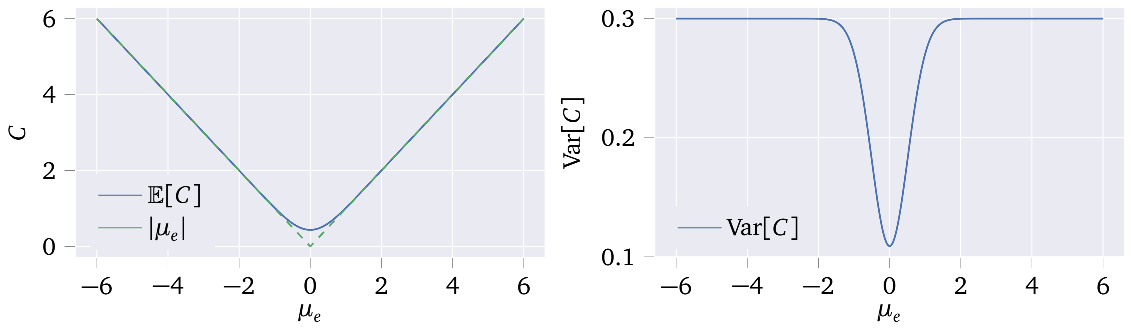
where N are the number of samples. Another option would be to choose the mean squared error (MSE):

$$\frac{1}{N}\sum_{i=1}^{N}e_i^2 \tag{4.11}$$

Because of the squaring, larger errors have a greater influence on the total error. That means that if the cost is computed by the squared error and the error is a random variable with a normal distribution, the variance of the cost will increase the farther away the error is from zero. If, on the other hand, the cost is computed with the MAE, the cost takes the absolute value of the error. Thus, the magnitude of error does not have an influence on the variance of the cost, if the error is far away from zero. However, for errors closer to zeros the variance of the cost decreases.
The difference between MSE and MAE is further clarified in Figure 4.7a and 4.7b. The plot shows the expected cost and variance of the cost $C = e^2$ and $C = |e|$ of a normally distributed error $e \sim \mathcal{N}(\mu_e, \sigma_e^2)$. The cost is therefore not normally distributed but follows a squared (a)/folded (b) normal distribution. Ideally we would like to have a cost transformation that follows the variance of the error. Therefore the MAE seems to be the better choice.



(a) Expected value and variance of the MSE of the cost $C = e^2$, where $e \sim \mathcal{N}(\mu_e, \sigma_e^2)$, $\sigma_e = \sqrt{0.3}$

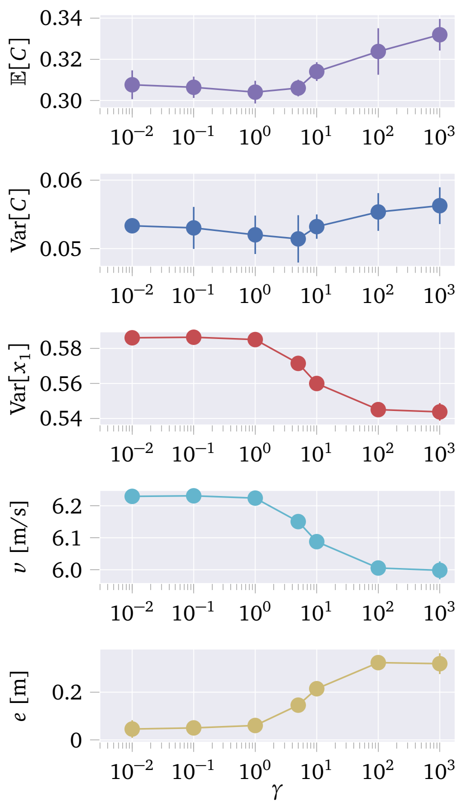


(b) Expected value and variance of the MAE of the cost $C = |e|$, where $e \sim \mathcal{N}(\mu_e, \sigma_e)^2$, $\sigma_e = \sqrt{0.3}$

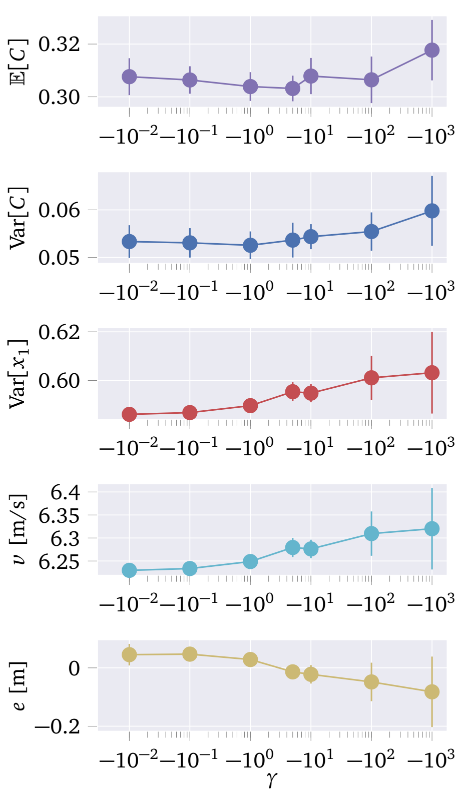
Figure 4.7.: Comparison of MSE and MAE for Gaussian distributed error. The variance of the error e is constant, but the variance of the MSE shows a squared increase in its variance the farther e is from zero. In contrast the variance of the MAE is equal to the error variance for errors not close to zero. Around zero the variance drops significantly.

4.2.2 Evaluation with Risk-Sensitive Natural Policy Gradient

After defining the system (4.8) and objective (4.9), the optimization was evaluated for multiple γ with the NPG and run until convergence with each roll-out consisting of 1000 samples. Moreover, each sequence was run over 10 different random seeds to ensure the conclusions generalize. The system variance was set to $\sigma_{\kappa,x} = \sigma_{\kappa,y} = 0.6$ and $x_0 = y_0 = 0$. The results can be seen in Figure 4.8a for risk-averse and 4.8b risk-seeking objective. The horizontal axis shows the development of the analyzed values for changing γ, where the scale is logarithmic for readability. In this context, $e = \tau_{\text{des}} - x(t_1)$ defines the error as the deviation of the actual x value at final time t_1 from the desired target and C is the associated MAE over all samples. The velocity $v_0 = \sqrt{v_{x,0}^2 + v_{y,0}^2}$ is the total initial velocity.



(a) NPG in risk-averse setting

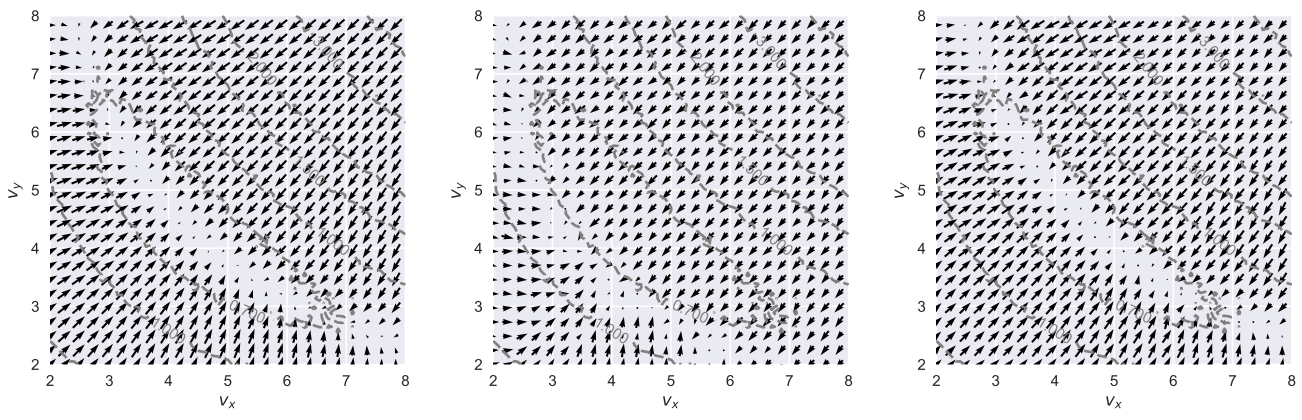


(b) NPG in risk-seeking setting

Figure 4.8.: Simulation results of the badminton toy-system using the NPG with a risk-sensitive objective for varying risk factor $\gamma \in \pm\{0.01, 0.1, 1, 5, 10, 100, 1000\}$. The system noise si $\sigma_{\kappa,x} = \sigma_{\kappa,y} = 0.6$ and the initial position is $x_0 = y_0 = 0$. In the risk-averse case the system variance, Var$[x_1]$, as well as the total initial velocity, v, are decreasing. The error also increases and indicates an undershooting of the desired target. Surprisingly, no trends in the expected cost nor the variance of the cost can be observed. The reason for this is probably, that the solutions fall in the nonlinearity of the cost variance displayed in Figure 4.7b. For the risk-seeking case the system variance and the total velocity increases. For large negative values the desired target is eventually overshot.

The results were inspected with the expectation of seeing clear tendencies of the variance of the cost decreasing for larger positive γ and vice versa increasing cost variance for smaller negative γ, mirroring the results observed in the portfolio management example in Section 4.1. A first look at the plots from Figure 4.8 is discouraging. The cost variance does not follow any clear trend of increasing nor decreasing. Rather, in the risk-averse case, Var$[C]$ is decreasing for $\gamma =\in \{0.01, 0.1, 1\}$ and increasing for $\gamma =\in \{5, 10, 100, 1000\}$. The same can be said for the risk-seeking scenario. The question that demands answering is, whether the algorithm is not working on the system or whether the variance of the cost might not be the right choice to judge the results. The latter seems to be the case, when noting the value of the Var$[C]$ being significantly smaller than the system variance. Remembering Figure 4.7b, taking the absolute value of the error as a cost transformation causes the variance of the cost to have a significant drop when being near $e = 0$. Hence, Var$[C]$ is not a suitable measure to compare the outcomes, as the results lie inside the nonlinearity of Var$[C]$.
In search for a different measure to compare the results, the variance of the system is calculated directly. The calculation can be simplified by setting $y_0 = 0$ in Equation (4.7) resulting in:

$$x_1 = x_0 + 2\frac{v_{x,0}v_{y,0}}{g}$$



(a) risk-neutral **(b)** risk-averse $\gamma = 10$ **(c)** risk-seeking $\gamma = -10$

Figure 4.9.: Gradient directions of the NPG for different combinations of the actions v_x and v_y. The contour lines define regions of the same cost. The regions, where the arrows diminish to points display the optimal solutions. Figure (a) shows the risk-neutral case. For the risk-averse (b) and -seeking (c) setting, the optima are shifted towards lower/higher variance spaces. Lower variance corresponds to lower initial velocities, while higher variance corresponds to higher initial velocities.

From here the variance of x_1 can be calculated:

$$\begin{aligned}
\mathrm{Var}[x_1] &= \mathrm{Var}[x_0] + \mathrm{Var}[2\frac{v_{x,0}v_{y,0}}{g}] = \frac{4}{g^2}\mathrm{Var}[v_{x,0}v_{y,0}] \\
&= \frac{4}{g^2}\mathrm{Var}\left[(u_x+\kappa_x)(u_y+\kappa_y)\right] \\
&= \frac{4}{g^2}\mathrm{Var}\left[(u_xu_y+u_x\kappa_y+u_y\kappa_x+\kappa_x\kappa_y)\right] \\
&= \frac{4}{g^2}\left(u_x^2\sigma_{\kappa,y}^2+u_y^2\sigma_{\kappa,x}^2+\sigma_{\kappa,x}^2\sigma_{\kappa,y}^2\right) \qquad (4.12)
\end{aligned}$$

In conclusion Var$[x_1]$ is dependent on the initial velocity in a quadratic manner. Thus, the variance can be minimized by reducing the initial velocity. With this new insight, we will revisit the results of the optimization of (4.9) with NPG.
First, the pessimistic risk-averse case is investigated. For this, positive $\gamma \in \{0.01, 0.1, 1, 5, 10, 100, 1000\}$ were chosen as the risk factor. For $\gamma \in \{0.01, 0.1, 1\}$ the cost as well as the error stay nearly the same. Further, no significant changes in the total velocity can be observed, which is probably due to the low variance of the system. Thus, the minimization of the expected average cost outweighs minimizing the variance. Starting at $\gamma = 5$ the variability in the returns gains impact on the objective. This becomes apparent in the decrease of both v_x and v_y which in return leads to a decrease in the system variance. This decrease however comes at the expense of the policy growing less accurately. The error e increases as the solution is 'pushed' towards a lower variance space, meaning that because of the lower velocity, the desired target is not reached. The cost variance is the only variance measure in the objective and the Var$[C]$ does not seem to follow the system variance. Surprisingly, pessimistic solutions still seem to minimize the system variance, although the latter is not known by the algorithm.
To inspect how the NPG performs in an optimistic risk-seeking setting, the risk factor was set to negative values, $\gamma \in \{-0.01, -0.1, -1, -5, -10, -100, -1000\}$. Recalling Equation (4.12), it is expected that a risk-seeking objective favors higher initial velocities. When examining the results, the total velocity v_0 shows a significant trend to increase. The system variance Var$[x(t_1)]$ follows the velocity. Hence, the assumption that an optimistic policy leads to higher velocities does hold. The error e is decreasing but reaches negative values at $\gamma >= 5$. The velocities are chosen too high and the target is eventually overshot. Interestingly, the variance of the simulations is grows over multiple seeds as γ increases, meaning that the algorithm converges to different solutions that are less and less predictable.

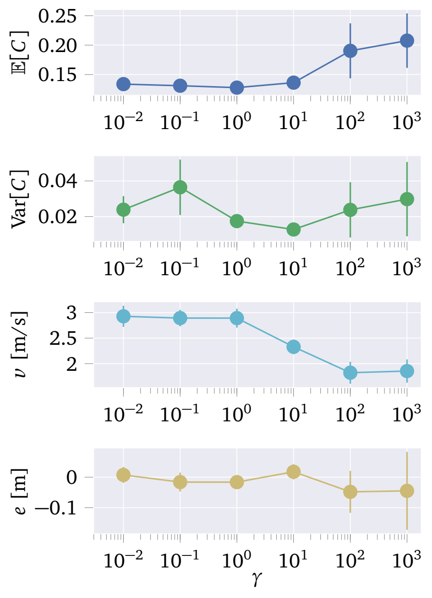
4.2.3 Effects of Risk-Sensitive Objective on Gradient Directions

As gradient methods are used, it is of interest to examine how the gradients behave depending on the risk-sensitivity. Figure 4.9 aims to clarify this. The system noise was set to $\sigma_{\kappa,x} = \sigma_{\kappa,y} = 0.6$ and the constant exploration noise in

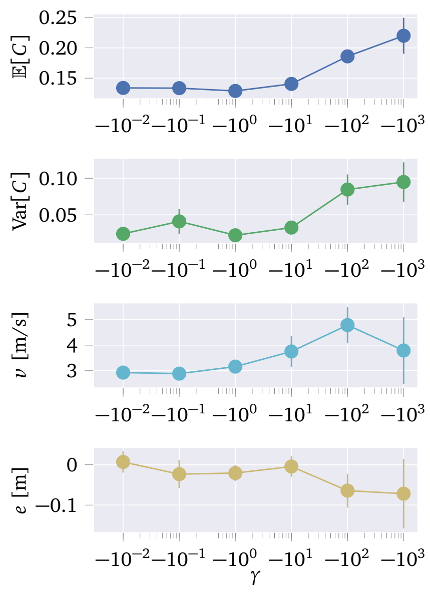
the actions to $\mathbf{\Sigma} = \mathbf{I}$. The x- and y-axis show the mean-action of a current policy and the arrows visualize the resulting gradient direction and thus, direction of policy improvement. The contour lines represent areas of the same cost. The areas, where the arrows diminish to points are the optimal solutions, as the policy cannot be further improved. As in the previous experiment, the NPG was used. The plots show the risk-neutral case, with the objective of minimizing the expected cost and both the risk-averse ($\gamma = 10$) and -seeking ($\gamma = -10$) cases that minimize the exponential risk measure from Equation (3.4).
The first apparent differences between the plots are the different locations of the optima. The optimum for the pessimistic and optimistic policies do not lie at the minimum of the average cost. Rather, as observed previously (see Figure 4.8), the pessimistic policy chooses lower velocities, while optimistic actions are shifted toward higher velocities. It is noticeable that the shift of the optimum in the risk-averse case is more apparent than in the risk-seeking one. Another important difference can be observed for gradients of the lower velocities (left bottom corner) between the risk-averse and the risk-neutral/-seeking case. The gradient directions of the pessimistic objective evidently strive towards the minimum velocity solution, while the risk-neutral objective only seeks the direction to the nearest optimum.

4.2.4 Contextual Risk-Sensitive Policy Search with Proximal Policy Optimization

This section examines the example toy system from Equation (4.8) but extends it to multiple contexts. The context $\mathbf{s}$ is defined as the initial ball position $\mathbf{s} = [x_0 \quad y_0]^\mathrm{T}$ and is sampled from a normal distribution at the beginning of each roll-out, while the desired target remains constant. Of course, for different $[x_0 \quad y_0]^\mathrm{T}$ the initial velocities have to change as well in order to reach the desired state. Therefore, the policy has to be conditioned on the context to choose $[v_{x,0}(\mathbf{s}) \quad v_{y,0}(\mathbf{s})]^\mathrm{T}$. A suitable possible policy has been presented in (2.4). As the feature vector $\boldsymbol{\phi}(\mathbf{s})$, 50 RFFs are used.
This example is curious, because additionally to the system and exploration noise, the system gains in stochasticity due to the sampling from the contexts. Furthermore, this extra disturbance does not act on the initial velocities but rather the initial positions, making it even harder to predict a minimum variance solution. It is investigated whether a risk-sensitive objective can be used in a multiple context system.



(a) PPO in risk-averse setting



(b) PPO in risk-seeking setting

Figure 4.10.: Simulation results of the contextual badminton toy-system using the PPO with a risk-sensitive objective for varying risk factor $\gamma \in \pm\{0.01, 0.1, 1, 10, 100, 1000\}$. The system noise is $\sigma_{\kappa,x} = \sigma_{\kappa,y} = 0.6$ and the initial position, x_0 and y_0, are sampled from a normal distribution. The plots show that a risk-averse objective forces the initial velocities to be smaller, while a risk-seeking policy clearly favors higher initial velocities. That the risk-sensitive objective has an influence on the variance of the cost, cannot be observed. This is probably because the cost variance falls into the nonlinearity displayed in Figure 4.7b.

The system was evaluated for various γ, run until convergence with each roll-out consisting of 1000 samples and executed over 5 different random seeds. The algorithm in use was PPO. The results, after convergence, for an optimistic and pessimistic objective compared over $\gamma \in \pm\{0.01, 0.1, 1, 10, 100, 1000\}$ are shown in Figure 4.10.
As discussed previously, following the conclusions from Equation (4.12), the initial velocity seems to be an indicator for the riskiness of the chosen policy. This insight can be recaptured in the present results. For a growing risk-averse γ the

total initial velocity decreases. Negative optimistic risk factors have the opposite response, the velocity is increasing. This trend seems to come to a hold at $\gamma = -1000$ as the velocity drops. In Appendix A.5 the convergence rate of the algorithm as well as the exploration noise is displayed. For $\gamma = -1000$ the exploration never converges to a small value, thus having a significant impact on the variance of the total system. This could be the reason why the velocity results in a lower value. Again the variance reduction cannot be observed in the costs distribution. The expected cost is increasing for both the optimistic and pessimistic objective which echoes the effect displayed in Figure 4.9 as the optimum is driven towards lower respectively higher velocities. It is noted that for both large negative and positive risk factors, the variance of the outcomes over different random seeds is quite large. Thus, the conclusions drawn are to be regarded with caution.

4.2.5 Conclusion

A risk-sensitive objective has been applied to a toy system that simplifies the return of a shuttlecock towards a desired target. The system was regarded in a singular context, where the initial position of the ball was constant, as well as for multiple contexts, where the initial position changed for each roll-out. The cost was calculated as the MAE of the derivation of the final state from a desired target. The absolute value transformation was chosen, because the alternative of using the MSE, depends nonlinearly on the variance of the error. The variance of the MAE has a minimum around error equal to zero but a linear dependency on the error variance if not close to zero. An “optimal” cost function would represent the variance of the error as it is and not be influenced by the error itself. So far the use of the absolute error as a cost is the best we can suggest.
The optimization was performed with NPG and PPO and evaluated for different risk factors. The results show that a risk-averse, low variance solution is associated with smaller initial velocities, while vice versa risk-seeking solutions favor high velocities. Unfortunately, both optimistic and pessimistic policies resulted in under-/overshooting the desired target. The found solutions were ’pushed’ toward a lower/higher variance space. It was not possible to find a risk-sensitive solution that hit the target. Instead a trade off between proximity to the target and hitting with lower/higher velocities could be observed. This leads to the conclusion that the exponential risk measure might not be the appropriate objective to optimize for the present system. In [31] it was suggested to minimize the expected cost while keeping a constraint on the variance. Such an approach could be helpful in our presented scenario and is open for further research.

5 Robot-Badminton on a Barrett WAM

Badminton is a racket sport played on a rectangular indoor court divided by a net. Points can be scored by striking the shuttlecock across a net and landing in the opponent's half of the playing field. The shuttlecock is a feathered projectile with a rounded cork base. The feathers expose the shuttlecock to high aerodynamic drag, causing it to decelerate quickly and giving it its distinctive flight trajectory. The sport requires excellent motor coordination, precision and speed. Depending on the shuttlecocks flight trajectory, players must decide in split seconds when, where and how to hit the projectile to return it over the net, back to the opponent. During competitive play a higher-level strategy must be involved to motivate the choice on where the shuttlecock should be returned to. This strategy will certainly depend on the opponent and the shuttlecocks trajectory, but might very well include some personal preferences, as the willingness to take risks.
The following chapter presents an approach to learn a robot-badminton task in a simplified environment. First, there will not be a net and the goal is to predict and return an incoming shuttlecock. Second, we will restrict the movement to one specific type, like a forehand stroke. Humans seem to learn movements as sets of templates, called motor primitives. Without the need to relearn, movements can be adapted quickly to new situations [36]. An example is the striking movement of the badminton forehand. Once learned how to perform the basic movement, it is relatively simple for a human to adjust to hit the ball in a new situation.
In the context of robot control, MPs offer similar properties as human motor primitives. After learning a task by demonstrations, their global behavior can be adapted using a small set of parameters, called meta-parameters [37]. These meta-parameters could consist of the final position and velocity of the movement or be introduced as via points. Learning only the meta-parameters, has the advantage of critically reducing the number of parameters, compared to relearning the numerous shape parameters of the MP-framework themselves. Throughout this chapter, a hitting movement is taught by a human instructor with kinesthetic teaching (see 5.2). Furthermore, the MP will be adapted to new situations through the use of previously introduced RL algorithms. The proposed method needs to perform a prediction of the ball trajectory into the future, which will be the topic of Section 5.4. The given setup is described in Section 5.1 before we move on to the formulation of meta-parameter learning as a RL-problem. The results are presented in Section 5.5.

5.1 Setup

The experiments were conducted on a Barrett WAM, a robot capable of fast movements and human-like kinematics. The robot is cable driven and can produce high accelerations needed for highly dynamic tasks. A standard badminton racket is attached to the end-effector of the arm, which has seven DOFs. An OptiTrack vision system consisting of ten infrared cameras is installed to track an incoming ball. At present the used ball is a standard table tennis ball, as it proved to be problematic to track a shuttlecock with the given tracking setup. In addition, a spherical ball has much simpler dynamics and is therefore easier to implement in a simulation environment. OptiTrack sends the positions of the tracked object in world frame coordinates with a frequency of 120Hz. Based on the measurements, the trajectory of the ball can be filtered and predicted with an EKF. The world frame is fixed at the base of the robot with the x-axis pointing toward the opponent, the y-axis to the right and the z-axis pointing up. The plane spanned by the y- and z-axis will be defined as the hitting plane. The intersection of the ball-trajectory with the plane determines the position at which the ball should be hit. If a ball is served towards the robot, a prediction of the trajectory must occur to find the desired hitting point and the time of when the ball will arrive (Figure 5.1). An EKF, that considers a simplistic ball model, including drag and gravity, is used to handle the prediction. The information calculated from the filter is then passed to the policy to decide which movement must be executed to hit the incoming ball.
The hitting movements have been taught to the robot using imitation learning. The movement primitive framework allows generalization to new situations to some degree, but should not deviate too much from the demonstrations. Operating the robot in a space that has never been visited by the imitation data is a security risk, as it is not predictable how the arm behaves in such situations. Thus, the robot only considers throws, if the ball lands in an area of the hitting plane that has been captured by the demonstration. Otherwise the ball is ignored.

5.2 Learning Movement Primitives from Imitation

The robot arm needs to be able to hit an incoming shuttlecock/ball in an area of the virtual hitting plane. A total of 25 demonstrations were provided by a human instructor with kinesthetic teaching. These demonstrations covered an area

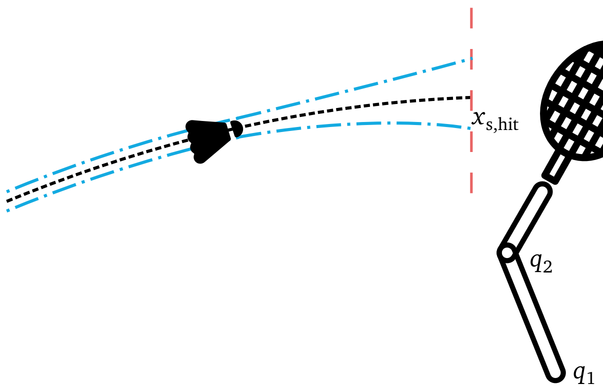


Figure 5.1.: Two dimensional representation of the badminton system. The red line defines the hitting plane and its intersection with the ball trajectory, the desired hitting point $\mathbf{x}_{s,\text{hit}}$. The variables q_1 and q_2 represent the multiple joints of the robot. The blue lines display the prediction of the shuttlecock. For estimations into the future the variance of the estimation increases.

of the hitting plane of approximately $1\text{m}^2 \times 0.5\text{m}$. The planning of a trajectory is later done in joints space, thus it is appropriate to learn a MP in joint space as well. In 2.3 two popular MP frameworks were presented. After some initial testing with both methods, it was decided to focus on ProMPs. The reason for this decision becomes clear when studying the collected imitation data in Figure 5.2, where the demonstrations of the first three joints are displayed. It is noticed, how the shape of the trajectories for joints q_1 and q_2 is very diverse. A single DMP could not properly capture all the information given by the data. Also, a ProMP has the advantage over DMPs of generalizing to via points. Hence, a single ProMP can capture both the hitting movement and the retraction to the rest posture. For DMPs the hitting and retraction must be split into multiple primitives.

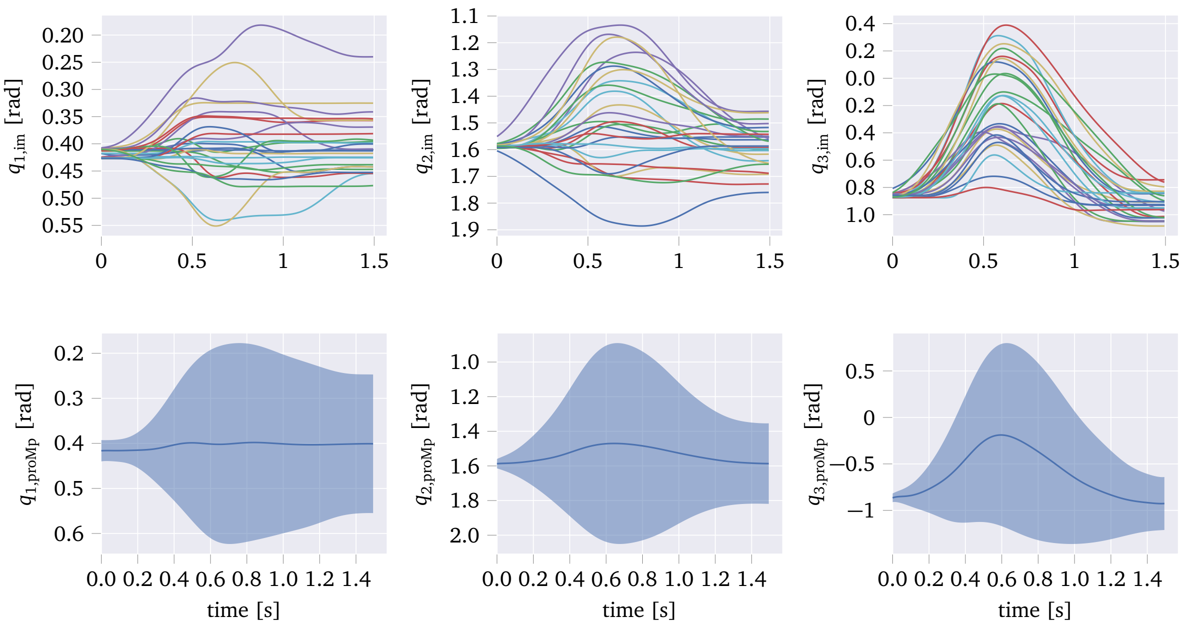


Figure 5.2.: The gathered imitation data for joints q_1, q_2, q_3 are displayed in the top three plots. A DMP could never capture the diversity of the demonstration. A ProMP on the other hand is designed to represent a distribution over trajectories. The distribution of the computed ProMP is shown in the lower plots.

In Figure 5.2 the imitation data and the computed ProMP can be observed. The data was synchronized and cut to have the same length and to capture comparable information about the hitting movement. The ProMP is able to include all demonstrations in its trajectory distribution.

Figure 5.3 plots the racket position in the world coordinate frame. Since the virtual hitting planes lies on the plane spanned by the y- and z-axis, it is possible to define a range over where the ball may cross and can still be hit. Any other ball positions fall into a space that is not captured by the trajectory distribution.

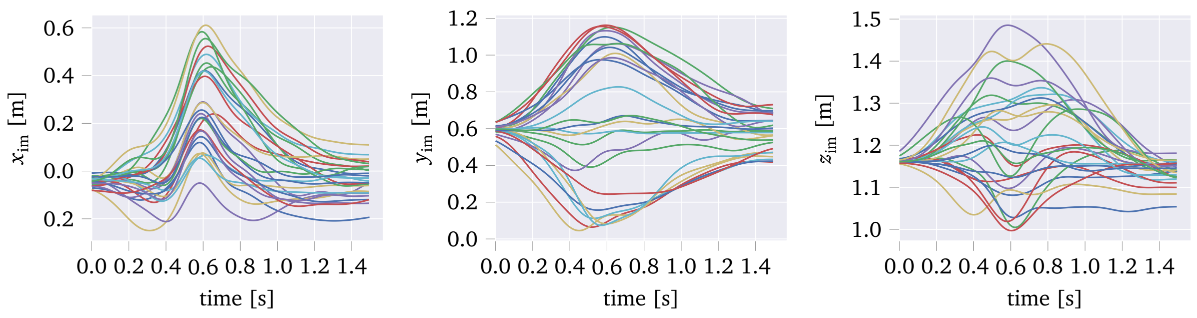


Figure 5.3.: The figure displays the coordinates of the end-effector in world frame of the imitation data. With this data, the area, in which the ball crosses the virtual hitting plane and can still be reached by the robot, can be found. The robot is prohibited to move to spaces not captured by the demonstration data, as stable behavior cannot be guaranteed.

5.3 Adapting Movement Primitives to New Situation

The goal of the badminton task is to hit an incoming shuttlecock and return it to the opponent. Rather than relearning the shape-parameters of the MP to fit each new situation, the advantage of ProMPs to generalize, is used. In practice this means that the same striking primitive is adapted for each incoming ball to reach the desired hitting point. The MP generated in Section 5.2 (see Figure 5.2) is a representation of a complete hitting movement, including both the hitting and retraction stage. Thus, hitting the ball does not take place at the final time and the ProMP must be conditioned on a via point. Said via point must be at the position where the ball-trajectory intersects the hitting plane. As the MP operates in joint space and the inverse kinematics are unknown, the meta-parameters $\boldsymbol{\rho}$ are build by the joint positions $\mathbf{q}_{t,\text{hit}}$ and velocities $\dot{\mathbf{q}}_{t,\text{hit}}$ of the robot at the hitting time step t_{hit}. For a seven DOF arm this results in a total of 14 tunable parameters.
Another critical parameter to consider is the timing T of the hitting movement that changes the duration of the primitive. In [37] it is suggested to include a timing in the meta-parameters. Within this thesis a different approach is used that includes the timing parameter in the state which needs to be set prior to movement execution. Additional state variables are the ball position and velocity at the hitting plane. Of course, the latter variables are not known before the movement starts. Thus, a ball prediction must be implemented (Section 5.4). The main assumption behind the reasoning to include the timing parameter in the states is that once the hitting movement has started, it will run in an open loop without any modifications being possible. Hence, the duration of the hitting movement corresponds to the prediction horizon of the ball estimation and by performing a shorter movement, the incoming ball can be observed longer, making the prediction of the ball more accurate. At later stages a higher level decision making policy could be used to decide how long the duration of the movement should be, depending on the given situation. Such a policy is not the topic of this thesis, but could constitute a research opportunity for the future.
In summary, the meta parameters consist of the joint positions and velocities of DOFs of the robot arm:

$$\boldsymbol{\rho} = [\mathbf{q}_{t,\text{hit}}^{\mathrm{T}} \quad \dot{\mathbf{q}}_{t,\text{hit}}^{\mathrm{T}}]^{\mathrm{T}} \tag{5.1}$$

The states are made up of the ball position, $\mathbf{x}_b = [x_{b,t,\text{hit}}, y_{b,t,\text{hit}}, z_{b,t,\text{hit}}]^{\mathrm{T}}$, and velocities $\dot{\mathbf{x}}_b = [\dot{x}_{b,t,\text{hit}}, \dot{y}_{b,t,\text{hit}}, \dot{z}_{b,t,\text{hit}}]^{\mathrm{T}}$ at the hitting plane and the timing parameter T, i.e.

$$\mathbf{s} = [\mathbf{x}_b^{\mathrm{T}} \quad \dot{\mathbf{x}}_b^{\mathrm{T}} \quad T]^{\mathrm{T}}, \tag{5.2}$$

where T is the prediction horizon of the ball estimation, respectively the duration of the movement until hitting of the ball.
After defining the meta-parameters and state variables a RL problem can be formulated. The problem is to find the stochastic policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ by optimizing

$$\underset{\boldsymbol{\omega}}{\text{maximize}} \quad J(\pi) = \int \mu(\mathbf{s}) \int \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s}) R(\mathbf{s}, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\mathbf{s}, \tag{5.3}$$

where $\mu(s)$ is the distribution over the context $\mathbf{s}$ and $R(\mathbf{s}, \boldsymbol{\theta})$ is the reward given the state $\mathbf{s}$ and parameter $\boldsymbol{\theta}$. The policy selecting the parameter $\boldsymbol{\theta}$ is defined as:

$$\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s}) \sim \mathcal{N}(\mathbf{M}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{s}) | \boldsymbol{\Sigma}_{\boldsymbol{\theta}}). \tag{5.4}$$

Algorithm 1: Learning meta-parameters for a robot-badminton task

```
Preparation: Learn ProMP by imitation learning
Initialization: Define number of samples $N$, initialize $\boldsymbol{\omega}_0$, define desired target $\boldsymbol{\tau}_{\text{des}}$
Result: policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s})$
repeat
    for each sample $n \leq N$ do
        sample timing parameter $T^{[n]}$
        do
            observe incoming ball
            predict when and where ball intersects hitting plane
        while time of intersection with hitting plane > $T^{[n]}$
        set states $\mathbf{s}^{[n]}$ = [ball position and velocity at hitting plane, $T^{[n]}$ ]
        sample $\boldsymbol{\theta}^{[n]}$ from stochastic policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s})$
        Calculate the meta-parameters $\boldsymbol{\rho}(\boldsymbol{\theta}^{[n]})^{[n]}$
        Condition and scale ProMP
        Execute ProMP
        Collect the reward $R(\mathbf{s}^{[n]}, \boldsymbol{\theta}^{[n]})$
    end
    Update policy parameters $\boldsymbol{\omega}$ using e.g. PPO
until convergence of $\boldsymbol{\omega}$
```

The learning parameters consist of the matrices $\boldsymbol{\omega} = \{\mathbf{M}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\}$. For the feature vector, RFFs (see Sec. 2.2.1) are used, since they are relatively easy to tune, with only one open parameter to be set.
The cost function is defined as

$$C = 100 \cdot \sum_{i \in x,y,z} |x_{i,b,t,\text{hit}} - x_{i,r,t,\text{hit}}| + c_t$$
$$c_t = \begin{cases} \sum_{i \in x,y,z} |\tau_{i,\text{des}} - x_{i,b,z_b=0}| & \text{if successfully hit} \\ 10 & \text{otherwise} \end{cases}, \tag{5.5}$$

where $\mathbf{x}_{b,t,\text{hit}}$ is the ball position and $\mathbf{x}_{r,t,\text{hit}}$ the robot position at the desired hitting time in world frame coordinates. The second term introduces a cost for the desired target $\boldsymbol{\tau}_{\text{des}}$. The target term has a constant cost if the robot failed to hit the ball and is otherwise calculated with the ball position at ground contact $\mathbf{x}_{b,z_b=0}$.
The desired hitting trajectory must not perturb "far" from the demonstration. To ensure the meta-parameters (respectively the via point) are always in a space captured by the imitation data, an interval of 3 standard deviations around the mean trajectory of the ProMP is defined:

$$\boldsymbol{\rho} \in [\boldsymbol{\rho}_{\min}, \boldsymbol{\rho}_{\min}] = [\bar{\mathbf{q}}_{t,\text{hit}} - 3\boldsymbol{\sigma}_{\mathbf{q}_{\text{t,hit}}}, \bar{\mathbf{q}}_{t,\text{hit}} + 3\boldsymbol{\sigma}_{\mathbf{q}_{\text{t,hit}}}] \tag{5.6}$$

The vector $\bar{\mathbf{q}}_{t,\text{hit}}$ consists of the mean joint positions and velocities at the hitting time step of the ProMP and $\boldsymbol{\sigma}_{\mathbf{q}_{\text{t,hit}}}$ is their standard deviations as defined by the distribution in Equation (2.32). Consequently, the output of the upper level policy does not correspond directly to the meta-parameters. Rather, the parameters $\boldsymbol{\theta}$ are transformed with a sigmoid function to fit into the allowed range $\boldsymbol{\rho} \in [\boldsymbol{\rho}_{\min}, \boldsymbol{\rho}_{\max}]$

$$\boldsymbol{\rho} = \frac{\boldsymbol{\rho}_{\max} - \boldsymbol{\rho}_{\min}}{1 + \exp(-\boldsymbol{\theta})} + \boldsymbol{\rho}_{\min}, \tag{5.7}$$

where $\boldsymbol{\rho} = [\mathbf{q} \quad \dot{\mathbf{q}}]^{\mathrm{T}}$ are the joint positions and velocitys and $\boldsymbol{\theta}$ the resulting parameter of the upper level policy. The ProMP can now be conditioned on the via point $\boldsymbol{\rho}$ using Equation (2.34) and subsequently scaled using the timing parameter T. The complete algorithm is summarized in Algorithm 1.

5.4 Ball Prediction

The following section deals with the problem of the ball prediction. So far it was mentioned that an estimation of the intersection of ball trajectory and hitting plane was necessary, but never stated how a prediction could be realized. We already decided to use a table tennis ball, with much simpler aerodynamics, instead of a shuttlecock. The next natural step to simplify could have been to neglect any drag, assume parabolic flight and fit a parabola to the observed measurements. Instead, an EKF is used, which brings the advantage that it can handle any nonlinear system and could be extended for shuttlecock prediction, in the future. Additionally, the EKF calculates the variance/certainty of the prediction. This could directly be used for a possible risk assessment of the state prediction. The disadvantage of an EKF is that some system knowledge is obligatorily, as the filter requires the system equations to predict the next state. It is however possible to include unknown, internal system parameters in the state of the filter. These parameters are treated in the dynamics of the EKF as constants but can be updated in the correction step.

The principle of an EKF has been explained in Section 2.4. As system model, a projectile under the influence of drag and gravity will be utilized. The aerodynamic drag is given by $F_D = 0.5\rho v^2 A C_D$ and acts in the negative direction of flight. Here v is the total velocity of the object $v = \sqrt{(v_x^2 + v_y^2 + v_z^2)}$, ρ is the air density, A the cross sectional area and C_D the drag coefficient. The drag can be split up into its directional components

$$\begin{aligned}
\mathbf{F}_D &= -\frac{1}{2} v^2 A C_D \frac{\mathbf{v}}{|\mathbf{v}|} && (5.8)\\
&= -\frac{1}{2} A C_D \frac{v_x^2 + v_y^2 + v_z^2}{\sqrt{(v_x^2 + v_y^2 + v_z^2)}} \mathbf{v} && (5.9)\\
&= -k_D \sqrt{(v_x^2 + v_y^2 + v_z^2)}\, \mathbf{v}, && (5.10)
\end{aligned}$$

where $k_D = 0.5 A C_D$ and $\mathbf{v} = [v_x\ v_y\ v_z]^{\mathrm{T}}$. Thus, the system equation is given in the following

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ -\dot{x} v \frac{k_D}{m} \\ -\dot{y} v \frac{k_D}{m} \\ -\dot{z} v \frac{k_D}{m} - g \end{bmatrix} \tag{5.11}$$

Here, the state vector is $\mathbf{x} = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}]^{\mathrm{T}}$ that includes the position and velocity of the object, $v = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}$ the total velocity and g the gravitational constant. The parameter k_D is usually unknown. For a spherical ball the parameter could be calculated but the idea arises to treat the parameter as a state of the EKF and make it open for estimation.

Thus, the state vector of the EKF consists of $\mathbf{x} = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ k]^{\mathrm{T}}$, where $k = k_d/m$. Notice that the mass m was included into the state as well. The resulting dynamical system is given by:

$$\begin{aligned}
\dot{\mathbf{x}} =& \mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ -\dot{x} v k \\ -\dot{y} v k \\ -\dot{z} v k - g \\ 0 \end{bmatrix} && (5.12)\\
\mathbf{y} =& \mathbf{h}(x) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} && (5.13)
\end{aligned}$$

Since the vision system only provides positions, the measurement model $h(\mathbf{x})$ contains x, y and z. For the prediction and correction step of the filter the system (Eq. (5.12)) and measurement model (Eq. (5.13)) need to be linearized (see Section 2.4):

$$\mathbf{A} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -vk - \frac{\dot{x}^2}{v}k & -\frac{\dot{x}\dot{y}}{v}k & -\frac{\dot{x}\dot{z}}{v}k & -\dot{x}v \\ 0 & 0 & 0 & -\frac{\dot{x}\dot{y}}{v}k & -vk - \frac{\dot{y}^2}{v}k & -\frac{\dot{y}\dot{z}}{v}k & -\dot{y}v \\ 0 & 0 & 0 & -\frac{\dot{x}\dot{z}}{v}k & -\frac{\dot{x}\dot{z}}{v}k & -vk - \frac{\dot{z}^2}{v} & -\dot{z}v \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.14}$$

$$\mathbf{C} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.15}$$

Having established the dynamical system, the matrices $\mathbf{Q}$ and $\mathbf{R}$ need to be set. Following the suggested approach described in [25] and some trial and error, good working values were found to be:

$$\mathbf{Q} = \text{diag}(0.001^2, 0.001^2, 0.001^2, 0.1^2, 0.1^2, 0.1^2, 0.01^2)$$
$$\mathbf{R} = \text{diag}(0.03^2, 0.03^2, 0.03^2)$$

The performance of the prediction is extremely dependent on accurate initial values for $\mathbf{x}_0$ and $\mathbf{P}_0$. The state can be initialized by the position measurement and the numerical differentiation of the positions to get the resulting velocities. An estimation for k can be calculated. A good initialization for $\mathbf{P}_0$ proved to be:

$$\mathbf{P}_0 = \text{diag}(0.02^2, 0.02^2, 0.02^2, 1^2, 1^2, 1^2, 0.2^2) \tag{5.16}$$

During our badminton task, before the hitting movement is executed, the KF updates its estimated current position at each time step and corrects its estimation when a new measurement is available. In addition, a prediction into the future is executed that predicts when and where the ball will pass the virtual hitting plane. Once the prediction estimates the time until impact is equal to the desired timing parameter T (duration of movement start until hitting/prediction horizon), the results get passed to the current policy to calculate the corresponding actions. To test whether the prediction is accurate enough even when no measurements occur, multiple ball trajectories were recored. The prediction was started 0.3s before the ball reached the hitting plane. Table 5.1 states the observed prediction errors in x, y and z position. The errors are around 2cm which on first sight seems to be fairly high. However, it needs to be taken into consideration that new measurements only arrive at a frequency of 120Hz. Thus, a ball flying at a velocity of 2m/s will move by 1.66cm between two consecutive measurements.

Error in m		
x	y	z
0.021 ± 0.023	0.021 ± 0.012	0.025 ± 0.012

Table 5.1.: MAE of the prediction with an EKF for a prediction horizon of 0.3s. The errors need to be considered with caution as new measurements are only captured with a frequency of 120Hz. In between measurements a ball with a velocity of only 2m/s will move 1.66cm.

A sample ball trajectory is displayed in Figure 5.4. The open loop prediction is started at $t \approx 0.4$s. The resulting estimated trajectory shows only small deviations from the measurements. It is clear to see, that after the prediction is run without any measurements updates, the variance of the prediction increases steadily. The results seem promising but rely heavily on the accuracy of the tracking system. If the initial velocity is faulty and deviates too much from the actual value, the prediction is not able to estimate the real trajectory. An example is displayed in the Appendix, Figure A.6.

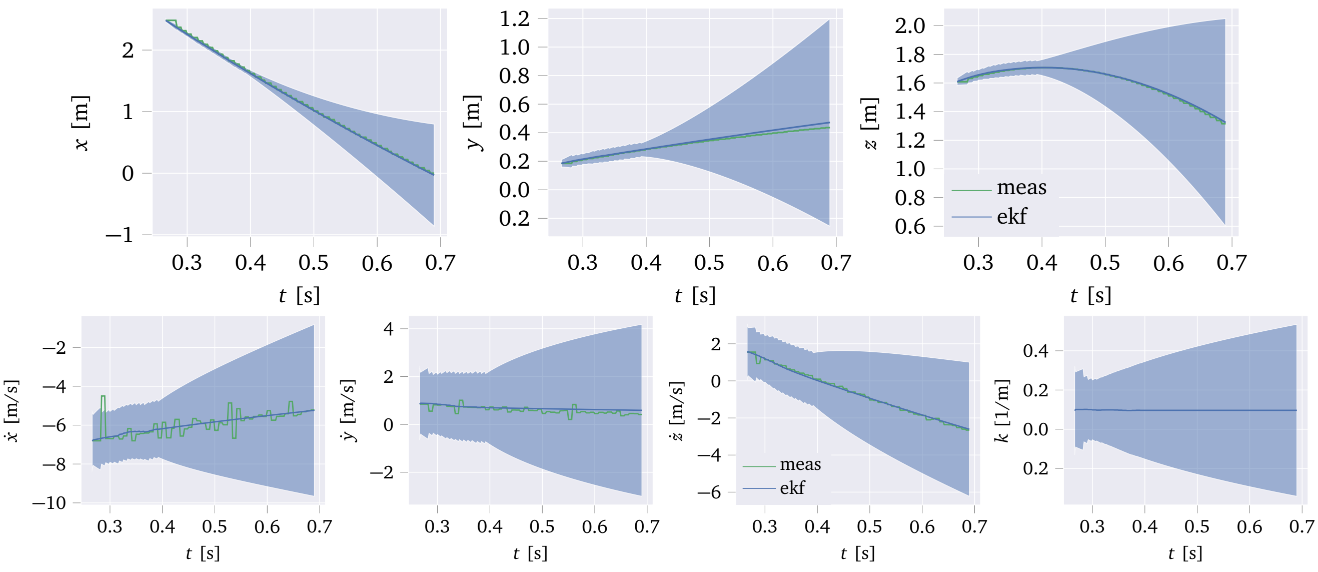
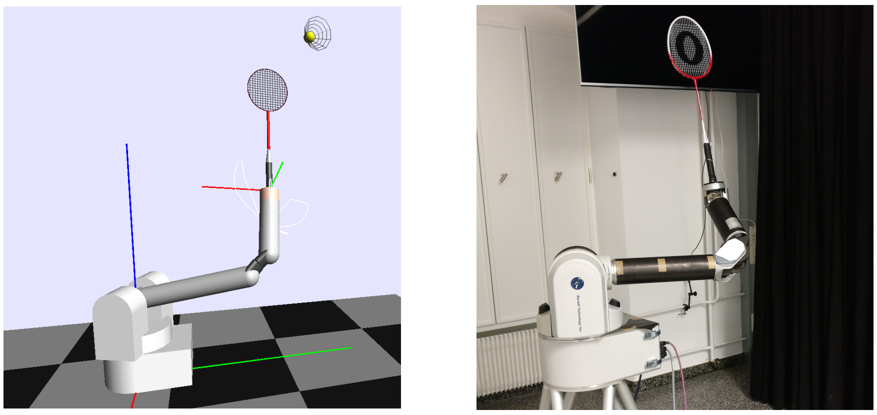


Figure 5.4.: This figure shows the prediction accuracy using an EKF. The open loop prediction starts at $t = 0.4$s. The data was recored on the real robot. The only measured values are the position of the ball. The velocities are computed by numerical differentiation and are noisy. Once the open loop prediction starts, the variance of the estimation increases steadily.

5.5 Learning Results

In Section 5.2, we generated a ProMP from imitation data that can be conditioned on meta-parameters. The learning problem of these parameters is described in Section 5.3. The complete algorithm, shown in Algorithm 1, requires a prediction of the desired hitting point on the virtual hitting plane in world coordinates. The prediction is implemented with an EKF and outlined in Section 5.4.



(a) Barrett WAM in the simulation environment

(b) Barrett WAM at IAS, TU Darmstadt

Figure 5.5.: Robot setup in simulation and the laboratory.

The learning of the meta-parameters is done in a simulation environment under near perfect conditions. The ball trajectories are sampled to reach the hitting plane in the area covered by the learned ProMP. The policy search algorithm to update the parameterized policy was chosen as PPO. Each roll-out consists of 100 samples and the algorithm was run for a total number of 800 iterations, resulting in a total number of $8 \cdot 10^5$ roll-outs. The feature vector was generated by 800 RFFs. At the beginning of the learning process, the robot was only able to hit about 10% of the incoming balls. After learning, the success-rate of a hit grew to 95%. The convergence rate of the algorithm can be seen in Figure 5.6. Due to the high number of roll-outs needed to get a satisfactory hitting percentage, its is unfeasible to do the learning on the real robot setup. Nevertheless, the results of the simulation can be evaluated on the real robot. When using the solutions learned in simulation with the real Barrett WAM, only a slight loss of performance could be observed. The robot is able to hit the ball most of the time. Most often, when a hit failed, the cause can be traced back to a poor ball prediction, which was most likely caused by an inaccurate initialization of the EKF.

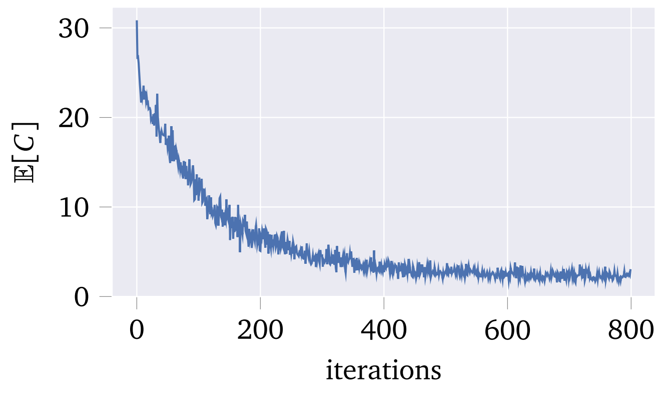


Figure 5.6.: This figure shows the convergence rate of the simulated badminton task using PPO. The simulation was run for 800 iterations with 100 roll-outs.

Overall the learning results are satisfactory. The robot learned to hit an incoming ball over an area of approximately 0.5m^2. The learning could be done in simulation and transfered well to the real system. The main problem on the real setup was the tracking and prediction of the ball. PPO needed a lot of roll-outs to learn sufficiently good solutions. In [37] cost-regularized kernel regression (CrKR) was shown to be more suitable for learning meta-parameters. The proposed methods required only a fraction of roll-outs compared to gradient methods.

5.6 Outlook

In the above sections, a method to teach a robot to return a ball was presented. However, in the scope of this thesis, risk-sensitivity was not considered in the context of the robot badminton task. In this section we discuss future research suggestions.
One possibility to include the awareness of risk, could be motivated by the ball prediction. A prediction can be intuitively associated with risk, because the results are estimates of future states, which are uncertain. In the presented approach the prediction horizon is given as the parameter T. Once the prediction occurs, the hitting movement starts and is run in an open loop. As a result the estimation becomes less accurate for a larger prediction horizon, making the hitting of the ball imprecise. Thus, the parameter T could be used to tune the riskiness of the task. Because the horizon T was included into the state vector, its value can and must be set before the hitting movement is initiated. This has the advantage that the riskiness of the next hit can be selected based on information from the environment.
The policy that was learned to return the ball, is from now on referred to as the hitting policy. We suggest to develop a decision making policy that has the responsibility to select all higher-level parameters of the underlying hitting policy, depending on the current situation, such as the positioning of the opponent on the court. In addition to the prediction horizon, these higher level parameters could include the desired target τ_{des} to return the ball to and the type of hitting movement the robot should execute. For selecting a desired target, the learning of the hitting policy can be modified to include τ_{des} inside the state vector as well. Hence, Algorithm 1 is modified to sample not only T at the beginning of each episode but also τ_{des}. Resulting is a hitting policy that depends on the state $\mathbf{s} = [\mathbf{x}_b^{\text{T}} \quad \dot{\mathbf{x}}_b^{\text{T}} \quad T \quad \tau_{\text{des}}]^{\text{T}}$, where the ball position and velocity - $\mathbf{x}_b$, $\dot{\mathbf{x}}_b$ - are set by the prediction, while T and τ_s are the open higher-level parameters.
Furthermore, to widen the skill-set of the robot, a library of multiple kind of movements can be created. Each skill will consist of a hitting policy that represents a certain type of stroke, i.e. a forehand or backhand shot. Depending on the situation, the decision making policy could select the type of movement from the skill library and set the higher-level parameters of the hitting policy. An approach to generalize between different MPs has been demonstrated in [38]. The decision making policy could be learned with the in Chapter 3 proposed risk-sensitive policy search methods.

6 Conclusion and Outlook

This thesis was dedicated to discuss reinforcement learning approaches with a risk-sensitive objective. Robot badminton was used as an application example. The following is a summary of the thesis and a discussion about possible extensions of the work that may be investigated in the future.

6.1 Summary

This thesis has contributed to the fields of reinforcement learning and robotics. In Chapter 2 a brief introduction to reinforcement learning was given followed by an insight into the topic of policy search methods. We analyzed state-of-the-art RL algorithms, such as PG, NPG, PPO and REPS. Furthermore, two movement primitive frameworks were explained, namely DMP and ProMP. A concise introduction to the EKF concluded the foundations chapter.
Chapter 3 contains our main contribution, the adaption of policy gradient methods to accommodate a risk-sensitive objective. A policy gradient algorithm for the mean-variance, as well as the exponential risk measure was developed. Depending on a risk factor, the objective can be varied in its risk-sensitivity. We found that the PG, NPG and PPO all can be extended to be risk-aware. We examined the effects of risk-sensitive objectives on the exploration-exploitation trade off, within the policy gradient methods. Good exploration strategies are crucial for finding efficient solutions and policy search methods enforce exploration by searching over distributions of actions. Therefore, the exploration introduces stochasticity in the returns, which will effect the risk-sensitive objective. It could be observes that optimistic behavior encourages the algorithm to explore the objective landscape, while being pessimistic penalizes excessive exploration. In conclusion, it can be of advantage to be risk-seeking in order to gather more information about the environment, while being risk-averse could hinder exploration and might cause premature convergence.
Furthermore, REPS was analyzed for its risk-sensitivity. Interestingly, the derivation of REPS leads naturally to optimistic solutions, without considering any variance related criteria in the objective function. The Lagrangian multiplier enforces the KL-bound of the policy update and can be interpreted as a risk factor. It is an open research question, whether REPS can be adapted to behave in a risk-averse way.
The risk-sensitive algorithms were examined on experiments in Chapter 4. On an example motivated by portfolio management, PG, NPG and PPO proved to successfully find risk-averse solutions when optimizing a pessimistic objective. Moreover, the risk-sensitivity of REPS could be verified by using a risk-neutral objective on the portfolio management problem. Depending on the KL-bound, REPS found risk-seeking solutions. Furthermore, we put the risky policy gradient methods to a test on a badminton related ball throwing example. The goal was to understand if risk-sensitive objectives are applicable in robot learning and control. The problem was solved with NPG and PPO optimizing the exponential risk measure. While both algorithms found lower/higher variance solutions in the risk-averse/-seeking case, we concluded that the exponential risk measure was not the best suited one. The trade-off between maximizing the expected reward and minimizing the variance, found solutions that resulted in overshooting/undershooting the desired target. A more suitable objective would enforce the hitting of the target, while allowing to find solutions that differ in their variance. Overall, the risk-aware gradient methods could be used successfully to solve the given systems, while considering the risk-sensitivity. Initial concerns that a risk-averse objective results in premature convergence or risk-seeking to overly excessive exploration, could not be observed. However, it is still not understood, how and if the algorithm can distinguish between internal system noise and noise introduced by the stochastic policy.
In Chapter 5 an approach was proposed to teach badminton to a real Barrett WAM. For an incoming shuttlecock the robot should predict the projectile's trajectory and its intersection with a virtual hitting plane. Subsequently, a trajectory towards the predicted intersection point is planned to hit the ball and return it. Because the tracking of a shuttlecock turned out to be difficult, a table tennis ball was used as an alternative. The ball prediction was successfully implemented with an EKF but is very prone to initialization errors. Instead of learning the complete hitting trajectory by RL, we exploit the advantage of MPs to generalize to new situations by adapting only a small set of meta-parameters. The hitting movement is captured by a ProMP and was learned by kinesthetic teaching. The learning of the meta-parameters, which consisted of the joint positions and velocities of the robot at the hitting time-step and a timing parameter, was done with PPO. The learning of the problem demands a large number of trials and was therefore executed in a simulation environment. In simulation, under near perfect conditions, the robot could successfully return about 95% of the balls, that fell in an area of 1m × 0.5m on the hitting plane. The simulation results transfer to the real robot and the Barrett WAM is able to return most of the served balls. In the scope of this thesis the badminton task was not learned with considerations of risk-awareness. Section 5.6 gives an outlook how the task can be extended in the future.

6.2 Open Problems and Outlook

This section will cover open problems that were met during the thesis and discuss possible future directions of research.

6.2.1 Risk-Sensitive Policy Search

The main problem with risk-sensitive policy search is that the system is exposed to multiple variance sources. The model free nature of the proposed policy search methods requires the estimating of the parameter update with samples from a policy distribution. Additionally, the system itself may be of stochastic nature. Hence, the resulting reward depends on both the system and policy noise. While we might want to minimize the variance of the system, we need to be optimistic to some degree in the policy to ensure exploration. It is conceivable that in some scenarios the risk-averse objective could force the exploration to decrease and thus leading to premature convergence. Surprisingly, it was discovered that risk-averse solutions can be found in the experiments that minimize the system variance without negatively impacting the exploration. The question of how to separate the two noises and whether it is possible, is still unanswered by this thesis and an interesting research topic.
The exponential risk measure might not be the best objective to minimize in certain cases. For the ball throwing example, low variance solutions were found, but the desired target was not reached. It might be preferable to find an objective that forces the ball to reach the target but depending on the risk-sensitivity find solutions that differ in their variance. Future work can be done in exploring different measures of risk and using them with policy gradient methods. For example it would be possible to minimize the variance of the return, while keeping a lower bound on the reward, or maximizing the reward, while keeping an upper bound on the variance as suggested in [31].

6.2.2 Robot Badminton

The main source of error in the robot badminton task is an inaccurate ball prediction. The error of the prediction was mostly caused by a wrong measurement of the tracking system that was used to initialize the EKF. The solution would be to include more cameras in the setup to reduce the chance of measurements errors. The disadvantage of the EKF is that it requires a system model of the ball. In the future, the EKF could be replaced by a different prediction method that might use a model-free approach. The prediction should get better over time and include the present state of the ball, as well as all observed ball states in the past. The task was learned and evaluated with a spherical ball instead of a shuttlecock. The reason for this was the difficulty in tracking it in the current setup. Additional, the spherical ball has much simpler dynamics and was therefore easier to implement in the simulation environment. With an improved tracking setup and a model free prediction, a shuttlecock could be used within the task in the future.
The learned badminton task is very limited to a certain situation, where the ball falls inside a predefined area of the hitting plane. The reason for this is, that only one type of striking motion has been taught by demonstration, which restricts the robot highly in its movements. At the current stage, an actual game of badminton against an opponent would not be possible. To widen the robot's capabilities, a skill library can be created that consists of multiple different hitting movements. In addition, the task could be extended to select a target. This would enable the robot to return the shuttlecock to different positions on the court. In Section 5.6 we suggested the learning of a decision making policy that would select, depending on the current situation, i.e. the opponents positioning, the best type of movement from the skill library, the optimal desired target and the duration of the prediction horizon. Finding a suitable objective, that could include risk-sensitivity, for optimizing the decision making policy is an interesting research topic.

6.2.3 Risk-Sensitive Movement Primitives

An interesting direction for research is the introduction of risk into movement primitive frameworks. Robots performing movements with high variances can be interpreted as being risky, because their behavior is uncertain. In some situations, for example when working together with a human, a risky movement should be prohibited to guarantee safety. Parts of the trajectory that are critical to safety and performance, should have a high certainty, while other areas that are not vital could be allowed a higher variance. In future research it might be worthwhile to study if movement primitives can be learned with risk-sensitive RL methods. ProMPs could be a candidate to look into, as they are a probabilistic representation of movements.

Bibliography

[1] K. Čapek, *RUR (Rossum's universal robots)*. Penguin, 2004.

[2] H. Simon, "The shape of automation for men and management," *Harper and Row*, vol. 40, 1965.

[3] R. S. Sutton, A. G. Barto, F. Bach, *et al.*, *Reinforcement learning: An introduction*. MIT press, 1998.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[5] M. P. Deisenroth, G. Neumann, J. Peters, *et al.*, "A survey on policy search for robotics," *Foundations and Trends® in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[6] R. Bellmann, "Dynamic programming princeton university press," *Princeton, NJ*, 1957.

[7] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, pp. 1177–1184, 2008.

[8] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade, "Towards generalization and simplicity in continuous control," in *Advances in Neural Information Processing Systems*, pp. 6550–6561, 2017.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[10] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[11] S. M. Kakade, "A natural policy gradient," in *Advances in neural information processing systems*, pp. 1531–1538, 2002.

[12] J. Peters, K. Mülling, and Y. Altun, "Relative entropy policy search.," in *AAAI*, pp. 1607–1612, Atlanta, 2010.

[13] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[14] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2015.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[16] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, no. 3, pp. 529–551, 2018.

[17] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[18] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[19] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *Advanced Robotics*, vol. 21, no. 13, pp. 1521–1544, 2007.

[20] S. Schaal, "Dynamic movement primitives–a framework for motor control in humans and humanoid robotics," 2002.

[21] J. Kober, K. Mülling, O. Krömer, C. H. Lampert, B. Schölkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *2010 IEEE International Conference on Robotics and Automation*, pp. 853–858, May 2010.

[22] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in neural information processing systems*, pp. 2616–2624, 2013.

[23] R. E. Kalman *et al.*, “Contributions to the theory of optimal control,” *Bol. Soc. Mat. Mexicana*, vol. 5, no. 2, pp. 102–119, 1960.

[24] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[25] R. Schneider and C. Georgakis, “How to not make the extended kalman filter fail,” *Industrial & Engineering Chemistry Research*, vol. 52, no. 9, pp. 3354–3362, 2013.

[26] P. Whittle, “Risk sensitivity, a strangely pervasive concept,” *Macroeconomic Dynamics*, vol. 6, no. 1, pp. 5–18, 2002.

[27] H. Markowitz, “Portfolio selection,” *The journal of finance*, vol. 7, no. 1, pp. 77–91, 1952.

[28] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath, “Coherent measures of risk,” *Mathematical finance*, vol. 9, no. 3, pp. 203–228, 1999.

[29] D. Jacobson, “Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games,” *IEEE Transactions on Automatic control*, vol. 18, no. 2, pp. 124–131, 1973.

[30] P. Whittle, “Risk-sensitive linear/quadratic/gaussian control,” *Advances in Applied Probability*, vol. 13, no. 4, pp. 764–777, 1981.

[31] A. Tamar, *Risk-Sensitive and Efficient Reinforcement Learning Algorithms*. PhD thesis, Israel Institute of Technology, 2015.

[32] W. F. Sharpe, “The sharpe ratio,” *Journal of portfolio management*, vol. 21, no. 1, pp. 49–58, 1994.

[33] J. Peters and S. Schaal, “Reinforcement learning by reward-weighted regression for operational space control,” in *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, ACM, 2007.

[34] J. Kober and J. Peters, “Learning motor primitives for robotics,” in *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*, pp. 2112–2118, IEEE, 2009.

[35] S. Lall, “Stochastic control.” `https://stanford.edu/class/ee365/lectures.html`, 2014.

[36] R. A. Schmidt and C. A. Wrisberg, *Motor learning and performance: A situation-based learning approach*. Human kinetics, 2008.

[37] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, “Reinforcement learning to adjust parametrized motor primitives to new situations,” *Autonomous Robots*, vol. 33, no. 4, pp. 361–379, 2012.

[38] K. Mülling, J. Kober, O. Kroemer, and J. Peters, “Learning to select and generalize striking movements in robot table tennis,” *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

A Appendix

In the Appendix additional material and derivations are given.

A.1 Gradient of Gaussian Policy

For the Gaussian policies introduces in 2.2.1, we present the gradients with respect to the parameters $\boldsymbol{\omega}$. First, we look at the policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \Sigma)$, where the mean is part of $\boldsymbol{\omega} = \{\boldsymbol{\mu}, \Sigma\}$ [5]

$$\nabla_{\boldsymbol{\mu}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) = (\boldsymbol{\theta} - \boldsymbol{\mu})^{\mathrm{T}} \Sigma^{-1} \tag{A.1}$$

If we assume that the covariance matrix is a diagonal matrix $\Sigma = \operatorname{diag}(\boldsymbol{\sigma}^2)$, the gradient simplifies and can be written for each dimension d as follows:

$$\nabla_{\mu_d} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) = \frac{(\theta_d - \mu_d)}{\sigma_d} \tag{A.2}$$

The gradient with respect to each diagonal element of the covariance matrix is:

$$\nabla_{\sigma_d} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) = -\frac{1}{\sigma_d^2} + \frac{(\theta_d - \mu_d)^2}{\sigma_d^3} \tag{A.3}$$

If the mean of the policy is dependent on the states and given as $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{M}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$, the gradient w.r.t. $\mathbf{M}$ is

$$\nabla_{\boldsymbol{M}} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s}) = \boldsymbol{\phi}(\mathbf{s})(\boldsymbol{\theta} - \mathbf{M}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}))^{\mathrm{T}} \Sigma^{-1} \tag{A.4}$$

Assuming a diagonal covariance matrix and $\mathbf{m}_d$ being the d-th column of $\mathbf{M}$:

$$\nabla_{\boldsymbol{m}_d} \log \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s}) = \frac{\boldsymbol{\phi}(\mathbf{s})(\theta_d - \mathbf{m}_d^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}))}{\sigma_d} \tag{A.5}$$

The gradient for σ_d is analogous to A.3 with $\mu_d = \mathbf{m}_d^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s})$.

A.2 Weighted Maximum Likelihood Estimates of Gaussian Policy

In this section we introduce the weighted maximum likelihood solutions for the policies presented in 2.2.1. For the policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \Sigma)$ the weighted EM solution for $\boldsymbol{\omega} = \{\boldsymbol{\mu}, \Sigma\}$ is [5]

$$\boldsymbol{\mu} = \frac{\sum_{i=1}^{N} d^{[i]} \boldsymbol{\theta}^{[i]}}{\sum_{i=1}^{N} d^{[i]}} \tag{A.6}$$

$$\boldsymbol{\Sigma} = \frac{\sum_{i=1}^{N} d^{[i]} (\boldsymbol{\theta}^{[i]} - \boldsymbol{\mu})(\boldsymbol{\theta}^{[i]} - \boldsymbol{\mu})^{\mathrm{T}}}{Z}, \tag{A.7}$$

where

$$Z = \frac{\left(\sum_{i=1}^{N} d^{[i]}\right)^2 - \sum_{i=1}^{N} (d^{[i]})^2}{\sum_{i=1}^{N} d^{[i]}}. \tag{A.8}$$

The $d^{[i]}$ defines the weightings for each sample i and N is the total number of samples.
For the contextual policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\mathbf{s}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{M}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ the weighted EM solution is

$$\mathbf{M}_{\text{new}} = (\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{D}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{D}\boldsymbol{\Theta}, \tag{A.9}$$

where $\mathbf{D} = \operatorname{diag}(d^{[1]}, ..., d^{[N]})$ and $\boldsymbol{\Theta} = [\boldsymbol{\phi}(\mathbf{s}^{[1]}) \dots \boldsymbol{\phi}(\mathbf{s}^{[N]})]$. For the covariance matrix the solution is

$$\boldsymbol{\Sigma} = \frac{\sum_{i=1}^{N} d^{[i]}(\boldsymbol{\theta}^{[i]} - \mathbf{M}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}^{[i]}))(\boldsymbol{\theta}^{[i]} - \mathbf{M}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s}^{[i]})^{\mathrm{T}}}{Z}, \tag{A.10}$$

where Z is given by (A.8).

A.3 Connection Mean-Variance to Exponential Risk Measure

This section clarifies that the mean-variance and exponential risk-measure are closely related. The mean-variance risk measure is given by,

$$J_{\mathrm{m/v}} = \mathbb{E}[C] + \frac{\gamma}{2}\mathrm{Var}[C] \tag{A.11}$$

and the exponential risk measure by,

$$J_{\exp} = \frac{1}{\gamma}\log \mathbb{E}[\exp(\gamma C)]. \tag{A.12}$$

We now use the Taylor-Series expansions to expand the exponential risk measure. The Taylor-Series expansion of $\exp x$ is given by:

$$\exp x = 1 + \frac{x}{1!} + \frac{u^2}{2!} + \frac{u^3}{3!} + \dots \tag{A.13}$$

The Taylor-Series expansion of $\log(x+1)$ is given by:

$$\log(x+1) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} + \dots \tag{A.14}$$

Using (A.13) on (A.12) results in:

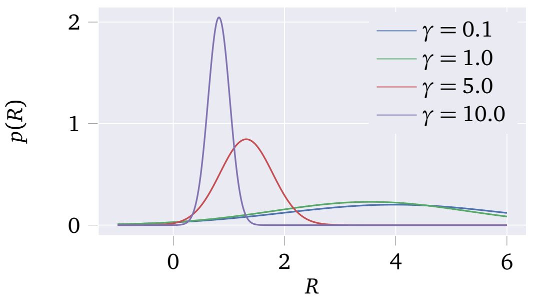
$$J_{\exp} = \frac{1}{\gamma}\log\left(1 + \mathbb{E}\left[\frac{\gamma C}{1!}\right] + \mathbb{E}\left[\frac{\gamma^2 C^2}{2!}\right] + \mathbb{E}\left[\frac{\gamma^3 C^3}{3!}\right] + \dots\right) \tag{A.15}$$

Using (A.14) on (A.15) and dropping terms with order higher γ^2, results in:

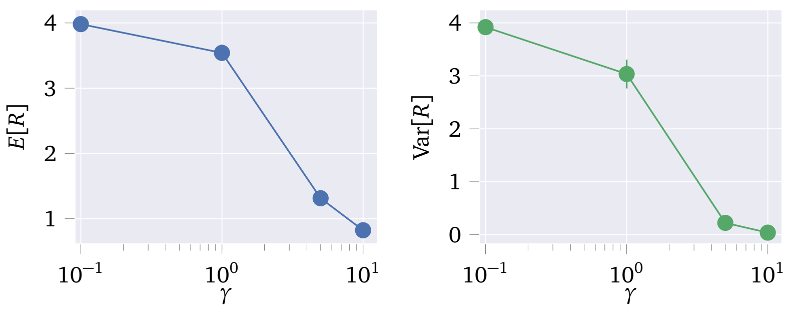
$$\begin{aligned} J_{\exp} &\approx \frac{1}{\gamma}\left(\mathbb{E}\left[\frac{\gamma C}{1!}\right] + \mathbb{E}\left[\frac{\gamma^2 C^2}{2!}\right] - \frac{1}{2}\left(\mathbb{E}\left[\frac{\gamma C}{1!}\right] + \mathbb{E}\left[\frac{\gamma^2 C^2}{2!}\right]\right)^2\right) \\ &\approx \mathbb{E}[C] + \frac{\gamma}{2}\mathbb{E}[C^2] - \frac{\gamma}{2}\mathbb{E}[C]^2 + \dots \\ &\approx \mathbb{E}[C] + \frac{\gamma}{2}\mathrm{Var}[C] + O(\gamma), \end{aligned} \tag{A.16}$$

where $O(\gamma)$ includes all higher order terms of γ. The higher order terms can be neglected for small γ, thus the exponential risk-measure corresponds to the mean-variance risk-measure for small γ.

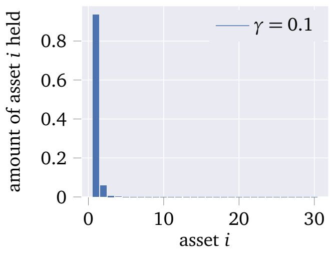
A.4 Portfolio Management



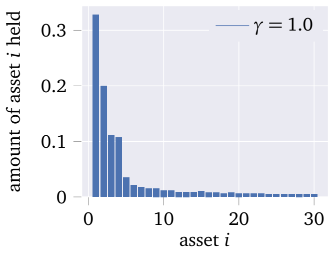
(a) Reward distribution



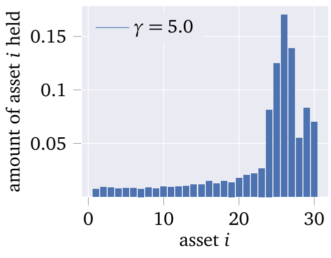
(b) Expected cumulated reward, $\mathbb{E}[R]$, and variance of the cumulated reward Var$[R]$ compared over different values of γ.



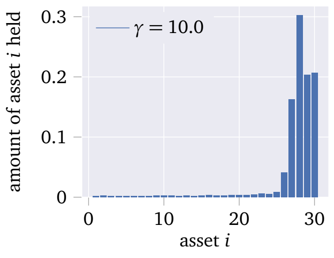
(c) Asset distribution, $\gamma = 0.1$



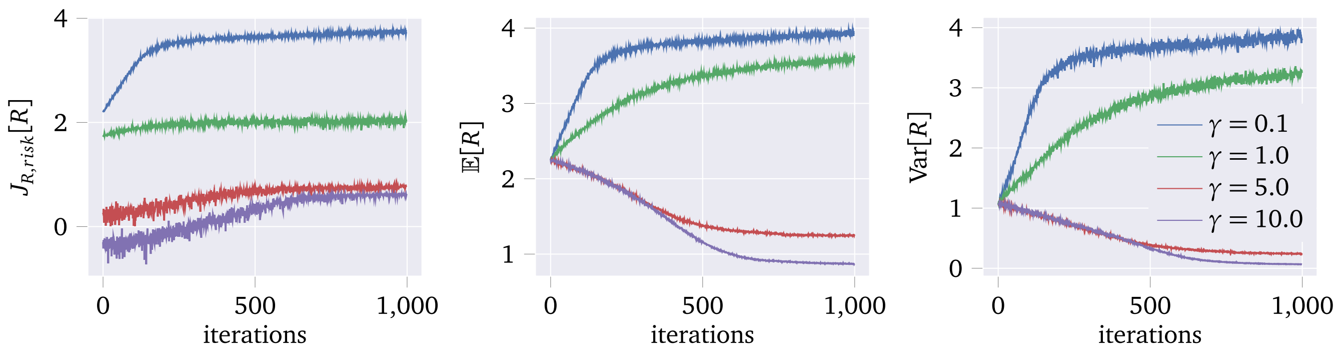
(d) Asset distribution, $\gamma = 1$



(e) Asset distribution, $\gamma = 5.0$

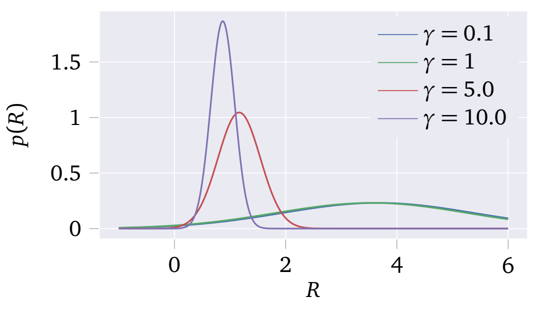


(f) Asset distribution, $\gamma = 10$

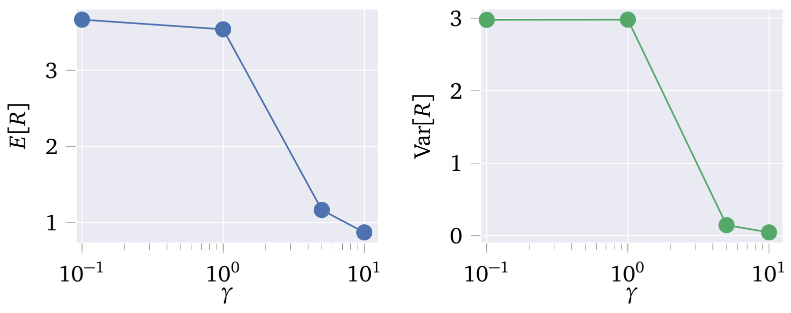


(g) Convergence rate of NPG for different γ.

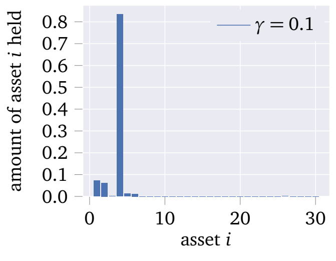
Figure A.1.: Portfolio Management example optimized with NPG and a risk-sensitive objective. Evaluated for different $\gamma \in \{0.1, 1, 5, 10\}$. Each roll-out consists of 1000 samples. For small γ the objective is close to being risk-neutral, the policy invests in assets with a higher expected reward. For increasing γ the penalty on the variance increases and the policies become risk-averse. The result is that investing in assets offering a lower expected reward but also having a lower variance, are preferable for the risk-averse case.



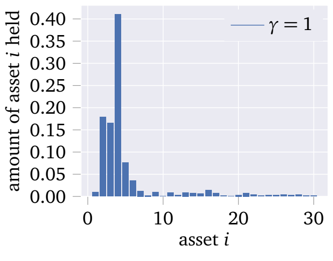
(a) Reward distribution



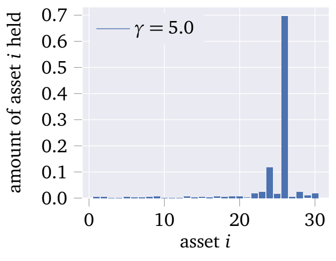
(b) Expected cumulated reward, $\mathbb{E}[R]$, and variance of the cumulated reward Var$[R]$ compared over different values of γ.



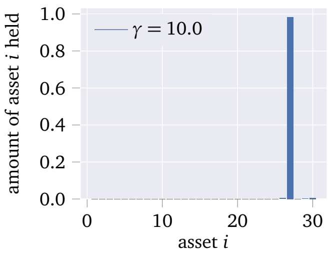
(c) Asset distribution, $\gamma = 0.1$



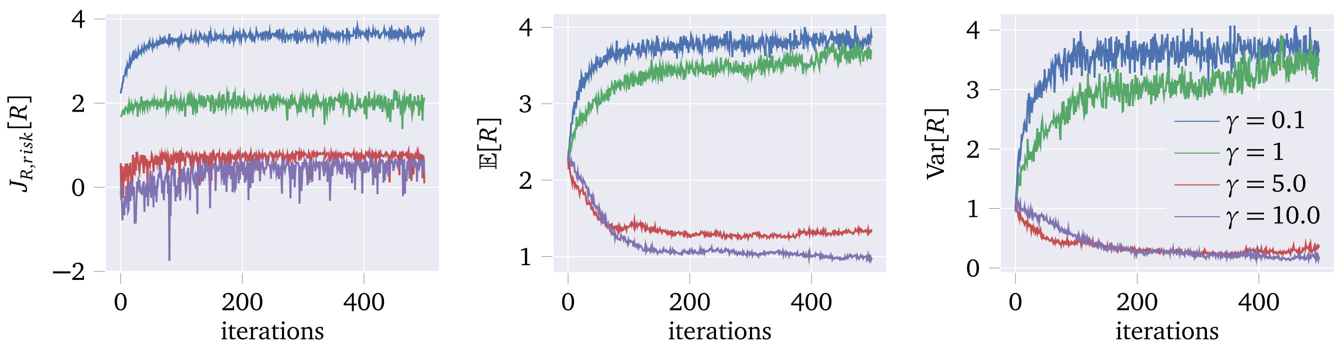
(d) Asset distribution, $\gamma = 1$



(e) Asset distribution, $\gamma = 5.0$

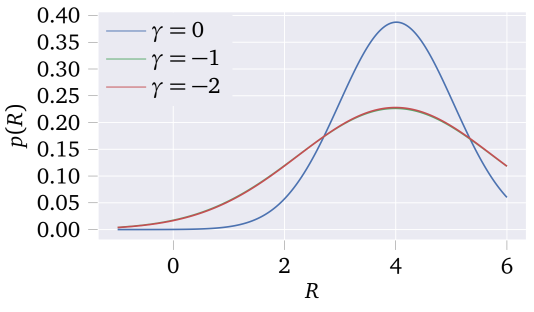


(f) Asset distribution, $\gamma = 10$

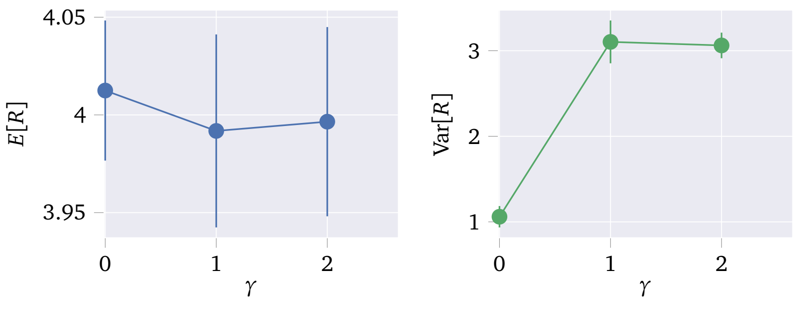


(g) Convergence rate of PPO for different γ.

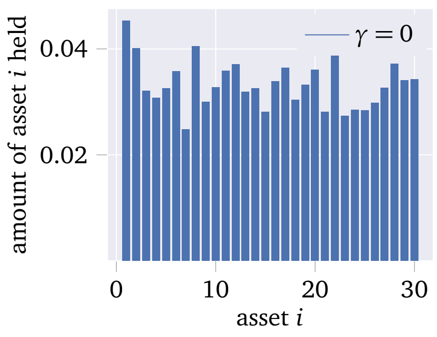
Figure A.2.: Portfolio Management optimized with PPO using a risk-sensitive objective. Evaluated for different $\gamma \in \{0.1, 1, 5, 10\}$. Each roll-out consists of 1000 samples. Results are similar to the ones observed in Figure A.1.



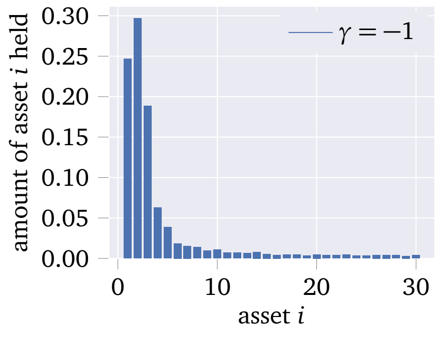
(a) Reward distribution



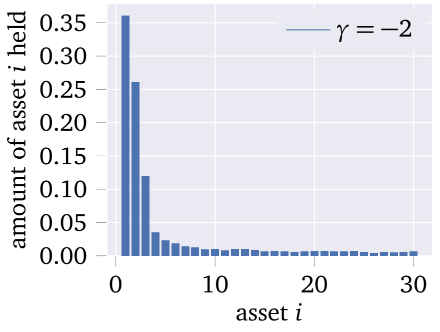
(b) $\mathbb{E}[R]$, Var$[R]$



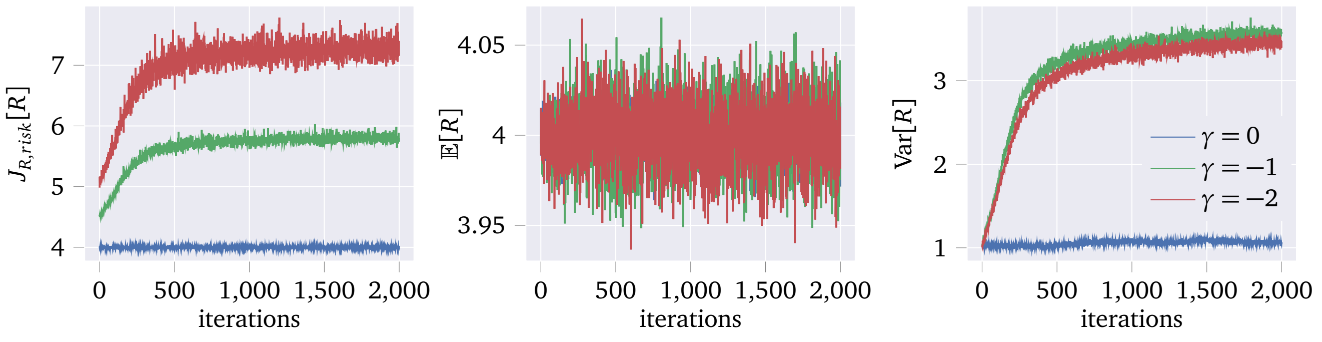
(c) Asset distribution, $\gamma = 0$



(d) Asset distribution, $\gamma = -1.0$



(e) Asset distribution, $\gamma = -2.0$



(f) Convergence rate of PG for different γ.

Figure A.3.: Portfolio Management optimized with PG using a risk-neutral/-seeking objective. Evaluated for different $\gamma \in \{0, -1, -2\}$. Each assets has the same expected return but differs in the variance. For a risk-neutral objective the policy cannot be improved by distributing the capital over different assets. The risk-seeking policy on the other hand invests in assets with higher variance.

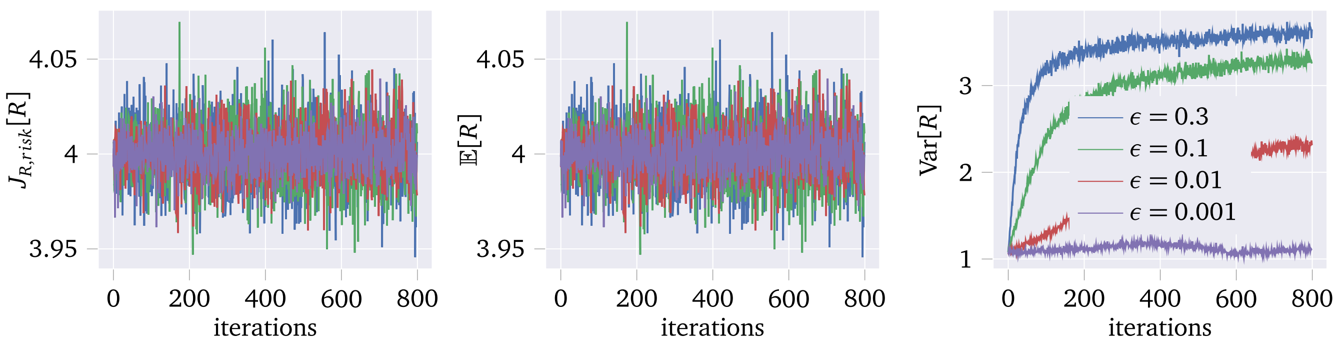
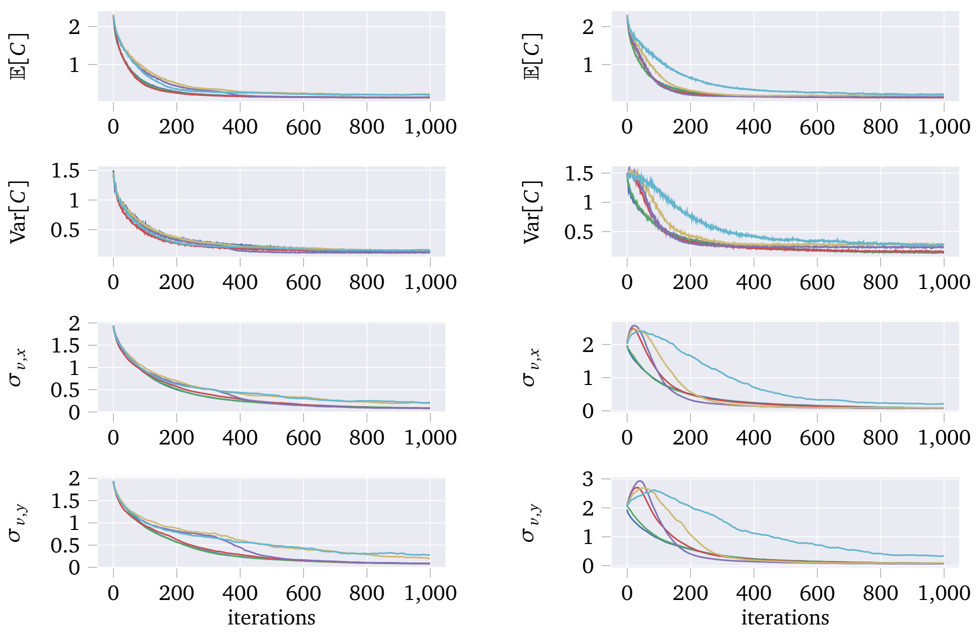


Figure A.4.: Convergence rate of REPS for the portfolio management problem with different ϵ for a risk-neutral objective. The final variance of the cumulated reward increases with the increase of ϵ. This leads to the conclusion that REPS is risk-seeking. As we are optimizing a risk-neutral objective, $J_{R,\text{risk}}[R] = \mathbb{E}[R]$.



(a) PPO with risk-averse objective

(b) PPO with risk-seeking objective

Figure A.5.: Convergence rate of PPO for the badminton toy-system problem for a risk-sensitive objective with different (γ. — $\gamma = 0.01$, — $\gamma = 0.1$, — $\gamma = 1$, — $\gamma = 10$, — $\gamma = 100$, — $\gamma = 1000$) The variables $\sigma_{v,x}$ and $\sigma_{v,x}$ represent the exploration noise of the Gaussian policy. In contrast to the risk-averse case, where the exploration decreases steadily, the exploration increases for strong risk-seeking objectives, before decreasing and converging. For $\gamma = -1000$ the exploration noise does not converge to a small value and still has a significant influence on the system.

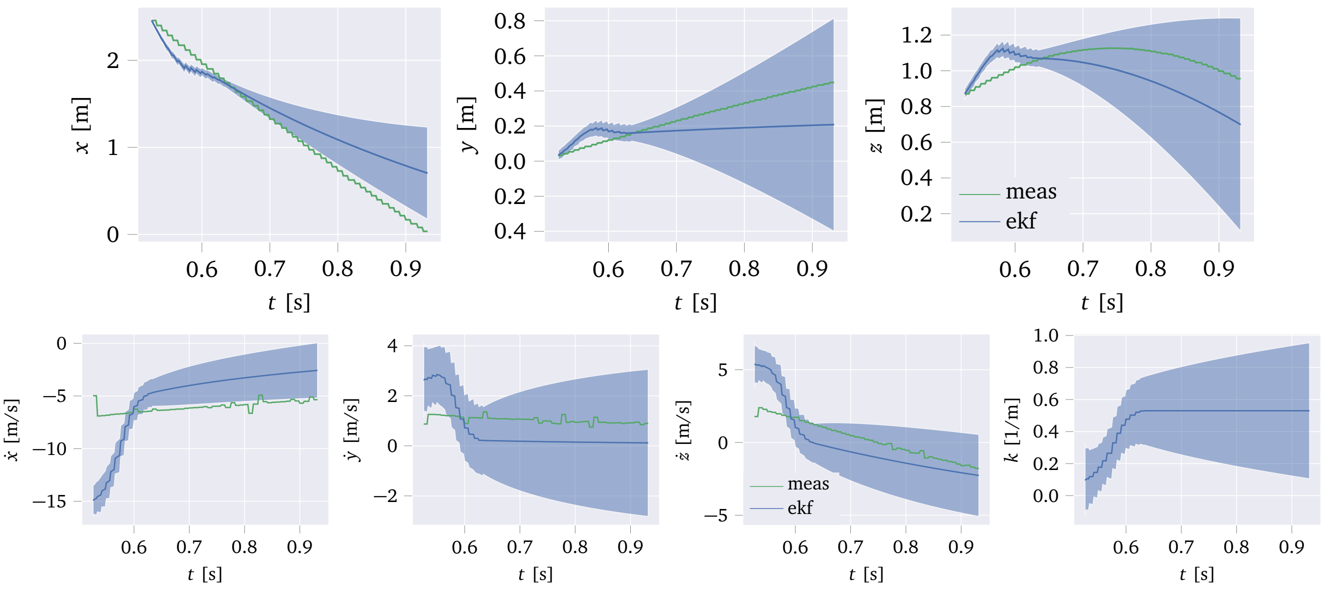


Figure A.6.: This figure shows a failed prediction of the ball using an EKF. The data was recored on the real robot setup. The initial velocities deviate too much from the real values and cannot be corrected by new measurements before the open loop prediction starts at $t \approx 0.65$s. Thus, an accurate estimation depends strongly on a good initialization of the states.