Safer reinforcement learning for robotics

Koryakovskiy, Ivan

DOI
10.4233/uuid:7923c257-e81f-4e29-adf7-bd6014d9da6a

Publication date
2018

Document Version
Final published version

Citation (APA)
Koryakovskiy, I. (2018). *Safer reinforcement learning for robotics*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:7923c257-e81f-4e29-adf7-bd6014d9da6a

Safer reinforcement learning for robotics

Ivan Koryakovskiy

Safer reinforcement learning for robotics

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Monday 3 December 2018 at 15:00 o'clock

by

Ivan KORYAKOVSKIY

Master of Science in Electrical Engineering and Computer Science,
Seoul National University, Seoul, South Korea
born in Sosnoviy Bor, Russia.

This dissertation has been approved by the promoters:

Prof. Dr.-Ing. H. Vallery
Prof. dr. R. Babuška

Composition of the doctoral committee:

Rector Magnificus	chairperson
Prof. Dr.-Ing. H. Vallery	Technische Universiteit Delft, promoter
Prof. dr. R. Babuška	Technische Universiteit Delft, promoter
Independent members:	
Prof. dr. M. Wisse	Technische Universiteit Delft
Prof. dr. A. Kheddar	Université de Montpellier
Dr. H. C. van Hoof	Universiteit van Amsterdam
Prof. dr. A. Nowé	Vrije Universiteit Brussel
Prof. dr. C. M. Jonker	Technische Universiteit Delft, reserve member
Other member:	
Dr. W. Caarls	Pontifícia Universidade Católica do Rio de Janeiro

The research presented in this thesis has received financial support from the European Commission's Seventh Framework Programme (FP7-ICT-2013-10) under grant agreement No. 611909.

Koryakovskiy I.
Safer reinforcement learning for robotics. – M.: Editus, 2018. – 170 p.

ISBN 978-5-00058-959-5

Email: I.Koryakovskiy@gmail.com

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`

Contents

Contents vii

List of symbols ix

List of abbreviations xiii

Preface xv

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Robot safety and learning . . . 3
- 1.3 Problem definition . . . 5
- 1.4 Research goal . . . 6
- 1.5 Approach . . . 7
- 1.6 Thesis outline . . . 8

2 Evaluation of physical damage associated with action selection strategies 11
- 2.1 Introduction . . . 12
- 2.2 Reinforcement learning . . . 13
  - 2.2.1 The Markov decision process . . . 13
  - 2.2.2 Action-selection methods . . . 14
- 2.3 Simulations results . . . 16
- 2.4 Discussion . . . 18
- 2.5 Conclusion . . . 22

3 Benchmarking model-free and model-based optimal control 25
- 3.1 Introduction . . . 26
- 3.2 Model-based and model-free optimal control methods . . . 29
  - 3.2.1 Optimal control . . . 29
  - 3.2.2 Nonlinear model predictive control . . . 29
  - 3.2.3 Reinforcement learning . . . 31
- 3.3 Benchmark system . . . 32
- 3.4 Problem formulation . . . 32
- 3.5 Evaluation protocol . . . 35
  - 3.5.1 Notations and methodology . . . 35
  - 3.5.2 Description of experiment and measures . . . 36
- 3.6 Results on the cart-pendulum . . . 37
- 3.7 Discussion . . . 41
- 3.8 Conclusion . . . 44

4 Model-plant mismatch compensation using reinforcement learning 45
4.1 Introduction . . . 46
4.2 Related work . . . 47
4.3 Background . . . 48
4.3.1 Problem statement . . . 48
4.3.2 Nonlinear model predictive control . . . 49
4.3.3 Reinforcement learning . . . 49
4.4 Proposed combination approaches . . . 50
4.4.1 Compensatory Action Learning . . . 50
4.4.2 Model-Plant Mismatch Learning . . . 50
4.5 Experiments . . . 52
4.5.1 Bipedal walking robot Leo . . . 52
4.5.2 Objective function and constraints . . . 52
4.5.3 Parameters . . . 53
4.5.4 Evaluation . . . 53
4.5.5 Simulation results . . . 53
4.5.6 Results on the real robot . . . 54
4.6 Discussion . . . 56
4.7 Conclusion . . . 59

5 Sample-efficient reinforcement learning via difference models 61
5.1 Introduction . . . 62
5.2 Related work . . . 62
5.3 Reinforcement learning . . . 63
5.4 Proposed method . . . 64
5.4.1 Notation . . . 64
5.4.2 Algorithm . . . 65
5.4.3 Training data . . . 65
5.5 Experiment details . . . 66
5.5.1 Inverted pendulum . . . 66
5.5.2 Bipedal walking robot Leo . . . 66
5.5.3 Training data and parameters . . . 68
5.5.4 Evaluation measures . . . 68
5.6 Results . . . 69
5.6.1 Inverted pendulum . . . 69
5.6.2 Robot Leo . . . 69
5.7 Discussion . . . 71
5.8 Conclusion . . . 74

6 Multitask reinforcement learning for safer acquisition of locomotion skills 77
6.1 Introduction . . . 78
6.2 Background . . . 81
6.2.1 Reinforcement learning . . . 81
6.2.2 Model-free deep reinforcement learning . . . 82

6.3 Proposed method . . . 83
6.3.1 Curriculum learning . . . 83
6.3.2 Supervised learning of the task-switching network . . . 85
6.4 Experiment details . . . 87
6.4.1 Systems . . . 87
6.4.2 Learning parameters . . . 88
6.4.3 Evaluation methodology . . . 90
6.4.4 Evaluation metrics . . . 91
6.5 Results . . . 92
6.5.1 TIM: curriculum learning with manual selection of time steps to practice . . . 92
6.5.2 BAL: automated curriculum learning with the duration of balancing as a task-switching indicator . . . 97
6.5.3 RNN: automated curriculum learning with RNN-based identification of task-switching moments . . . 98
6.6 Discussion . . . 99
6.7 Conclusion . . . 102

7 Conclusions and future directions **105**
7.1 Conclusions . . . 106
7.1.1 Influence of exploration strategies . . . 106
7.1.2 Safer learning with an approximate dynamical model . . . 107
7.1.3 Safer learning without the approximate dynamical model . . . 108
7.2 Directions for future research . . . 109
7.2.1 Composite approach towards damage minimization . . . 109
7.2.2 Future directions for safer reinforcement learning research . . . 110

Acknowledgements **115**

A Appendix. Experimental setups **117**
A.1 Bipedal walking robot Leo . . . 118
A.2 The inverted pendulum on a movable cart . . . 120

B Appendix. Additional results **123**
B.1 Influence of the reward shaping on the trajectory cost . . . 124
B.2 Influence of the discount rate on learning with MPML . . . 124
B.3 The mass distribution of Leo and Roboschool systems vs. human . . . 126
B.4 Curriculum learning with samples obtained from different models . . . 126

Bibliography **129**

List of publications **143**

Summary **145**

Samenvatting **149**

Curriculum Vitae **153**

List of symbols

Standard math symbols

xOy	Cartesian coordinate system with center in O
$\nabla_{\boldsymbol{a}} f$	gradient of function f w.r.t. vector $\boldsymbol{a}$
$\left\|\boldsymbol{a}\right\|_p$	ℓ_p-norm of vector $\boldsymbol{a}$
$\left\|\boldsymbol{a}\right\|_{\mathbf{W}}$	$\sqrt{\boldsymbol{a}^\top \mathbf{W} \boldsymbol{a}}$, the ℓ_2-norm of $\boldsymbol{a}$ weighted with a positive definite matrix $\mathbf{W}$
$\boldsymbol{a}^\top, \mathbf{A}^\top$	transpose of vector $\boldsymbol{a}$ or matrix $\mathbf{A}$
$\mathbb{R}$	set of real numbers
$\mathbb{E}$	expectation
p	probablility
$\boldsymbol{\Sigma}$	covariance matrix
$\mathcal{N}, \mathcal{M}$	random processes
$\boldsymbol{n}, \boldsymbol{m}$	random variables
j	vector element
n	size of vector
N	dataset size

Physical variables

$s, \dot{s}, \ddot{s}$	position (m), linear velocity ($\mathrm{m\,s^{-1}}$) and acceleration ($\mathrm{m\,s^{-2}}$)
$\phi, \dot{\phi}, \ddot{\phi}$	angle (rad), angular velocity ($\mathrm{rad\,s^{-1}}$) and acceleration ($\mathrm{rad\,s^{-2}}$)
$\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$	generalized position, velocity and acceleration
$\boldsymbol{F}$	generalized force
$\mathbf{H}$	mass matrix
$\boldsymbol{B}$	vector with Coriolis, centrifugal, and gravitational terms
g	gravitational acceleration ($\mathrm{m\,s^{-2}}$)
t	continuous time (s)
T_s	sampling period (s)
c	center of mass
m	body mass (kg)
l	length (m)

J	moment of inertia ($\mathrm{kg\,m^2}$)
μ	viscous friction coefficient ($\mathrm{N\,s\,m^{-2}}$)
κ	friction area ($\mathrm{m^2}$)
τ	torque ($\mathrm{N\,m}$)
F	force (N)
E	motor work ($\mathrm{J\,m^{-1}}$)
S	travel distance (m)
I	current (A)
U	voltage (V)
R	resistance (Ω)
K_τ	torque constant ($\mathrm{N\,m\,A^{-1}}$)
K_G	gearbox ratio
K_hip	stiffness of hip joint ($\mathrm{N\,m\,rad^{-1}}$)
C_u	control scale factor
τ_knee	knee temperature (°C)
$\boldsymbol{p}$	model parameters
$\boldsymbol{\rho}$	uncertainty in structure or model parameters

Optimal control and moving horizon estimation

T	finite horizon (s)
$\mathcal{T}$	finite time interval
L	integral (Lagrange) term of objective function
M	terminal (Mayer) term of objective function
$\boldsymbol{y}$	measurement of state $\boldsymbol{x}$
$\mathbf{W}$	state weighting matrix
$\mathbf{V}$	control weighting matrix

MDPs and RL

$\mathcal{X}$	continuous state space
$\mathcal{U}$	continuous action space
$\mathbb{U}$	discrete action space
$\mathcal{P}$	transition function
$\mathcal{R}$	reward function
$\boldsymbol{x}$	state
$\bar{\boldsymbol{x}}$	reference trajectory

$\boldsymbol{u}$	action/control
$\Delta\boldsymbol{u}$	control discretization step
r	reward
r^{a}	reward given in absorbing states
G	return
R	regret
π	policy
π^*	optimal policy
V^π	state value function for policy π
Q^π	state-action value function for policy π
q	value of the state-action value function
k	discrete time
K	number of time steps in finite horizon
ζ	trajectory (a sequence of states and controls)
α	learning rate
γ	reward discount factor
ϵ	exploration rate
$\sigma, \theta, \boldsymbol{m}_0$	Ornstein-Uhlenbeck process parameters
λ	target networks update weight
$\boldsymbol{\theta}^{\mathrm{a}}, \boldsymbol{\theta}^{\mathrm{c}}$	actor and critic parameters
Ω	squashing function
ω	squashing factor
P	potential function
Ψ	shaping function
ψ	shaping weight
$\mathcal{J}$	termination indicator
$\mathcal{D}$	difference model
$\mathbb{B}$	experience replay buffer

Curriculum learning

τ	task
$\mathbb{T}$	collection of tasks τ
Δr	task difference (expressed by the difference in reward)
$\boldsymbol{I}$	learning performance indicators
$\boldsymbol{\theta}^{\mathrm{s}}$	switching network parameters
k_{w}	sliding window width
ϵ^{CE}	Cross-Entropy percentile

Additional performance measures

$\mathcal{L}$	cost of trajectory ζ
$\boldsymbol{e}$	model-plant mismatch
$\mathcal{E}$	cumulative model-plant mismatch
$\mathcal{F}$	gearbox fatigue
$N^{\mathcal{F}}$	number of completely reversed cycles withstood before failure
ξ	exploration intensity
Ξ	cumulative mean exploration intensity

List of abbreviations

CAL	Compensatory Action Learning
CE	Cross-Entropy method
DDPG	Deep Deterministic Policy Gradient
DNN	deep neural network
DoF	degree of freedom
DPG	deterministic policy gradient
IMC	internal model control
MHE	moving horizon estimation
ML	machine learning
MPML	Model-Plant Mismatch Learning
MSE	mean squared error
MTBF	mean time before failure
NMPC	nonlinear model predictive control
OC	optimal control
ODE	Open Dynamics Engine
OU	Ornstein-Uhlenbeck noise
PADA	Previous Action-Dependent Action algorithm
PD	proportional-derivative controller
RBDL	Rigid Body Dynamics Library
RL	reinforcement learning
RMSE	root mean squared error
RNN	recurrent neural network

Preface

We live in a very exiting period of human history when automation has a true potential to liberate human from the hardships of the labor for survival. Concurrently with the diffusion of automation, it is important to continue developing our social and economic systems to prevent disproportional amplification of the already existing inequalities in the society and share automation benefits with all people and countries around the world.

Ivan Koryakovskiy
Delft, June 2018

1

Introduction

1.1 Motivation

Machine learning (ML) is an artificial intelligence discipline. Given the input and/or output data, ML tries to discover the underlying process which generated the data. Such knowledge usually has regularities and structure, and this is what ML strives to find. It can substitute or augment human in areas where tedious, computationally intensive or dangerous and at the same time intelligent activities are required. Various supervised and unsupervised learning algorithms have been developed for classification and recognition, automatic translation and web-search, content recommendation and pharmaceutical drug development (Alpaydın, 2014). Nowadays, many fields of human activity with access to large amounts of data use these algorithms to help people take faster actions of better quality.

Despite its efficiency, supervised and unsupervised learning have one critical bottleneck – namely, they demand human-provided data. It is often difficult to obtain training data, and data may also be subjective in cases when it is labeled manually. Reinforcement learning (RL) is another subfield of ML which aims at developing intelligent agents capable of making decisions, acting upon them and, most importantly, to learn from the outcomes of these decisions. Therefore, RL provides the self-learning framework of the underlying process.

Since their emergence in the late 1950s, the robots were controlled by classical approaches which flawlessly execute a set of commands to achieve a certain task. Unfortunately, application of these methods is limited to controlled environments such as production lines or factories, or continuous dynamical systems, e.g. aerospace. ML has a great potential to expand the number of real-life applications in robotics. Already there are intelligent robots that can autonomously clean the floor, monitor surroundings or entertain people, e.g. Ubtech's bipedal walker, Honda 3E-A18. There are also numerous start-up companies that aim at substituting people working in hazardous environments (e.g. robot Talos from PAL Robotics) or increase human productivity and safety (self-driving cars). However, simulated environments allow to learn more complex tasks or use more complex robots. The problem of transferring the learned policies from simulators to the real world or learning policies from scratch in the real world is very challenging due to several factors.

First, learning from experience involves making mistakes. RL is a trial-and-error process which requires a good balance between exploration and exploitation. In RL, exploration is often achieved by adding random noise directly to the actions or indirectly to policy parameters. A similar mechanism may be found in humans. For example, Wu et al. (2014) showed that actively regulated biological motor variability facilitates human motor learning. Just like with people, noisy actions may damage a robot or its surroundings. Depending on the severity of the damage, repair may be very costly. If the models of the robot and its surroundings are available, a possible solution may be to detect dangerous situations and restrict the robot from taking unneeded risk. This strategy guarantees a minimum damage, and therefore it is widely used in commercially available robots mentioned above. However, for some applications, such strategy may come at a price of greatly limiting the poten-

tial of learning, thereby hampering the quality of service provided by the robot.

Second, the real world is stochastic and much more diverse than it is assumed during modeling. Obtaining exact models of various physical effects such as contact impacts, backlash, friction or sliding is a complicated undertaking. Moreover, even if such models were known, simulating them would require an exceptional processing power. When this complication is coupled with a large variety of environment realizations and external disturbances, it becomes obvious that simulating every detail of reality is simply infeasible. A typical solution to this problem is to standardize the robot operating environment, e.g. to ensure the flatness of the floor, fixed positions of objects and specific lighting (Sünderhauf et al., 2018). These assumptions complicate the operation of robots in a diverse, dynamic and unstructured environment. Therefore, the ability to generalize a control policy to uncertainties in the environment is of high demand.

ML can achieve human-level performance in specific intelligent tasks, and in some tasks such as playing Atari games (Mnih et al., 2015) or Go (Silver et al., 2017) it outperforms most people. This trend is expected to continue. Thus, there is a need to develop new approaches towards a safer learning in order to benefit from the full potential of ML algorithms adopted in the real world.

1.2 Robot safety and learning

Garcia and Fernandez (2015) provide an in-depth overview of safe RL. Formally, it is possible to guarantee safe learning in a very limited number of cases: one can either predict repercussions of bad actions (Moldovan and Abbeel, 2012; Fulton and Platzer, 2018) or have a backup policy to lead the system back to the safe state (Hans et al., 2008; Fisac et al., 2017). The worst-case criterion (Heger, 1994) or the risk-sensitive criterion (Howard and Matheson, 1972; Geibel and Wysotzki, 2011) adopted from classical control approaches focus on risk-avoiding policies, which can be very restrictive in practice. Furthermore, these methods usually require an accurate model of the system. Without such a model, learning or fine-tuning the control policy on the real system is often necessary. In many cases, this process is more damaging than the execution of an optimal policy; therefore, learning algorithms should account for damage to avoid system failures. The review of literature summarizes the ideas that facilitate learning on real robots and also some promising simulated results.

It is commonly assumed that quicker learning reduces the damage of robots. A straightforward model-based technique is to learn the suboptimal policy in a simulator and then transfer the policy directly to a real robot (Jakobi, 1998; Svinin et al., 2001; Oßwald et al., 2010). However, in most cases, one encounters a simulation bias which does not allow to apply the policy directly to the real robot. Therefore, additional learning is usually required (Endo et al., 2008). Further reduction of sample complexity may be achieved by learning the difference between the simulator and the real world and incorporating the difference into the policy-learning loop (Morimioto et al., 2003; Abbeel et al., 2006; Cutler et al., 2014; Chatzilygeroudis and Mouret, 2017) or nonlinear model predictive control (NMPC) loop (Ostafew

et al., 2016). Similarly, this difference can be captured by the specially developed neural network structures called progressive nets (Rusu et al., 2017). In the model-free setting, a small number of dimensions and relatively simple tasks allow to learn the full probabilistic model of real dynamics and derive the policy from it (Deisenroth et al., 2015; Gal et al., 2016). The latter approach was recently scaled up to more complex tasks and dataset sizes (Higuera et al., 2018).

Learning from high-dimensional observations often requires dimensionality reduction as a part of data processing. State representation aims at finding the compact description of physical aspects relevant to the learning task. The efficiency of several architectures such as deep autoencoders (Lange and Riedmiller, 2010; Lange, 2010), robotic priors (Jonschkowski and Brock, 2015), and convolutional neural networks (Finn et al., 2015) was demonstrated in the end-to-end learning on real robots. Furthermore, disentangled state representations facilitate the generalization of control policies. Higgins et al. (2017) validated the possibility of transferring the learned visual representation from a simulator to a real robot without retraining.

Alternative techniques gain computational speedup from parallelizing learning algorithms across multiple robots (Levine et al., 2017; Gu et al., 2017). An additional benefit is that damage is also spread across all the robots. Physiological studies of infant development inspire another technique called curriculum learning. Asada et al. (1996) and Andrychowicz et al. (2017) demonstrated how learning could quickly progress if the robot proceeds to a more complex task after mastering the easier one.

Another popular technique of accelerating the learning process and at the same time reducing the damage is to parametrize controllers using smooth classes of functions. If the structure of the system is available, it is possible to derive the controller and then tune its parameters online (Calandra et al., 2016; Marco et al., 2016). Otherwise, initialization from human expert demonstrations can facilitate learning the control policy, which is especially useful for complex tasks. Reduction of the search space using a spline-based (Miyamoto et al., 1995), central pattern generator-based (Matsubara et al., 2006) or dynamic movement primitive-based (Peters and Schaal, 2008; Pastor et al., 2009; Kober and Peters, 2011) parametrization of trajectories is an effective learning technique. Demonstrations can also be used for learning the locally correct dynamical model (Levine and Koltun, 2013) or for fitting model parameters (Abbeel et al., 2010). Alternatively, experts can be substituted by a suboptimal controller available at hand (Kohl and Stone, 2004; Schuitema, 2012; Farshidian et al., 2014; Zhang et al., 2016). Furthermore, one can reduce the dimensionality of the robot by decreasing the number of actuated joints and exploiting the symmetry of a task (Schuitema, 2012). For dealing with even more complex tasks, the hierarchical strategy of sequencing low-level controllers for manipulation (Stulp and Schaal, 2011) or walking (Heess et al., 2016) has shown its effectiveness.

Even when a basic locomotion controller is available, navigation in a real-world environment is dangerous to a robot. Traditional approaches build geometric models using depth sensors and stereo cameras (Bachrach et al., 2009; Schmid et al., 2013). Another promising approach developed by Sadeghi and Levine (2016) is to

avoid robot collisions with obstacles by learning to predict collisions from a simulator with highly randomized images of walls and stairs.

A smaller number of approaches explicitly consider the risk of damage while learning. Mihatsch and Neuneier (2002) and Shen et al. (2014) derive risk-sensitive RL methods by modifying state or action value functions. Unfortunately, risk-sensitive RL may have undesirable effects such as the distortion of the long-term utility, overly pessimistic policies (Garcia and Fernandez, 2015), and additional parameters. Therefore, the risk-directed exploration forms an attractive alternative to risk-sensitive RL. Schuitema (2012) distinguishes between positive and negative rewards to quickly learn about dangerous states. Modifying exploration process based on temporal-difference error as a risk measure (Gehring and Precup, 2013) or incorporating prior assumptions about damaging actions (Meijdam et al., 2013; Van Hoof et al., 2017b,a) demonstrates faster and/or safer learning. The model-free approach of Mannucci et al. (2018) uses prior knowledge of fatal states and a data-driven backup controller to avoid collisions with obstacles. Alternatively, if the dynamical model is available, it is possible to increase the robustness of the policy by training it for the wide range of simulated models with varied model parameters. Rajeswaran et al. (2017), Yu et al. (2017) and Clavera et al. (2018) implement this approach and demonstrate the generalization of the policy beyond the observed model parameters.

Mechanical design of robots is another important factor that contributes to safer learning. Lightweight and small-sized walking robots reduce heel-strike or fall impacts (Schuitema, 2012; Calandra et al., 2016). Impact forces can also be reduced by compliant actuators integrated into the robot body (Albu-Schaeffer et al., 2008; Cardona et al., 2016). High endurance industrial manipulators allow setups where learning may last for several months (Levine et al., 2017). Finally, statically stable mobile robots provide a secured learning platform (Lange, 2010; Sutton et al., 2011; Jonschkowski and Brock, 2015). Such platforms are convenient for testing RL but also somewhat restrictive – for example, they do not allow learning intrinsically unstable tasks such as dynamic walking.

1.3 Problem definition

Classical control techniques are usually applied in structured environments such as research labs, factories, or Earth's atmosphere and the space beyond. It is believed that learning can bridge the gap between unstructured human-friendly environments and robots. However, learning requires exploration actions, which are potentially unsafe, and as such, the intention of staying safe is almost antithetical to learning. Adventurous exploration actions increase chances of mechanical damage because abnormal operating conditions increase the intensity of wear and tear and can easily lead to catastrophic failures in unstructured environments. The problem is exacerbated by the model-free learning algorithms, which usually do not constrain exploration and therefore may suffer from hundreds of failures before something meaningful is learned. Needless to say, each failure requires a significant amount of resources to restore the failed system back to the operating condition.

Literature presented in the previous section sheds some light on the existing approaches towards safety in robotics. The commonly accepted belief among computer scientists is that quicker learning will prevent damage to robots and their surroundings. In particular, developing one-shot learning approaches is the crucial step in the direction towards safety. Sufficient amount of prior knowledge is the necessary condition for one-shot learning. However, it may also hamper the learning performance. Therefore, the study of the techniques that achieve learning goals with the minimum of prior knowledge and damage is in high demand.

For example, an approximate model of the real system is often available. However, policies trained on it during simulation do not generalize well across similar models or real robots. Essentially, the current status of RL is such that transferred policies fail, often to the extent of not being able to produce any reliable control sequence at all. Therefore, additional learning on the real system is usually required. More efficient and by far the most successful approach towards reducing the real interaction time is to introduce task-dependent prior knowledge such as providing demonstrations or suboptimal controllers. However, these approaches require skilled human operators, some knowledge of the underlying proceeds; and may also hamper learning performance because this form of prior knowledge limits the possibility of finding optimal solutions. Besides, they also require several adaptation roll-outs performed on a real system, which again may result in severe damage.

It is a reasonable assumption that quicker learning prevents damage, but it should not be taken as a decided fact. It is reasonable because it minimizes maintenance efforts, but on the other hand, it relies on the experimenter to provide safety means during learning. Therefore, in search for the alternative techniques, a closer look at the examples of learning in nature might be helpful. Humans are not born self-sufficient, but attain maturation after a long development process as a result of interaction with a physical and social environment. During this time, people learn complex locomotion skills like walking or cycling, cognitive skills like language communication, logic abilities and deductive reasoning. Several learning paradigms are involved in mastering these skills, but undoubtedly progressive learning plays an important role. Since the first days of life, people continuously build up skills and knowledge on top of what is already acquired rather than try to pursue a certain skill without any ground basis prepared. Perhaps, failure to recognize this aspect of learning is the reason why the RL generalization results have been poor so far.

Current problems in RL applied to realistic robot tasks can be summarized as follows:

1. There is a lack of literature that aims at finding the minimum prior knowledge required for achieving learning goals while minimizing the damage.

2. It is problematic to learn policies that generalize well across similar models and robots.

3. It is unsafe to learn policies without task-dependent prior knowledge. Even then, the safety is not guaranteed, and additionally, task-dependent prior knowledge may hamper the final performance.

4. Ensuring safety during learning is usually considered to be an experimenter duty rather than accounted by the learning algorithm itself.

5. Methods that acquire new skills by gradually solving more complex tasks received little attention so far.

1.4 Research goal

The goal of this thesis is to discover learning strategies that are effective for real-world robotics. Specifically, the focus is on fragile systems, because such systems clearly establish the difference between the learning approaches suitable for simulators and the real unstructured world. In this context, the following research questions are addressed:

1. How do exploration strategies affect the damage of a real robot?

2. How can an approximate model of a robot help to mitigate the risks of damage?

3. Without having a model at hand, is it possible to reduce robot damage by sequencing learning tasks similar to the way humans learn gradually?

1.5 Approach

Previous research conducted in Delft Biorobotics Lab provides the basis for this work. Schuitema (2012) has developed the bipedal robot Leo specifically for learning with RL. The purpose of the development was to identify practical complications that arise from applying RL techniques to Leo. Among several complications, the author discovered that fragility of the robot's hardware is the crucial limit imposed on the real-world learning. This work can be viewed as the extension towards approaches that seek for safer learning on a fragile hardware.

Robot Leo and its simulated model are shown in Figure 1.1. The experimental setup is designed for conducting hours of the autonomous learning. For this purpose, the robot is connected to the boom, which prevents lateral falls, and is equipped with a single arm to facilitate stand-up recovery after falls. See Appendix A.1 for more details about the robot.

Dynamic bipedal walking is a challenging task due to nonlinear hybrid dynamics, unilateral foot-ground interaction, statically unstable single-support phase and susceptibility to modeling uncertainties. For these reasons, it is very difficult to create a feasible control policy or apply human-expert demonstrations. Therefore, RL is the appropriate tool for acquiring the walking policy. The intrinsic vulnerability of walkers to damage further motivates the usage of Leo in this research.

The following damaging factors can be identified for the real robot Leo:

1. physical damage due to falls and backlash reengagements,

2. sample-inefficient learning,

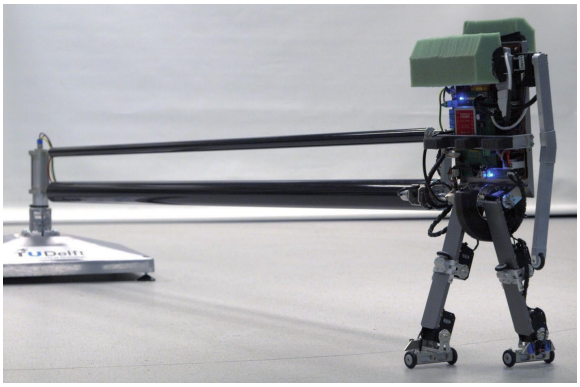


Figure 1.1: (*left*) 7 degree of freedom bipedal robot Leo developed by Schuitema (2012), and (*right*) its simulated model.

3. modeling uncertainties,

4. control delays.

Falls and backlash reengagements directly result in the physical damage of Leo, while the other factors indirectly influence it. The impacts of the control delays were already investigated by Schuitema (2012). Therefore, the first three items are in the focus of this work. Choosing the right prior knowledge is the key to mitigating the damaging factors. However, the amount of knowledge can vary. To answer the research questions, a varying amount of prior knowledge is injected, and hence its relation to the factors listed above is identified. The balance between prior knowledge and the system damage is the central thread of this work, which is schematically shown in Figure 1.2.

The work in this thesis is predominantly applied to the bipedal robot Leo. Nevertheless, the research results obtained in this work are generic and also apply to other robots.

1.6 Thesis outline

The remainder of the thesis is structured as follows:

Chapter 2 studies the influence of RL exploration strategies on the gearbox damage due to backlash re-engagements and its relation to falls and the final performance of the robot, thereby answering research question 1. This result applies to low-level control. Hence, it holds for generic model-free RL.

Chapter 3 begins to address research question 2. The final performance of the model-based and model-free approaches is compared in the presence of parametric and structural uncertainties. As a model-based approach, two versions of the NMPC framework are employed: the one that satisfies real-time constraints and

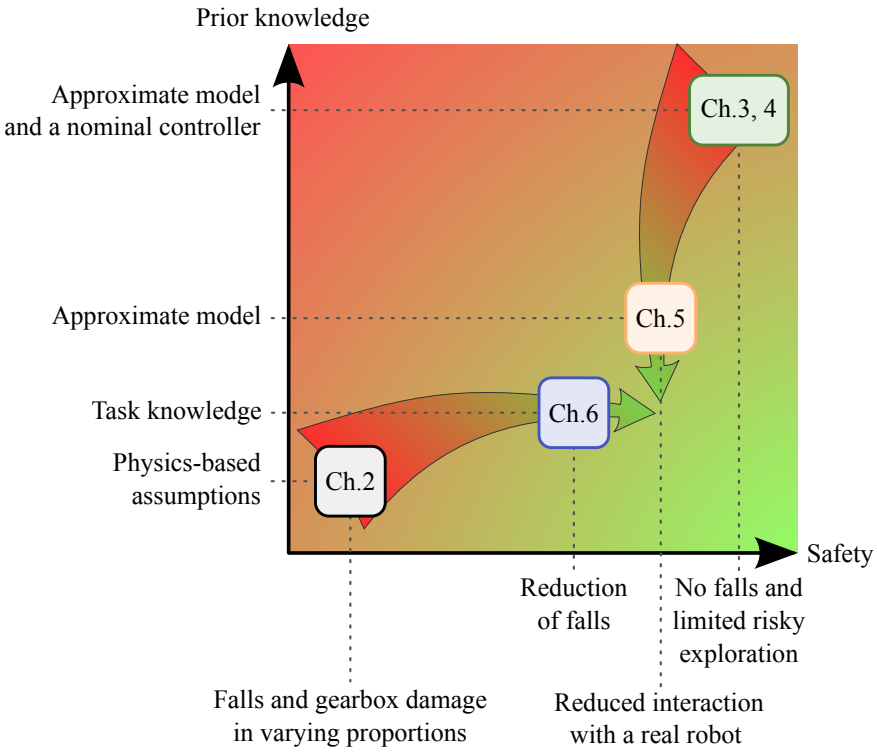


Figure 1.2: Structure of the thesis. The safety of the proposed approaches is evaluated relative to standard model-free learning.

the one that neglects them. Obtained results provide the basis for the combination of both approaches, which is proposed in the next chapter.

Chapter 4 continues to address research question 2. Two methods of compensating model-plant mismatch using a combination of NMPC and RL are proposed. The role of NMPC is to provide safety barriers to constrain RL exploratory actions near dangerous state space regions. Safe learning is not guaranteed, although the proposed approaches can be safe in practice. Moreover, the approaches can in principle compensate any type of uncertainty that preserves the Markov property; and without employing the time-consuming structure identification process or expert-designed models of friction, backlash, etc. Unfortunately, NMPC requires the dynamic system equations, optimization objectives and constraints to be at least twice continuously differentiable with respect to the optimization variables, which makes it difficult to apply this method to systems with hybrid dynamics. Nevertheless, this chapter highlights the significance of the model for safer learning.

Chapter 5 suggests another way of compensating model-plant mismatch by leveraging an approximate forward model of the robot. Interaction time with the robot

is minimized by learning the difference between the dynamics of the robot and its model. This approach can scale to systems with highly non-linear and contact-rich dynamics with continuous state and action spaces, but no safety barriers are provided. This chapter concludes the pursuit towards the answer to the research question 2.

Chapter 6 draws attention to the nature-inspired method of reducing the damage while learning, thereby answering research question 3. It is observed that often children learn to walk by progressing through intermediate stages of sitting, crawling and balancing. The specific arrangement of tasks is also practiced for rehabilitation of the after-stroke patients. The results of the simulations demonstrate that the approach of scheduled tasks can significantly reduce the number of robot falls compared to the case when the robots learns to walk immediately. Furthermore, it shows that the task switching moments can be automatically detected based on a set of performance indicators, which characterize the learning progress. Although in this chapter the focus is on gradually learning the control policy, in future works, curriculum learning can also facilitate learning of the system model. This model can later be used with damage-reducing approaches proposed in the previous chapters.

Chapter 7 summarizes conclusions and offers recommendations for future research.

2

Evaluation of physical damage associated with action selection strategies

This chapter introduces several exploration strategies and investigates their influence on the damage and the performance of Leo, thereby providing the answer to research question 1. The exploration strategies are studied in the context of model-free learning. Simulation results reveal a previously unknown trade-off between the two sources of damage: gearbox fatigue and the cumulative number of falls. Interestingly, one of the proposed methods with a time-correlated noise outperforms the well-known ϵ-greedy method. The main contribution of this chapter is that it provides guidance towards the choice of the exploration strategy for reinforcement learning (RL) applied to real mechanical systems.

Published in: Koryakovskiy, I., Vallery, H., Babuska, R., and Caarls, W. (2017). Evaluation of Physical Damage Associated with Action Selection Strategies in Reinforcement Learning. *IFAC-PapersOnLine*, 50(1):6928 – 6933.

2.1 Introduction

Until recently, robotic applications were mostly limited to controlled and well-predictable environments such as factories or space. However, currently scientists and engineers strive to bring robots to uncontrolled, partially observable and human-friendly environments. Despite the existence of advanced software and hardware, many challenges remain in the integration of robots into our society.

Machine learning techniques enable robots to deal with unknown environments without using explicit models or preprogrammed policies. In simulations, impressive results were obtained with deep learning in the actor-critic setting (Lillicrap et al., 2015). The authors use a deep neural network for learning both from low-dimensional state descriptions and high-dimensional renderings of the environment. In both cases, they have shown the ability of their approach to scale to complex tasks such as control of a seven-degree-of-freedom arm and bipedal locomotion, reaching a good control policy in at most 2.5 million steps.

However, the application of learning on real robots can be very costly. For example, our robot Leo, shown in Figure 2.1, can learn to walk by first observing a preprogrammed controller and then improving the observed policy using RL (Schuitema, 2012). Without the preprogrammed controller, Leo's gearboxes can only withstand five minutes of learning as a direct result of the aggressive nature of its learning strategy, involving large and rapidly changing motor torques (Meijdam et al., 2013). Therefore, in this chapter we investigate possibilities of reducing the damage while learning.

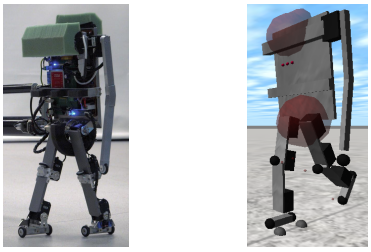


Figure 2.1: (*left*) Seven degree of freedom robot Leo and (*right*) its model.

Garcia and Fernandez (2015) give an overview of Safe RL. Perhaps the most prominent method of limiting damage is to define specific parameterized policies that are benign to the hardware at hand and then to learn the parameters only. This can, for example, be done from optimal control roll-outs (Levine and Koltun, 2013) or kinesthetic teach-in (Kober and Peters, 2011). In general, they can achieve good-quality policies within dozens of episodes but require a few human demonstrations for each task that needs to be learned.

An exploration method by Moldovan and Abbeel (2012) requires a model with known uncertainty in the dynamics. It restricts a set of policies to ergodic ones, which are policies that intrinsically encode the possibility of returning to an initial state from any other state.

On the hardware level, multiple contact dynamics were used in order to dissipate impacts with a minimum damaging effect on the robot (Ha and Liu, 2015). This planning strategy requires a model and explicit formulation of damage measures.

When aiming at higher robot autonomy and better generalization to unknown environments and new tasks, learning to control fragile systems in a model-free set-

ting is essential. Only a few methods have been proposed that explicitly consider safe exploration in this setting. For instance, trust region policy optimization (Schulman et al., 2015) generates near-monotonic improvements of a policy by choosing sufficiently small step sizes. Unfortunately, as mentioned by Lillicrap et al. (2015), it appears to be less data-efficient than unconstrained policies.

Another method, proposed by Gehring and Precup (2013), identifies areas of high randomness in the rewards or transitions and avoids those during exploration. It was shown that the approach can scale to high-dimensional problems and noisy state information.

Finally, superior results regarding mean time before failure (MTBF) were achieved by the Previous Action-Dependent Action (PADA) algorithm of Meijdam et al. (2013), where the author constrained a set of possible actions to remain within a fixed distance from a previous action. Our work can be seen as a continuation of this research. We select four commonly-used exploration methods (Greedy, ϵ-greedy, PADA, Ornstein-Uhlenbeck (OU)) for the comparison on the bipedal robot Leo. Earlier experiments (Meijdam et al., 2013) indicated that robot falls and foot impacts also contribute significantly to the MTBF. To distinguish these two sources, we compute the cumulative number of falls in addition to calculation of fatigue, MTBF and undiscounted return. The obtained results reveal a previously unknown trade-off between the number of falls and gearbox fatigue. Furthermore, by proposing four new exploration methods, we bridge the gap between the methods mentioned above and provide a better insight into the influence of exploration on the damage of Leo. As an outcome, we provide guidance towards a choice of exploration strategy for physical RL applications.

2.2 Reinforcement learning

2.2.1 The Markov decision process

Reinforcement learning can deal with unmodelled and noisy environments. The dimension of the state space is n_x with $\mathcal{X} \subset \mathbb{R}^{n_x}$ being the set of possible states. The dimension of the action space (the space of the control signals) is n_u with $\mathcal{U} \subset \mathbb{R}^{n_u}$ being the set of possible actions. Then a Markov decision process is defined as the quadruple $\langle \mathcal{X}, \mathcal{U}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{P} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ is a transition function that defines the probability of ending in state $\boldsymbol{x}_{k+1} \in \mathcal{X}$ after executing action $\boldsymbol{u}_k \in \mathcal{U}$ in state $\boldsymbol{x}_k \in \mathcal{X}$. The reward function $\mathcal{R} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ gives a real-valued reward $r_{k+1} = \mathcal{R}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1})$ for the particular transition between states. A Markov decision process satisfies the Markov property, which assumes that the current state $\boldsymbol{x}_k$ provides enough information to determine an optimal action $\boldsymbol{u}_k$.

A deterministic control policy $\pi : \mathcal{X} \to \mathcal{U}$ defines an action $\boldsymbol{u}_k$ taken in a state $\boldsymbol{x}_k$. The goal of learning a *continuing* task is to find an optimal control policy π^* that maximizes the discounted return,

$$G(\boldsymbol{x}_k) = \mathbb{E}\left\{ \sum_{i=0}^{\infty} \gamma^i r_{k+i+1} \right\},$$

where the immediate rewards are exponentially decayed by the discount rate $\gamma \in [0, 1)$ – rewards further in the future contribute less to the return.

The state-action value function $Q^\pi(\boldsymbol{x}_k, \boldsymbol{u}_k)$ denotes the expected return assuming that the system starts in the state $\boldsymbol{x}_k$ with the action $\boldsymbol{u}_k$ and then follows a prescribed control policy π. The optimal control policy maximizes the value for each state-action pair.

In this chapter, we solve a bipedal walking task using the well-known model-free temporal-difference RL algorithm SARSA (Sutton and Barto, 1998). The value function is represented by a linear function approximator using binary features defined by tile coding (Albus, 1975). A discrete action $\boldsymbol{u}_k$ is selected in state $\boldsymbol{x}_k$ according to one of the action-selection methods, and then the value function is updated according to

$$Q^\pi(\boldsymbol{x}_{k-1},\boldsymbol{u}_{k-1}) = Q^\pi(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \alpha(r_k + \gamma Q^\pi(\boldsymbol{x}_k, \boldsymbol{u}_k) - Q^\pi(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1})).$$

We implement standard accumulating eligibility traces for speeding up the convergence of SARSA.

In RL, exploration is achieved either by taking suboptimal actions with a certain probability or by initializing the value function optimistically, that is with values higher than the expected return. This causes visited states to become less attractive than states that have not been visited yet (Matignon et al., 2006). In this chapter, we only focus on methods of suboptimal action selection and do not consider optimistic initialization.

2.2.2 Action-selection methods

All studied action-selection methods and the relations between them are summarized in Figure 2.2. In the following, we explain details of each method.

Greedy

This method always takes the expected best possible action

$$\boldsymbol{u}_k = \arg\max_{\boldsymbol{u} \in \mathbb{U}} Q^\pi(\boldsymbol{x}_k, \boldsymbol{u}),$$

where $\mathbb{U} \subset \mathcal{U}$ is a discrete subset of possible actions.

ϵ - Greedy

This method takes a greedy action most of the time, but with a small probability $\epsilon > 0$ it samples a random action from a uniform distribution,

$$\boldsymbol{u}_k = \begin{cases} \arg\max_{\boldsymbol{u} \in \mathbb{U}} Q^\pi(\boldsymbol{x}_k, \boldsymbol{u}), & \text{with probability } 1 - \epsilon \\ \text{uniform}(\mathbb{U}), & \text{otherwise.} \end{cases}$$

PADA

Greedy and ϵ-greedy methods choose a future action independently from the previous action. However, it was shown by Meijdam et al. (2013) that selection of a new

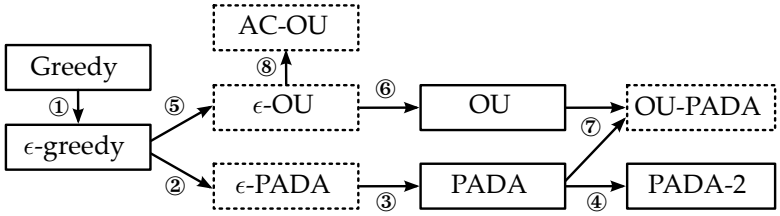


Figure 2.2: A relation between conventional (solid line) and proposed (dashed line) exploration methods. ① Take a random action with probability ϵ. ② Select a random action within a $\Delta \boldsymbol{u}$ interval. ③ Select greedy and random actions within the $\Delta \boldsymbol{u}$ interval. ④ Include $\boldsymbol{u}_{k-1} \pm 2\Delta \boldsymbol{u}$ actions to the action selection set. ⑤ Add a time-correlated noise to a greedy action taken with probability ϵ. ⑥ Add a time-correlated noise to a greedy action taken with probability 1. ⑦ Add a time-correlated noise to a greedy action constrained by the action selection set. ⑧ With probability ϵ, take an action correlated with a previous action.

action from a subset of actions defined around the previous action dramatically reduces the MTBF of RL. In case of Leo, the authors used a previous action and two neighboring actions:

$$\boldsymbol{u}_k = \begin{cases} \underset{\boldsymbol{u} \in \tilde{\mathbb{U}}(\boldsymbol{u}_{k-1})}{\arg\max}\ Q^{\pi}(\boldsymbol{x}_k, \boldsymbol{u}), & \text{with probability } 1-\epsilon \\ \text{uniform}(\tilde{\mathbb{U}}(\boldsymbol{u}_{k-1})), & \text{otherwise,} \end{cases}$$

where the set of neighboring actions is defined as

$$\tilde{\mathbb{U}}(\boldsymbol{u}_{k-1}) = \{\boldsymbol{u}_{k-1} - \Delta \boldsymbol{u}, \boldsymbol{u}_{k-1}, \boldsymbol{u}_{k-1} + \Delta \boldsymbol{u}\},$$

and $\Delta \boldsymbol{u}$ is a vector with all elements equal to the discretization step of controls Δu. In the case of the PADA-2 method, the set of neighboring actions is extended with actions located $\pm 2\Delta \boldsymbol{u}$ away from a previous action.

OU

Rather than taking an entirely random action such as with the ϵ-greedy method, the OU process (Lillicrap et al., 2015) adds time-correlated noise to a greedy action. The OU exploration term $\boldsymbol{m}_k$ is the integral over a Gaussian noise signal $\boldsymbol{n}_k \sim \mathcal{N}(0, 1)$, but pulled towards an asymptotic mean $\boldsymbol{m}_0$,

$$\begin{aligned} \boldsymbol{m}_k &= \boldsymbol{m}_{k-1} + \theta(\boldsymbol{m}_0 - \boldsymbol{m}_{k-1}) + \sigma \boldsymbol{n}_k \\ \boldsymbol{u}_k &= \underset{\boldsymbol{u} \in \mathbb{U}}{\arg\max}\ Q^{\pi}(\boldsymbol{x}_k, \boldsymbol{u}) + C_u \boldsymbol{m}_k. \end{aligned}$$

The three parameters, $\theta > 0$, $\sigma > 0$ and $\boldsymbol{m}_0$, influence the dynamics of the process, and C_u scales the noise to the values of admissible actions.

We establish a connection between the described methods by introducing four new action-selection methods.

ϵ - PADA

The method selects a greedy action at exploitation steps and a random action within $\pm\Delta\boldsymbol{u}$ bound at exploration steps, therefore bridging ϵ-greedy and PADA methods.

$$\boldsymbol{u}_k = \begin{cases} \arg\max\limits_{\boldsymbol{u}\in\mathbb{U}} Q^\pi(\boldsymbol{x}_k, \boldsymbol{u}), & \text{with probability } 1-\epsilon \\ \text{uniform}(\tilde{\mathbb{U}}(\boldsymbol{u}_{k-1})), & \text{otherwise.} \end{cases}$$

ϵ - OU

The method bridges the gap between ϵ-greedy and OU by only adding the OU process noise at exploration steps,

$$\begin{aligned} \boldsymbol{m}_k &= \boldsymbol{m}_{k-1} + \theta(\boldsymbol{m}_0 - \boldsymbol{m}_{k-1}) + \sigma\boldsymbol{n}_k \\ \boldsymbol{u}_k &= \begin{cases} \arg\max\limits_{\boldsymbol{u}\in\mathbb{U}} Q^\pi(\boldsymbol{x}_k, \boldsymbol{u}), & \text{with probability } 1-\epsilon \\ \arg\max\limits_{\boldsymbol{u}\in\mathbb{U}} Q^\pi(\boldsymbol{x}_k, \boldsymbol{u}) + C_u\boldsymbol{m}_k, & \text{otherwise.} \end{cases} \end{aligned}$$

OU - PADA

The method adds the OU process noise to the greedy action selected within $\pm\Delta\boldsymbol{u}$ bounds, therefore bridging OU and PADA methods.

$$\begin{aligned} \boldsymbol{m}_k &= \boldsymbol{m}_{k-1} + \theta(\boldsymbol{m}_0 - \boldsymbol{m}_{k-1}) + \sigma\boldsymbol{n}_k \\ \boldsymbol{u}_k &= \arg\max_{\boldsymbol{u}\in\tilde{\mathbb{U}}(\boldsymbol{u}_{k-1})} Q^\pi(\boldsymbol{x}_k, \boldsymbol{u}) + C_u\boldsymbol{m}_k \end{aligned}$$

AC - OU

Inspired by the OU process, we introduce an Action-Correlated Ornstein-Uhlenbeck (AC-OU) action-selection method. As in the ϵ-greedy method, we separate exploratory and greedy actions. An exploratory action is selected based on the previous action so that it does not stress the system as much as a random action would do. As in the OU process, we add a θ-multiplied term, which works as an action regularization,

$$\boldsymbol{u}_k = \begin{cases} \arg\max\limits_{\boldsymbol{u}\in\mathbb{U}} Q^\pi(\boldsymbol{x}_k, \boldsymbol{u}), & \text{with probability } 1-\epsilon \\ \boldsymbol{u}_{k-1} + \theta(\boldsymbol{m}_0 - \boldsymbol{u}_{k-1}) + \sigma\boldsymbol{n}_k, & \text{otherwise.} \end{cases}$$

Note that here σ and θ are applied on the action level and do not require scaling.

In addition to the above-described methods, we tried the Softmax action-selection method (Sutton and Barto, 1998), but there was no temperature for which it performed better than the Greedy method.[1] For this reason, we excluded Softmax from further investigation.

Table 2.1 gives the parameters of the methods presented. Those used for ϵ-greedy (Schuitema, 2012) and PADA (Meijdam et al., 2013) were taken from the

[1] The temperature was kept constant throughout the experiment.

corresponding articles, while for the other methods we tested a range of values and selected the ones that led to the highest undiscounted return. Additionally, a SARSA learning rate $\alpha = 0.2$, a discount rate $\gamma = 0.9962$, an eligibility trace decay rate of 0.8582 and a sampling period of $0.033\,\mathrm{s}$ of Leo's controller were taken from Schuitema (2012). The state-action value function $Q^\pi(\boldsymbol{x}_k, \boldsymbol{u}_k)$ was initialized with random values in the range of $[0; 0.01]$.

2.3 Simulations results

We evaluate properties of the described action-selection methods using the Leo dynamics simulator. Following Schuitema (2012), we exploit the symmetry of the bipedal walking problem to reduce the state and action space dimensions to ten and three, respectively. Actions from a voltage range of $[-10.7\,\mathrm{V}, 10.7\,\mathrm{V}]$ are discretized into seven linearly spaced values. We selected $C_u = 10.7\,\mathrm{V}$ to account for the whole range of admissible actions. Each episode lasted for $25\,\mathrm{s}$ or until Leo fell. Upon termination, Leo was initialized in the initial upright position with the right leg bent at the hip and knee joints. To make the experiment more realistic, all joint angles of the initial position were also perturbed by values uniformly drawn from the range of $\pm 5\,^\circ$ at the beginning of each learning episode. The reward was constructed with the goal of promoting a fast but energy-efficient forward walking. The simulator includes a realistic model of the Dynamixel RX-28 motor with the last gear of the gearbox made of anodized aluminum. Torque τ applied to the last gear is calculated from voltage U, the motor's torque constant K_τ, gearbox ratio K_G, the joint velocity $\dot{\phi}$ and the winding resistance R by

$$\tau = K_\tau K_\mathrm{G} \frac{U - K_\tau K_\mathrm{G} \dot{\phi}}{R}.$$

Following Meijdam et al. (2013), we use torque amplitude to estimate the number $N_k^{\mathcal{F}}$ of completely reversed cycles withstood before failure.[2] The completely reversed stress cycle is the cycle with zero mean and an equal magnitude of positive and negative stress. Assuming that each of the 45 teeth of the last gear is equally stressed, the fatigue $\mathcal{F}$ of the gear is calculated by

$$\mathcal{F} = \sum_{k=1}^{K} \frac{1}{45 N_k^{\mathcal{F}}},$$

where K is the number of gear re-engagements during learning. Note that our measure of fatigue accounts only for the cases when the torque sign changes, and fatigue is not influenced by falls of the robot. MTBF during learning is predicted as the time when $\mathcal{F} \geq 1$.

Figure 2.3 shows control trajectories of the left hip before and after learning. In the final policy, the ϵ-greedy and ϵ-OU methods showed high-frequency oscillations involving a change of voltage polarity. Greedy, OU, ϵ-PADA methods

[2] Note that every time k a gear re-engagement happens, the different torque amplitude results in different $N_k^{\mathcal{F}}$.

Table 2.1: Parameters of action-selection methods.

Method	Parameter values			
ϵ-greedy	$\epsilon = 0.050$			
PADA	$\epsilon = 0.050$	$\Delta u = 3.570$		
PADA-2	$\epsilon = 0.050$	$\Delta u = 3.570$		
OU	$\boldsymbol{m}_0 = 0.000$	$\theta = 0.001$	$\sigma = 0.020$	
ϵ-PADA	$\epsilon = 0.050$	$\Delta u = 3.570$		
ϵ-OU	$\epsilon = 0.050$	$\boldsymbol{m}_0 = 0.000$	$\theta = 0.001$	$\sigma = 0.020$
OU-PADA	$\boldsymbol{m}_0 = 0.000$	$\theta = 0.001$	$\sigma = 0.020$	$\Delta u = 3.570$
AC-OU	$\boldsymbol{m}_0 = 0.000$	$\theta = 0.100$	$\sigma = 2.000$	$\epsilon = 0.050$

showed moderate voltage oscillations, and PADA, PADA-2, OU-PADA and AC-OU showed the least ones.

Table 2.2 summarizes the performance of the methods in terms of gearbox fatigue, MTBF at the beginning of learning and final MTBF after learning (i.e., when only greedy actions are applied), the cumulative number of falls of Leo and the undiscounted return obtained. A careful comparison of fatigue and MTBF during learning results of ϵ-greedy, ϵ-PADA and ϵ-OU with the help of Figure 2.4a reveals the difference between these benchmarks. The rate of fatigue accumulation was nonlinear and slowed down after approximately 25 min since the beginning of learning. This value can be regarded as an average number of gear replacements during learning. Therefore, fatigue gives a more accurate estimation of loss during learning comparing to MTBF, which only accounts for a fail-free learning time at the beginning of a simulation. To avoid clutter in plots, we decided to present the curves of the five most characteristic methods, Greedy, PADA, OU, OU-PADA and AC-OU.

PADA and OU-PADA methods resulted in a remarkably low fatigue, leaving behind all other methods. Extending the action selection set with just two more actions (PADA-2) already increased fatigue caused by the change of a torque sign, and most noticeably reduced final MTBF by more than four times. It also significantly decreased the cumulative number of falls.

All action-selection methods succeeded in learning a walking gait and reaching reasonable rewards, see Figure 2.4b. PADA and OU-PADA rising slopes were slightly less steep comparing to other methods, but OU-PADA reached a much higher level of end performance comparing to PADA. Table 2.2 shows that OU significantly outperformed the other methods.

The cumulative number of falls encountered during learning is shown in Figure 2.4c. The smallest number of falls was achieved by the Greedy method, which was closely followed by OU-PADA and then AC-OU. PADA and OU methods re-

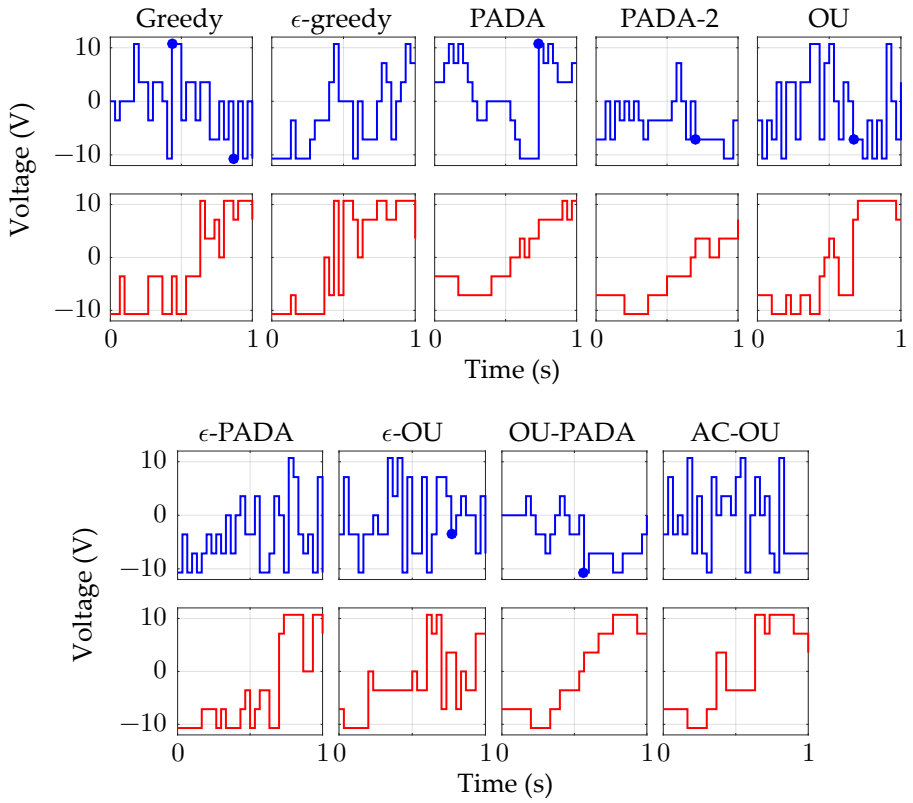


Figure 2.3: Initial (*blue*) and final (*red*) control signals. Solid dots (•) denote the beginnings of new episodes.

sulted in approximately 2.5 and 8 times larger numbers of falls compared to the Greedy method, respectively.

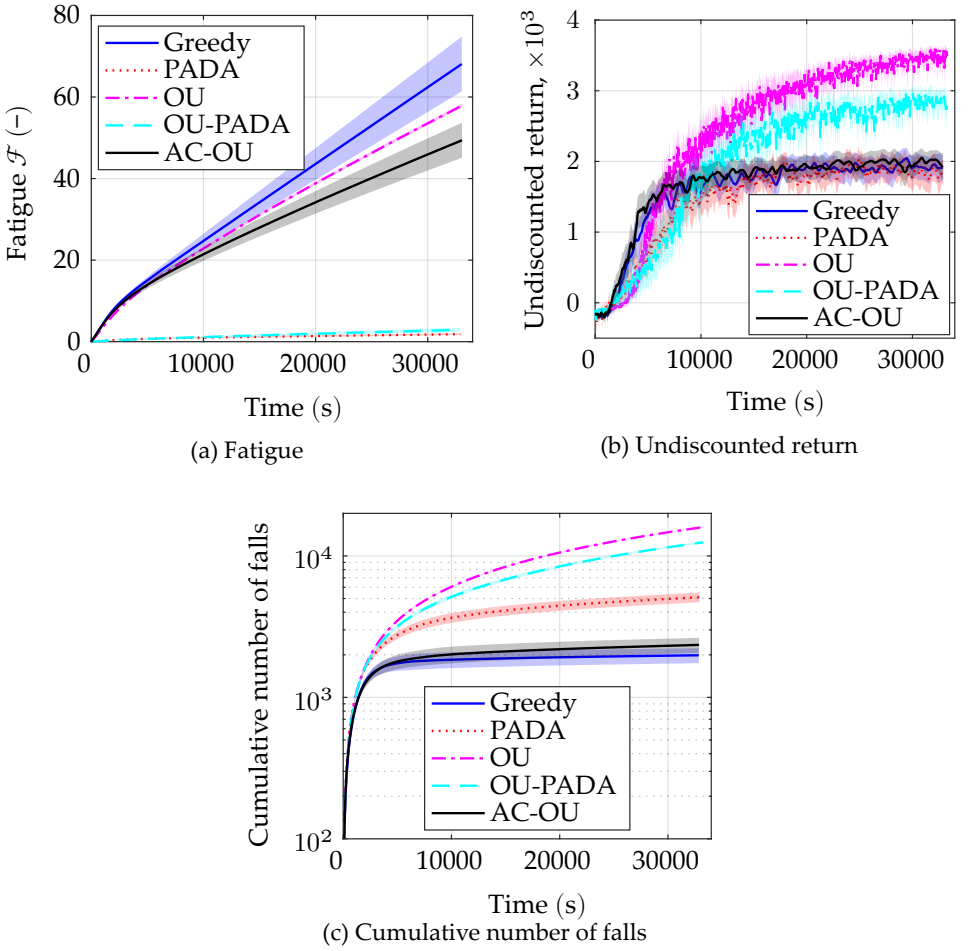
In this chapter, we do not experiment with the real robot, because that would incur a continuing damage. Meijdam et al. (2013) demonstrated the increase of MTBF by limiting the changes in a control signal applied to the real Dynamixel RX-28 motor. This fact correlates well with our results.

2.4 Discussion

PADA significantly outperformed all exploration methods in terms of MTBF and fatigue. However, during learning under this action-selection method, the simulated Leo underwent a significant number of falls and achieved the worst performance. While the decrease in performance was already described, the trade-off between number of falls and MTBF was previously unknown. The explanation of this could be the following: PADA always selects an action that is the same as or close to the previous one. This reduces fatigue because gear re-engagements happen much

Table 2.2: Mean and 95 % confidence interval of fatigue, MTBF, cumulative number of falls and undiscounted return obtained by each studied method averaged over 50 independent runs.

Method	Learning fatigue $\mathcal{F}$	MTBF at start (in min)	Final MTBF (in min)	Cumulative # of falls	Return
Greedy	68.68 ± 34.00	4.38 ± 0.72	12.67 ± 8.89	$\mathbf{1984 \pm 241}$	1908 ± 193
ϵ-greedy	79.13 ± 21.18	4.41 ± 0.71	11.68 ± 4.66	3529 ± 351	2109 ± 122
PADA	$\mathbf{1.86 \pm 1.11}$	$\mathbf{338.73 \pm 250.86}$	$\mathbf{699.20 \pm 356.65}$	5099 ± 399	1824 ± 180
PADA-2	4.92 ± 1.00	73.33 ± 36.32	166.09 ± 57.21	2962 ± 206	1930 ± 150
OU	58.27 ± 2.27	5.19 ± 0.48	54.36 ± 27.30	15919 ± 144	$\mathbf{3501 \pm 110}$
ϵ-PADA	55.47 ± 22.08	4.40 ± 0.60	18.45 ± 10.85	2478 ± 294	2193 ± 129
ϵ-OU	63.24 ± 30.75	4.40 ± 0.68	15.77 ± 12.29	$\mathbf{2098 \pm 246}$	2012 ± 154
OU-PADA	$\mathbf{2.94 \pm 2.73}$	$\mathbf{377.67 \pm 295.50}$	$\mathbf{1292.41 \pm 855.44}$	12435 ± 227	2811 ± 174
AC-OU	49.73 ± 21.66	4.38 ± 0.70	21.07 ± 10.87	2348 ± 288	1951 ± 176



(a) Fatigue

(b) Undiscounted return

(c) Cumulative number of falls

Figure 2.4: During learning three benchmarks are calculated: (a) fatigue accumulated due to gear re-engagements, (b) undiscounted return, and (c) cumulative number of falls. Means with upper and lower 95% confidence limits are shown for 50 samples.

more rarely. However, the prevention of falls may require an immediate reaction, which may involve a rapid change of the control signal sign. This hypothesis closely correlates with the fact that PADA resulted in the smallest consecutive change of control signal among all studied methods. Reducing the constraints on actions as in PADA-2 also supports this hypothesis, because the cumulative number of falls was reduced at the expense of larger fatigue.

However, the absence of any constraint also led to more damage, which can be observed in the results of OU and OU-PADA. The OU explores very well in physical environments, but in the experiment it was the most demanding with respect to hardware endurance. Constraining actions as in OU-PADA not only reduced

the fatigue, but also reduced the number of falls, at the cost of decreased walking performance.

It is important to note the difference between uniform noise (ϵ-greedy) and time-correlated noise (ϵ-OU) during exploration. The results in Table 2.2 demonstrate that time-correlated noise reduced the number of falls by more than 40%, leaving all other benchmark values within the confidence intervals of ϵ-greedy. ϵ-PADA and AC-OU showed similar results with a slight shift towards a lower fatigue, but a higher number of falls.

Both Greedy and AC-OU showed intermediate performance. Greedy underwent the lowest number of falls during learning, but AC-OU outperformed Greedy in terms of fatigue and MTBF. Interestingly, AC-OU obtained the lowest MTBF among methods that did not constrain actions during the exploitation step.

For a clear overview of the results, we summarize them in Table 2.3. First, we note that none of the methods surpassed others in both fatigue and number of falls of the robot. This suggests that to minimize damage from both sources, a faster learning algorithm is required. In the context of exploration strategies, faster learning may be achieved by a problem-driven high-level guided exploration. Second, exploration based on time-correlated noise outperformed the ϵ-greedy method, therefore for actual experiments with a robot, the ϵ-greedy strategy is not advised. Finally, no definite conclusion can be drawn about which exploration method is better for a generic physical system. Nevertheless, some insight can be provided. If the falls are highly damaging, then either Greedy, ϵ-PADA, ϵ-OU or AC-OU should be used. On the other hand, if the robot can withstand falls, but the gear re-engagements are damaging, then PADA, PADA-2 or OU-PADA methods are advisable. This is the case for the robot Leo, whose gears are made of aluminum and can easily be damaged by random exploration. Gears made of hardened steel instead of aluminum are more robust against gear re-engagements. Thus, when the amount of damage induced by crashes is little, it would be practical to use OU or OU-PADA, as they achieve high performance.

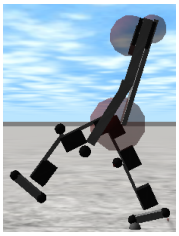
Further reduction of falling or fatigue can be achieved by a time-dependent decay schedule applied to ϵ or σ. We expect that such strategies will only affect the benchmark results relatively, and our conclusions will still hold.

It is worth mentioning that in addition to the above factors, the damage depends on the configuration of the environment, the protection of the robot, the severity of contact impacts, and other factors. For example, visual observation of Leo's gait after learning with OU (Figure 2.5) exhibited high lifts of a swing leg, therefore large steps and presumably high damage due to higher swing leg velocities right before heel strikes, compared to ϵ-greedy. The figures of fatigue in Table 2.2 do not account for this source of damage. We expect that our future experiments with real Leo will unveil the contribution of the described factors to the total damage of the robot.

Finally, we note that there might not be a single supreme exploration strategy when controlling physical systems, but exploration can rather be system- and task-driven. Similar findings were made in neuroscience, where dynamic regulation of exploration strategies has been observed in human and animals. Wu et al. (2014)

Table 2.3: A simplified overview of benchmark performances of action-selection methods.

Method	Minimizes gear re-engagements	Minimizes cumulative number of falls	Maximizes return
Greedy	$-$	$+$	$-$
ϵ-greedy	$-$	$+/-$	$+/-$
PADA	$+$	$-$	$-$
PADA-2	$+$	$+/-$	$-$
OU	$+/-$	$-$	$+$
ϵ-PADA	$+/-$	$+$	$+/-$
ϵ-OU	$+/-$	$+$	$+/-$
OU-PADA	$+$	$-$	$+$
AC-OU	$+/-$	$+$	$-$



$0.66 \pm 0.16\,\mathrm{m\,s^{-1}}$

$0.68 \pm 0.24\,\mathrm{m\,s^{-1}}$

$0.87 \pm 0.22\,\mathrm{m\,s^{-1}}$

Figure 2.5: Maximum raise of the swing leg after learning with (*left*) ϵ-greedy, (*middle*) ϵ-OU and with (*right*) OU. Swing leg velocities and standard deviations right before heel strikes obtained after five independent learning runs are shown below each picture.

provide experimental support for the hypothesis that motor variability is centrally driven and is regulated according to the nature of the task.

2.5 Conclusion

In this chapter, we studied properties of several conventional and newly proposed action-selection methods in terms of their performance and the damage they cause to motor gears on the one hand and to the overall system on the other hand. We showed that none of the methods was capable of minimizing both sources of damage. Based on the quantitative comparison, we characterized conditions required for the selection of a certain method for learning in a physical system. Results indicate that uniform exploration, commonly achieved by the well-known ϵ-greedy exploration method, was not a good choice for learning on a physical robot. Our sim-

ulation results demonstrated that exploration based on the time-correlated noise (ϵ-OU) achieved similar performance and fatigue levels, but additionally it reduced the number of falls of the robot. In contrast, limiting the action set (OU-PADA) resulted in better performance and much less fatigue, but a larger number of falls.

3

Benchmarking model-free and model-based optimal control

Ivan Koryakovskiy, Manuel Kudruss[1]

This chapter begins to address research question 2, where the difference of a model-based optimal control (OC) and model-free reinforcement learning (RL) is studied in the context of parametric and structural uncertainties. Parametric uncertainties describe the uncertainties observed in the values of the dynamic model parameters. Their values are not known a priori, but can be inferred from interactions with the real system, i.e., the parameters are observable. Structural uncertainties describe uncertainties that originate from the lack of knowledge about the true physics of the underlying dynamic system. Results demonstrate that nonlinear model predictive control (NMPC) has advantages over RL if uncertainties can be eliminated through identification of system parameters. Otherwise, there exists a break-even point after which model-free RL performs better than NMPC with an inaccurate model. These findings suggest that benefits can be obtained by combining these methods for real systems being subject to such uncertainties. Two possible combinations of the approaches are proposed and evaluated in the next chapter.

Published in: Koryakovskiy, I., Kudruss, M., Babuska, R., Caarls, W., Kirches, C., Mombaur, K., Schloder, J. P., and Vallery, H. (2017). Benchmarking Model-Free and Model-Based Optimal Control. *Robotics and Autonomous Systems*, 92:81 – 90.

[1] M.Kudruss is with Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Im Neuenheimer Feld 205, 69120 Heidelberg, Germany.

3.1 Introduction

In robotics, one cannot expect to work with ideal models of the systems under control, or of their environments. Rather, we have to face unforeseen situations and unknown conditions, and aim for reactions that are feasible and, ideally, optimal with respect to given task performance criteria. A typical task is bipedal locomotion, where a robot needs to maintain stability and pace on an uneven floor with uncertain roughness and slope (Schuitema, 2012).

Two common approaches to control dynamic systems are NMPC and RL. Both approaches can cope with uncertainties in the form of model-plant mismatch. Reinforcement learning has been proven suitable as a real-time closed-loop control concept in robotics (Kober et al., 2013), and NMPC in industry (Qin and Badgwell, 2003). However, the use of NMPC in robotic applications, especially humanoid robotics and bipedal walking, is still an open research field (Herdt et al., 2010; Erez et al., 2013; Kuindersma et al., 2015).

In this chapter, we use a swing-up and balancing problem for a *cart-pendulum* system (Barto et al., 1983; Kimura and Kobayashi, 1999) to quantitatively assess both control approaches. Our choice of this benchmark problem is motivated by the fact that main features of passive dynamic walking can be modeled by an inverted pendulum (Wisse, 2004). The same equivalence holds for the upper body of a more detailed model of a bipedal walker. The study presented in this chapter highlights the differences in performance of NMPC and RL under structural and parametric uncertainties for this benchmark problem.

Nonlinear model predictive control Nonlinear model predictive control is a closed-loop control strategy in which the control action at the current sampling instant is computed by solving an open-loop OC problem over a finite prediction horizon. NMPC, as a model-based optimal control method, relies on a given mathematical model of the real-world system to be controlled. In this context, advanced direct methods of optimal control, see the survey by Biegler (2013), are the methods of choice for computing NMPC feedback control actions in real-time.

For NMPC, full state and parameter information of the model is required to compute the control action. Whenever the full state is not measurable or model parameters are not exactly known, methods of on-line state and parameter estimation have to be applied. For this purpose, extended Kalman filters (Jazwinski, 2007) or moving horizon estimation (MHE) techniques (Muske et al., 1993; Kühl et al., 2011) have been successfully applied. In this chapter, MHE is used to estimate uncertain parameters in the model.

Reinforcement learning Reinforcement learning is an active research area in the field of artificial intelligence and machine learning, with applications in control. The most important feature of RL is its ability to learn without prior knowledge about the system. The goal of the learning task is supplied externally in the form of a reward function. RL is a trial-and-error method, which generally takes many iterations before it finds an optimal solution. To reduce the number of interactions with

the system, model-learning methods such as Dyna (Caarls and Schuitema, 2016), learning from demonstration (Abbeel et al., 2010; Smart and Kaelbling, 2000), or optimized control policy parameterizations (Kober and Peters, 2011) can be applied. Because RL does not require an explicitly given model, it can naturally adapt to uncertainties of the real system. In this sense, RL can be viewed as a model-free adaptive optimal control approach (Sutton et al., 1992).

Related work of comparison and combination of RL and NMPC In this chapter, we provide a comprehensive comparison of the performance of RL and NMPC both for an ideal system as well as in the presence of parametric and structural uncertainties. To our knowledge, this is the first time this is done in a systematic and quantitative way for uncertain systems. Based on the presented comparison results, we identify the strong and weak points of both algorithms, which suggests a presence of mutual benefits for their combination.

A related comparative study for ideal systems can be found in (Ernst et al., 2009). The authors highlight similarities of NMPC and RL, including optimality of methods, truncation of a time horizon, and continuous vs. discrete actions. They show that, for an electrical power oscillations damping problem, NMPC slightly outperforms RL, yet both policies were essentially similar. Furthermore, the authors propose the idea of combining RL and NMPC. In an on-line (local) mode, NMPC could start optimization from the initial guess of the optimal trajectory precomputed by RL in an off-line (globally optimal) mode. Our conclusions go beyond the validation of similarity of solutions or computational benefits for ideal models. We provide numerical evidence that under uncertainties, situations may arise in which one or the other method can be favorable for performance.

A distinction of both methods was observed and successfully realized in a hybrid approach, a variant of the Guided Policy Search algorithm (Levine and Koltun, 2013). The approach was able to learn obstacle avoidance policies for a quadrotor (Zhang et al., 2016). It adopted a collection of model predictive control roll-outs obtained under full state observation and trained a deep control policy that required only the on-board sensors of the vehicle.

A study of the influence of the RL reward function on steady-state error was performed in (Engel and Babuska, 2014). It was shown that direct translation of a quadratic objective function used in standard linear-quadratic regulator resulted in a consistent, though not acceptable steady-state error. In contrast, using the absolute-value reward function yielded a response with negligible error.

Computational study The study conducted in this chapter is set up as follows, see Figure 3.1. In the first step (I), we establish optimal control ("OC") solutions for the ideal benchmark problem. Then we consider the NMPC formulation and derive the corresponding RL formulation from it. We highlight the changes introduced in both formulations and discuss their effects.

Subsequently, we address the strengths and weaknesses of NMPC and RL in terms of their ability to adapt to *structural* and *parametric* uncertainties. In the second step (II), we investigate NMPC and RL methods that are explicitly unable to

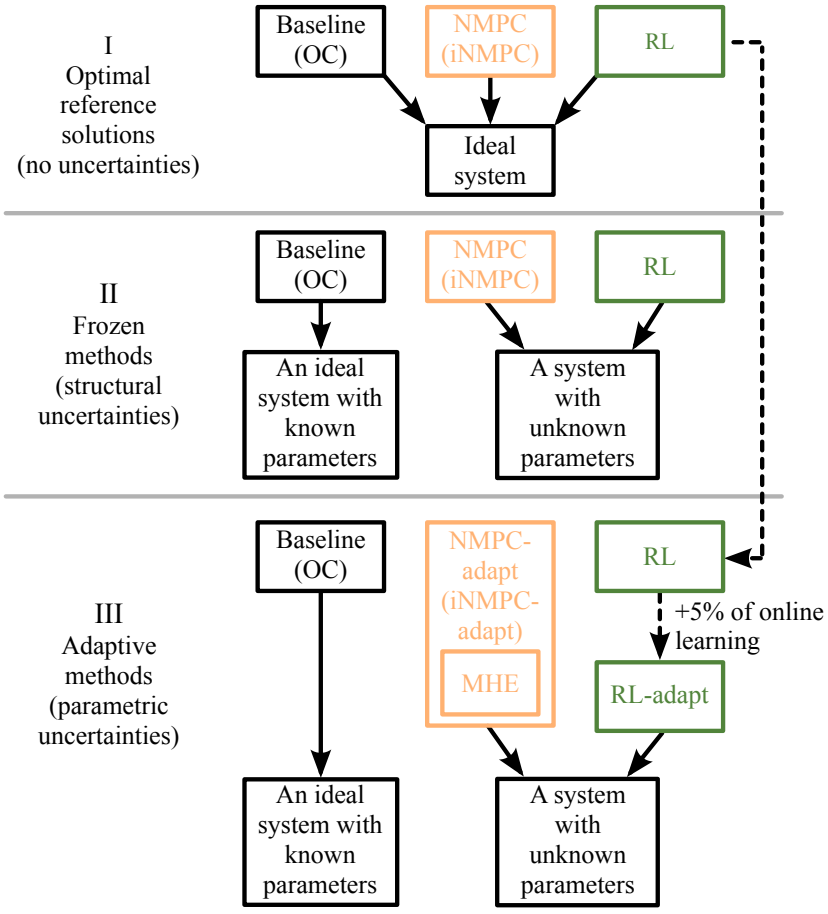


Figure 3.1: Overview of the computational study. Step I corresponds to a verification of state and control trajectories when problem formulations are equivalent. In steps II and III uncertainties of a varied magnitude are introduced. In the former case "NMPC", "iNMPC" and "RL" are not equipped with an adaptation mechanism while in the latter case they are. In "NMPC-adapt" and "iNMPC-adapt" adaptation is accomplished by means of MHE, and for "RL-adapt" we allow 5 % of additional interaction with the real system.

adapt to uncertainties. We introduce the term *frozen* to refer to this inability. In the third step (III), the effect of uncertainties and the ability to adapt to them is analyzed for NMPC methods that have explicitly been equipped with the knowledge about the uncertainties and for RL that is allowed to interact with the real system for an additional 5 % of the learning time. We introduce the term *adaptive* to distinguish these from the *frozen* methods.

In the remainder of the chapter, we use a single RL method denoted as "RL", and two NMPC versions denoted as "iNMPC" and "NMPC". "iNMPC" is an ideal NMPC controller that neglects computational time and returns an optimal control signal immediately. In turn, "NMPC" represents an actual NMPC implementation tuned for real-time feasible control.

3.2 Model-based and model-free optimal control methods

3.2.1 Optimal control

The optimal solutions used as baseline for the comparison are the solutions of the open-loop optimal control problem given by

$$\begin{aligned}
\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \quad & \int_0^T L(\boldsymbol{x}(t),\boldsymbol{u}(t))\,\mathrm{d}t + M(\boldsymbol{x}(T)) && \text{(3.1a)}\\
\text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}), \quad t \in \mathcal{T}, && \text{(3.1b)}\\
& \boldsymbol{x}(0) = \boldsymbol{x}_0, && \text{(3.1c)}\\
& 0 \leq \boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t)), \quad t \in \mathcal{T}, && \text{(3.1d)}
\end{aligned}$$

where we strive to find a control trajectory $\boldsymbol{u} : [0,T] \to \mathbb{R}^{n_u}$ such that an objective function composed of a Lagrange term $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and a Mayer term $M : \mathbb{R}^{n_x} \to \mathbb{R}$ is minimized on a finite time interval $\mathcal{T} = [0,T] \subset \mathbb{R}$. The state trajectory $\boldsymbol{x} : \mathcal{T} \to \mathbb{R}^{n_x}$ is characterized by the dynamic system defined by a set of ordinary differential equations with right hand side $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$, which depends in particular on the model parameters $\boldsymbol{p} \in \mathbb{R}^{n_p}$ of the system. In addition, mixed state-control path constraints $\boldsymbol{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$ are imposed on the system.

We follow a direct and all-at-once approach to optimal control and discretize the control trajectory $\boldsymbol{u}(\cdot)$ on a time grid $0 = t_0 < t_1 < \cdots < t_K < t_{K+1} = T$ by means of basis functions parametrized by piecewise constant control parameters. A direct multiple shooting approach to optimal control (Bock and Plitt, 1984) then further parametrizes the state trajectory $\boldsymbol{x}(\cdot)$ by introducing $K+1$ variables and by solving initial value problems separately on the time grid. From this discretization and parametrization, a large but structured nonlinear programming problem is obtained that can be solved efficiently with tailored structure-exploiting sequential quadratic programming methods. The evaluation of sensitivities requires L, M, f, $\boldsymbol{g}$ in the OC problem (3.1) to be at least twice continuously differentiable with respect to the optimization variables $\boldsymbol{x}$ and $\boldsymbol{u}$.

3.2.2 Nonlinear model predictive control

NMPC is a closed-loop control strategy in which the control action is computed from the current system state by solving an open-loop optimal control problem on a finite prediction horizon $\mathcal{T}_{NMPC} \subseteq \mathcal{T}$ on-line, therefore NMPC is also denoted as receding horizon control. In contrast to objective function (3.1a) of the OC problem, the tracking NMPC minimizes a nonlinear least-squares function composed of

$$L(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \|\boldsymbol{x}(t) - \bar{\boldsymbol{x}}(t)\|^2_{\mathbf{W}} + \|\boldsymbol{u}(t)\|^2_{\mathbf{V}}, \tag{3.2}$$

which minimizes the distance to given reference trajectories $\bar{\boldsymbol{x}} : \mathcal{T}_{NMPC} \to \mathbb{R}^{n_x}$ by means of a weighted ℓ_2-norm $\|\boldsymbol{x}\|_{\mathbf{W}} = \sqrt{\boldsymbol{x}^\top \mathbf{W} \boldsymbol{x}}$ with a positive definite weighting matrix $\mathbf{W}$. Similarly, controls are regularized with a positive definite weighting matrix $\mathbf{V}$.

At the current time instant $t = 0$, the full state $\hat{\boldsymbol{x}}_0 \in \mathbb{R}^{n_x}$ and parameter vector $\hat{\boldsymbol{p}} \in \mathbb{R}^{n_p}$ of the system are embedded into the open-loop optimal control problem by additional constraints replacing (3.1c) of the OC problem

$$\begin{aligned} 0 &= \hat{\boldsymbol{x}}_0 - \boldsymbol{x}(0), \\ 0 &= \hat{\boldsymbol{p}} \quad - \boldsymbol{p}. \end{aligned} \tag{3.3a}$$

In contrast to the OC problem, the parameters $\boldsymbol{p} \in \mathbb{R}^{n_p}$ are part of the optimization variables for the NMPC problem and we denote an estimate of the respective quantity by $\hat{}$.

State-of-the-art NMPC methods based on nonlinear programming rely on the real-time iteration scheme due to (Diehl, 2001; Diehl et al., 2005) to compute feedback in real-time. This is achieved by careful initialization of the sequential quadratic programming method by separating each iteration into three phases, i.e., 1) preparation (setup of quadratic program), 2) feedback (solution of quadratic program) as soon as the current system information $\hat{\boldsymbol{x}}_0$, $\hat{\boldsymbol{p}}$ is estimated and 3) transition (perform step and shifting). In this way, computationally expensive parts can be separated from time-critical ones and the computational delay of the feedback is reduced to the time required to solve a single quadratic program. Advanced methods further these ideas by dividing the real-time iteration into sub steps that can provide feedback even faster by evaluating only parts of the required Jacobian information, c.f. Bock et al. (2007); Frasch et al. (2012), or applying specialized iterative linear algebra, as in (Johnson et al., 2015).

Moving horizon state and parameter estimation For the state and parameter estimates, i.e., $\hat{\boldsymbol{x}}_0$, $\hat{\boldsymbol{p}}$ in (3.3), we apply MHE, c.f. (Kühl et al., 2011). After the estimates for $\hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{p}}$ have been obtained, they are embedded in the NMPC problem via the constraints (3.3) and are considered in the computation of the next feedback control action. In this way, model-plant mismatch and uncertainties can be tackled by including parameters in the NMPC formulation and computing an estimate in every pass of the control loop. Approaches similar to those used to achieve real-time feedback control for NMPC can be used to solve the MHE problem on-line.

In contrast to NMPC, the MHE horizon $\mathcal{T}_{MHE}$ refers to the passed time and the objective function minimizes the squared error between the model response h and measurements $\boldsymbol{y} = h(\boldsymbol{x}(\boldsymbol{p}^*), \boldsymbol{p}^*) + \boldsymbol{n}$ from the real system defined by the true but unknown parameters $\boldsymbol{p}^*$, subject to an additive measurement error $\boldsymbol{n} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ with zero mean and the diagonal covariance matrix $\boldsymbol{\Sigma}$ given by

$$\sum_{k=-K}^{0} \|h_k(\boldsymbol{x}(t_k), \boldsymbol{p}) - \boldsymbol{y}_k)\|^2_{\boldsymbol{\Sigma}_k^{-1}}.$$

Here, the parameters $\boldsymbol{p}$ are part of the optimization variables and the control actions $\boldsymbol{u}$ are fixed to the ones applied to the system in the past.

3.2.3 Reinforcement learning

A common approach in RL is to model the task as a Markov decision process. The process is defined as a quadruple $\langle \mathcal{X}, \mathcal{U}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{X} \subset \mathbb{R}^{n_x}$ is a set of possible states, $\mathcal{U} \subset \mathbb{R}^{n_u}$ is a set of possible control actions, $\mathcal{P} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ is a transition function that defines the probability of ending up in state $\boldsymbol{x}_{k+1} \in \mathcal{X}$ after executing action $\boldsymbol{u}_k \in \mathcal{U}$ in state $\boldsymbol{x}_k \in \mathcal{X}$. The reward function $\mathcal{R} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ gives a real-valued reward $r_{k+1} = \mathcal{R}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1})$ for the particular transition between states. A Markov decision process satisfies the Markov property, which assumes that the next state $\boldsymbol{x}_{k+1}$ depends only on the current state $\boldsymbol{x}_k$ and action $\boldsymbol{u}_k$, but not on previous states or actions.

A deterministic control policy $\pi : \mathcal{X} \to \mathcal{U}$ defines an action $\boldsymbol{u}_k$ taken in a state $\boldsymbol{x}_k$. The goal of the learning process is to find an optimal control policy π^* that maximizes the discounted return

$$G_k = \mathbb{E}\left\{ \sum_{i=0}^{K} \gamma^i r_{k+i+1} \right\},$$

where immediate rewards r are exponentially decayed by the discount rate $\gamma \in [0, 1]$ the further they lie in the future. The task that we consider in this chapter is a *continuing* task, for which the final time step is infinite, $K \to \infty$. This requires the use of $\gamma < 1$ to avoid infinite returns.

The value function $V^\pi(\boldsymbol{x})$ denotes the expected return assuming that the system starts in the state $\boldsymbol{x}$ and then follows a prescribed control policy π. The optimal control policy π^* maximizes the value for each state. Therefore, an optimization of the control policy is tightly coupled with the maximization of the value function in RL.

For real-world systems, continuous control is preferred. This requires a parametrization of the policy $\pi(\boldsymbol{x})$, e.g. using a set of basis functions and their associated weights. The weights are usually optimized by gradient-descent methods (Williams, 1992; Bhatnagar et al., 2009; Grondman et al., 2012), or by global gradient-free methods (Hansen and Ostermeier, 2001; Botev et al., 2013). We use a standard gradient-descent method because the latter methods require a substantial number of interactions with the robot which can be especially damaging in

case of walking tasks. Since the estimation of policy gradients often results in a high variance, the policy update is often coupled with an explicit estimation of a parametrized value function $V^{\pi}(\boldsymbol{x})$. This combination is known as the actor-critic method, where the policy is referred to as the actor, and the value function is referred to as the critic.

The method we use is the standard model-free temporal-difference-based method described by Grondman et al. (2012). Since RL is a trial-and-error learning method, the quality of the policy as well as the learning speed depend on the exploration method. Exploration is commonly achieved either by perturbation of the so far optimal action, or by optimistic initialization, or by both. Optimistic initialization is a method of initializing the value function with a value equal to or greater than the maximum possible value of a state. This causes the visited states to become less attractive than the states that have not been visited yet (Matignon et al., 2006). Optimistic initialization can speed up the learning in the absence of negative rewards. For the parametrization of the policy and value-function we use a linear in parameters tile coding approximator (Sutton and Barto, 1998).

3.3 Benchmark system

The two-dimensional benchmark example studied in this chapter is a pendulum attached to a cart (Barto et al., 1983; Kimura and Kobayashi, 1999), which is shown in Figure 3.2. The system consists of a cart with mass m_M and a pendulum that is attached to the cart's center of mass c_M.

The pendulum is a point mass m_m attached at the end of a massless rod of length l. The system has two degrees of freedom, namely the linear motion of the cart along the x-axis, described here by the coordinate $s \in \mathbb{R}$, and the rotary motion of the pendulum with respect to the cart, described by the angle $\phi \in \mathbb{R}$. The only actuation is realized by a horizontal force $F_s \in \mathbb{R}$ acting on the cart body.

The system's state is given as $\boldsymbol{x} = \left[s, \phi, \dot{s}, \dot{\phi}\right]^{\top} \in \mathbb{R}^4$. Here, $s, \dot{s}$ denote the cart position and velocity, and $\phi, \dot{\phi}$ denote the pendulum's angle and angular velocity, respectively. The control $\boldsymbol{u} = [F_s, 0]^{\top} \in \mathbb{R}^2$ is the force acting on the cart body.

In an ideal scenario, both the cart and the pendulum can move without friction along their respective degrees of freedom. For our second and third experiment, we employ uncertainty in the form of viscous friction at the rotary joint, i.e., in the pendulum joint bearing. This produces an internal torque $\tau_{\phi} = -\kappa\mu\dot{\phi}$ applied to the pendulum, where κ is a coefficient that depends on the configuration of the rotary joint, and μ is a viscous friction coefficient. Depending on whether or not this friction is included in the model, uncertainty in friction can be considered as a parametric or as a structural uncertainty.

More details of the benchmark implementation are given in Appendix A.2.

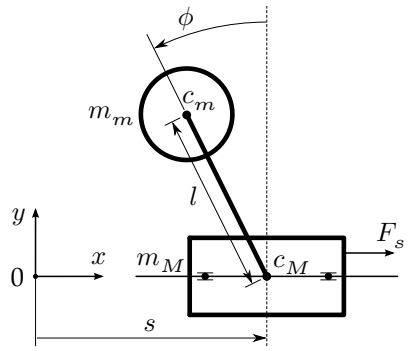


Figure 3.2: The inverted pendulum on a movable cart.

3.4 Problem formulation

In this section, we provide formulations of the objective function used in the OC, NMPC and RL problems.

We investigate control scenarios for swing-up motions of the cart-pendulum system from the given initial state $\boldsymbol{x}(0) = [s(0), \phi(0), \dot{s}(0), \dot{\phi}(0)]^\top = [0, \pi, 0, 0]^\top$, which implies the system starts from rest, with the cart in the origin of the coordinate system and the pendulum pointing downwards. The goal of the task is to swing the pendulum up and to drive the cart back to the origin, i.e., to reach the final state $\boldsymbol{x}(T) = [s(T), \phi(T), \dot{s}(T), \dot{\phi}(T)]^\top = [0, 0, 0, 0]^\top$. This is realized for both the OC and the NMPC problem by the Lagrange term in the objective function

$$L(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \|\boldsymbol{x}(t) - \bar{\boldsymbol{x}}(t)\|^2_{\mathbf{W}} + \|\boldsymbol{u}(t)\|^2_{\mathbf{V}} \tag{3.4}$$

as defined in (3.2), where the weights $\mathbf{W} = \mathrm{diag}(1, 0.5, 2, 0.2)$ and $\mathbf{V} = \mathrm{diag}(0.0005)$ were chosen to scale the state elements to approximately the same range. Set-point $\bar{\boldsymbol{x}} \equiv [0, 0, 0, 0]^\top$ is set according to the definition of the task. The benchmark constraints defined in Appendix A.2 can be directly formulated as path constraints (3.1d) on both states and controls, while the prediction horizon of the NMPC controller is a subinterval of the problem horizon.

The discount rate γ, which is inevitable for solving a continuing task in RL, affects the obtained RL solution. Therefore, to make the NMPC and RL results comparable, we include the same discount rate value into the objective function of OC and NMPC. The effect of the discount on the NMPC formulation is that it increases the focus on the beginning of the horizon by providing a weighting over time, i.e., the further the event lies in the future of the horizon, the less it will be considered for the computation of the optimal behavior.

To allow solving the control problem in real-time using NMPC alone and together with MHE, the problem formulation has to be adapted for the current algorithmic setup in the optimal control software package MUSCOD-II (Leineweber et al., 2003a,b). Due to the nonlinear behavior of the cart-pendulum system, at least a control rate of 40Hz has to be chosen to generate sufficient contraction in the real-

time iteration scheme and to enable the standard structure exploitation for the sequential quadratic programming method. However, this increases the number of shooting nodes and the computational time. Therefore, the horizon was set to 3 s.

In RL, we construct the reward from the same Lagrange term (3.4), but we additionally add a negative reward and a shaping function $\Psi(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})$:

$$r(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) = \begin{cases} -1000 & \text{if } \boldsymbol{x}_{k+1} \in \mathcal{X}^{\mathrm{a}}, \\ -L(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) + \Psi(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) & \text{otherwise}, \end{cases} \tag{3.5}$$

where $\mathcal{X}^{\mathrm{a}}$ is a set of absorbing states that lie outside of the cart's position constraints defined in Appendix A.2.

The shaping function denoted by $\Psi(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})$ leaves the target objective unchanged but allows to reduce steady-state error. Due to the quadratic terms in the definition of $L(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k)$, rewards become small for balancing states where all elements of the state are close to zero, except possibly the cart position s_k. This effect has previously been described by Engel and Babuska (2014), where authors noticed that the quadratic reward, the ℓ_2-norm, penalized large velocities much more than small steady-state errors. They solved the issue by showing that the absolute value reward, the ℓ_1-norm, yielded a response with negligible errors. This solution is not directly applicable here, because our aim is to obtain results as similar to OC as possible, which uses a quadratic cost function. Instead, inspired by Engel and Babuska (2014), we introduce a potential-based shaping function (Ng et al., 1999) encoded as $\Psi(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) = \gamma P(\boldsymbol{x}_{k+1}) - P(\boldsymbol{x}_k)$, where potential $P(\boldsymbol{x}_k) = \psi \left\| \mathbf{W} \boldsymbol{x}_k \right\|_1$ is the sum of absolute values of weighted state elements multiplied by a shaping weight ψ. The purpose of this shaping function is to provide a stronger guidance towards $\boldsymbol{x} = [0, 0, 0, 0]^\top$ in the region of the state space where the quadratic reward function fails to do so. Influence of the shaping function is analyzed in Appendix B.1.

It is possible to include hard constraints directly into the OC formulation by equation (3.1d). However, in the RL formulation, they have to be reformulated as soft constraints, which is done by including them directly in the reward function in the form of a negative reward as in (3.5). This essentially changes the original optimization problem by introducing a trade-off between receiving positive rewards and avoiding negative ones. For example, once a very large negative reward is received, it will force the system to never violate this constraint again, even at a price of obtaining lower positive rewards. On the contrary, a small negative reward will allow infrequent violation of the constraint, which will slow down learning and may even damage a real-world system. In this benchmark example, the trade-off has a mild effect, because the cart position constraints are rather loose. While constraints are violated a few times in the process of learning, the final result is free of constraint violations.

For the cart-pendulum benchmark, the optimal combination of parameters can be found in Table 3.1. For NMPC, the tolerances were chosen according to best practices, the horizon length as well as the sampling period were chosen such that "iNMPC" uses the same formulation as "OC" and that "NMPC" computes feedback in less than 5 ms. The discount rate γ was chosen according to the RL formulation. For RL, we found the parameters using the exhaustive grid search. It generated tu-

ples of candidate parameters by selecting them from a set of predefined parameter values commonly used in the actor-critic literature, c.f. Grondman et al. (2012).

For the RL policy and value function approximation, we used tile coding with 16 tilings, each of size $[2.5, 0.1\pi, 2.5, 0.5\pi]^\top$. However, pendulum states close to the state $\boldsymbol{x} = [0, 0, 0, 0]^\top \in \mathbb{R}^{n_x}$ require a finer resolution of the function approximator. Therefore, before projecting a state on the tiles, we rescale each state element to the interval $[-1, 1]$, and then apply a squashing function with the parameter $\omega = 5$:

$$\Omega(x_j, \omega) = \frac{(1+\omega)x_j}{1+\omega|x_j|}, \quad \forall j \in \{1, \ldots, n_x\},$$

where x_j denotes a scaled element. This effectively controls resolution by a multiplier that varies continuously, from $(1+\omega)^{-1}$ in the downward position of the pendulum, to $1+\omega$ in the balancing state $\boldsymbol{x} = [0, 0, 0, 0]^\top$.

3.5 Evaluation protocol

3.5.1 Notations and methodology

As summarized in Figure 3.1, we use the "OC" notation to denote the optimal solution obtained by off-line optimal control. The cost of this solution serves as a baseline for all subsequent methods. As structural uncertainties, we consider uncertainties that originate from the lack of knowledge about the true physics of the underlying dynamic system. Examples in walking robots might include model-plant mismatch due to uneven floor, friction in joints, softness of the ground, etc. Being unaware of possible uncertainties in a system, the following three *frozen* methods are not explicitly equipped with an ability to adapt to the system:

- "iNMPC" denotes an ideal NMPC controller that neglects computational time constraints and returns an optimal control signal immediately.
- "NMPC" denotes an NMPC controller tuned to real-time performance for the specific task.
- "RL" denotes an RL controller that plays the optimal policy π^* after having learned on an ideal system.

Neither "iNMPC" nor "NMPC" apply MHE for state and parameter estimation.

As parametric uncertainties, we consider parameters which are included in the dynamic model of the system and whose values are not known *a priori*, but can be inferred from interactions with the real system, i.e., the parameters are observable. Accordingly, by the term *adaptive* we denote methods that have (in the case of NMPC) been explicitly equipped with knowledge about the uncertainties and an algorithm to adapt to them, or that are (in the case of RL) given additional time to interact with the system:

- "iNMPC-adapt" denotes a controller of a combination of both MHE and NMPC. The controller is able to estimate a specified unknown parameter of a model and adjust its control signal accordingly.

Table 3.1: Parameters of OC, NMPC and RL for the cart-pendulum problem. The first group of parameters is relevant to OC and NMPC, where the value in brackets is given for real-time NMPC. The second group of parameters is relevant only to RL.

Parameter		Value
OC/NMPC:		
Horizon length	T	$5\,\mathrm{s}$ $(3\,\mathrm{s})$
Discount rate	γ	0.99
Sampling period	T_s^{OC}	$0.05\,\mathrm{s}$
KKT-Tolerance		10^{-7}
Integration accuracy		10^{-6}
RL:		
Episode length	T	$5\,\mathrm{s}$
Discount rate	γ	0.99
Sampling period	T_s^{RL}	$0.05\,\mathrm{s}$
Number of learning episodes		$2.0 \cdot 10^5$
Additional learning episodes		5%
Eligibility discount rate		0.65
Exploration variance		$0.20^2\, u_{\mathrm{max}}^2$
Critic learning rate	α^{c}	0.10
Actor learning rate	α^{a}	0.01

- "NMPC-adapt" denotes a combination of both MHE and NMPC tuned to real-time performance for the specific task.

- "RL-adapt" denotes the RL controller that is initialized using the optimal policy π^* learned by "RL" on the ideal system. To cope with uncertainties in the system, we allow "RL-adapt" to learn for an additional small number of episodes, c.f. Table 3.1.

Note that the NMPC approach requires explicit specification of the parameters to be estimated in the model, while RL can cope with them without explicit consideration.

3.5.2 Description of experiment and measures

For the benchmark problem, we provide a comprehensive comparison of the described methods for an ideal system, and for a system with structural and parametric uncertainties.

First, we investigate whether the three frozen methods produce similar trajectories on our benchmark system. For that we employ the coefficient of determination, R^2, as a similarity measure of trajectories. The measure quantifies the deviation of the trajectories obtained by "RL", "iNMPC", and "NMPC" from an optimal trajectory. Denoting a trajectory ζ as a sequence of states and controls, $\zeta = \{\zeta_k\}$, $0 \leq k \leq K$ and $\zeta_k = (\boldsymbol{x}_k^\top, \boldsymbol{u}_k^\top)^\top$, we measure similarity between corresponding components by means of R^2. Formally, the R^2 measure is defined as

$$R^2 = 1 - \frac{\sum_i^K (\zeta_i^\ddagger - \zeta_i^{\text{OC}})^2}{\sum_i^K (\zeta_i^\ddagger - \bar{\zeta})^2}, \quad \bar{\zeta} = \frac{1}{K} \sum_i^K \zeta_i^\ddagger,$$

where $\ddagger$ is a wildcard for one of {"RL", "RL-adapt", "iNMPC", "iNMPC-adapt", "NMPC", "NMPC-adapt"}. For the "RL" method, which exhibits variability in trajectories, we compute both the R^2 measure of the mean trajectory, and the mean of R^2 values obtained across individual trajectories.

Second, after the similarity of the methods is verified, we employ regret as a measure to evaluate the performance of the methods against uncertainty $\boldsymbol{\rho}$ which, in our experiments, is the viscous friction coefficient at the rotary joint, $\boldsymbol{\rho} = [\mu]$. Regret is commonly used in evaluation of online machine learning and optimization methods (Shalev-Shwartz, 2012; Kaelbling et al., 1996). It quantifies the amount of additional cost which is incurred due to suboptimal actions taken by a controller with respect to the optimal control actions. Lower values of regret indicate a controller, whose behavior is closer to the optimal one. Since the optimal controller has zero regret, it becomes convenient to plot regrets of the methods instead of the direct costs.[2]

We compute the regret $R^\ddagger(\zeta; \boldsymbol{\rho})$, defined as the difference between $\mathcal{L}^\ddagger(\zeta)$, the total cost of the method denoted by the wildcard $\ddagger$, and the baseline $\mathcal{L}^{\text{OC}}(\zeta; \boldsymbol{\rho})$, i.e.,

$$R^\ddagger(\zeta; \boldsymbol{\rho}) = \mathcal{L}^\ddagger(\zeta) - \mathcal{L}^{\text{OC}}(\zeta; \boldsymbol{\rho}),$$

where we use the NMPC cost (3.2) for all methods directly,

$$\mathcal{L}(\zeta) = \sum_{i=0}^{K} \gamma^i L(\boldsymbol{x}_i, \boldsymbol{u}_i) T_s. \tag{3.6}$$

By means of T_s, we take into account the possibly different sampling periods of the methods. Note that all of the tested methods are unaware of the true extent of the uncertainty introduced. In this chapter, we pursue a generic comparison across methods and benchmarks. Therefore, we do not include specific stability measures, such as Lyapunov stability, orbital stability, or gait sensitivity norm, but rather stick to a general notion of the cost of trajectories.

Due to the stochastic nature of RL, we plot a mean value of the regret averaged over 50 runs. If a run for a given uncertainty $\boldsymbol{\rho}$ is prematurely terminated due to the violation of constraints, then the corresponding value is not drawn.

[2] Note that for "RL" and "RL-adapt" methods the regret is calculated during the assessment run, i.e., after the learning or adaptation has been completed.

3.6 Results on the cart-pendulum

In order to assess the similarity of the frozen methods, we analyze their performance for the cart-pendulum system without uncertainties. The resulting trajectories and the R^2 measure results are shown in Figure 3.3 and Table 3.2, respectively. All three methods show a qualitatively similar behavior and are successful in swinging the pendulum up and balancing it there. An overall similarity between "RL" and "iNMPC" of more than $90.3\,\%$ was achieved in terms of the R^2-measure. Comparing the R^2 values of a mean trajectory with the mean of R^2 values of the individual trajectories of "RL", we observe that the mean trajectory is closer to "iNMPC", while the individual trajectories exhibit some variability around their mean. The control trajectories F_s of "iNMPC" and "RL" differ in a small time delay and are otherwise comparable. However, after approximately 1.0 s, we find that "RL" demonstrates variability between control trajectories compared to "iNMPC". Due to differences in control actions, the state trajectories start to differ slightly after approximately 0.5 s and recover from that after approximately 2.0 s; only the "RL" cart position s remains to show small steady-state errors. "NMPC" results deviate more from "iNMPC", and an overall similarity of approximately $48.2\,\%$ was achieved in terms of the R^2-measure.

Next, we show the behavior of both frozen and adaptive methods under the effect of uncertainties. Simulation results against variations of the friction parameter μ are shown in Figure 3.4. To indicate the scale of the plot, we employ three filled markers at $\mu = 0\,\mathrm{N\,s\,m^{-2}}$ that correspond to the trajectories in Figure 3.3. A further reference for interpretation of the scale: If the cart stood still and the pendulum hung down, then the regret of the solution would be equal to 11.73. In the absence of friction, "iNMPC" does not reach zero regret due to numerical approximations. All of "RL", "iNMPC" and "NMPC" show a similar asymptotic behavior in reaction to the variation of the friction coefficient. "iNMPC" shows the lowest regret for low values of viscosity and "NMPC" regret is the largest. Larger values increase the regret, and for the value of $\mu = 0.09\,\mathrm{N\,s\,m^{-2}}$, "RL" yields a lower regret than "iNMPC". All three frozen methods violate the position constraints of the cart. For "RL", this happens at $\mu = 0.12\,\mathrm{N\,s\,m^{-2}}$ and for "iNMPC" and "NMPC" at $\mu = 0.18\,\mathrm{N\,s\,m^{-2}}$ and $\mu = 0.19\,\mathrm{N\,s\,m^{-2}}$, respectively.

The adaptive methods, "RL-adapt", "iNMPC-adapt", and "NMPC-adapt", show a different reaction to the variation of the friction coefficient. Both "iNMPC-adapt" and "NMPC-adapt" show a constant performance under the effect of the variation of friction. Note the logarithmic scale; the variances in performance of "iNMPCadapt" and "NMPC-adapt" are similar, but appear differently due to the logarithmic scale. "RL-adapt" performs better than "NMPC-adapt" for smaller uncertainties of up to $0.07\,\mathrm{N\,s\,m^{-2}}$ and is then outperformed by NMPC. "RL-adapt" regret is an order of magnitude higher than the regret of "iNMPC-adapt", and the RL performance deteriorates with higher friction. For friction coefficients larger than $0.09\,\mathrm{N\,s\,m^{-2}}$, "RL-adapt" shows a much higher variance in regret than for lower friction.

Comparing "RL-adapt" with the frozen method "iNMPC", the graphs show that

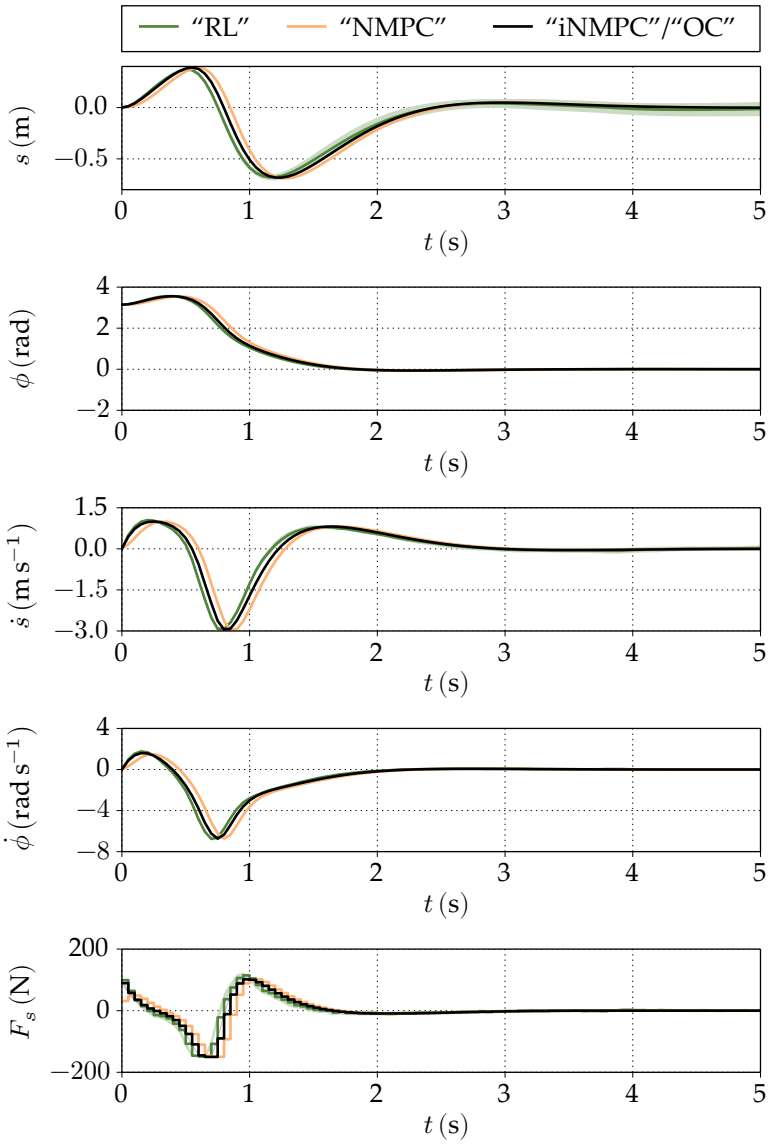


Figure 3.3: State and control trajectories obtained by the frozen methods for the cart-pendulum system without uncertainties. For "RL" the mean and standard deviation of 50 trajectories is shown. Y-axes variables s, $\dot{s}$ denote the cart position and velocity and ϕ, $\dot{\phi}$ are the respective quantities of the pendulum. F_s is the force acting on the cart body.

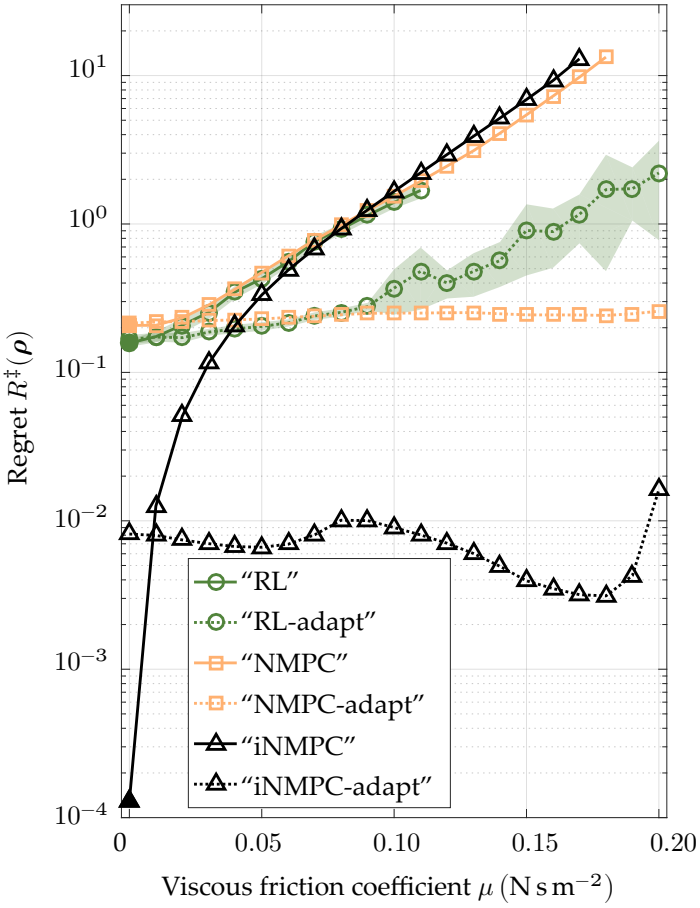


Figure 3.4: The graph of regrets for the cart-pendulum system that shows the optimal performance of the described methods. The means with the upper and lower 95% confidence limits are shown for controllers with a stochastic component (50 samples per viscous friction coefficient were used).

Table 3.2: Similarity of the cart-pendulum trajectories in terms of the coefficient of determination (R^2). For RL we first report similarity of the mean trajectory, and second we report the mean of R^2 values.

Methods		s	ϕ	$\dot{s}$	$\dot{\phi}$	F_s
"RL"- "OC", R^2 mean trajectory	(%)	98.1	99.8	96.4	98.4	92.9
"RL"- "OC", mean R^2	(%)	93.9	99.8	95.3	98.2	90.3
"NMPC"- "OC"	(%)	56.6	98.6	66.7	86.9	48.2
"iNMPC"- "OC"	(%)	100.0	100.0	100.0	100.0	100.0

"iNMPC" cannot compete with RL after the break-even point at $0.04\,\mathrm{N\,s\,m^{-2}}$. This viscous friction coefficient value corresponds to $6.1\,\%$ of the difference in energy consumed by the ideal ($0.00\,\mathrm{N\,s\,m^{-2}}$) and disturbed ($0.04\,\mathrm{N\,s\,m^{-2}}$) system. Compared to all three frozen methods, "RL-adapt" performs much better after the break-even point, and the gap grows with larger uncertainties.

A summary of the main results of comparison is presented in Table 3.3.

3.7 Discussion

Our results demonstrate that, with a proper formulation of the optimal control task, it is possible to obtain similar results for the three frozen methods on an ideal system. For the cart-pendulum benchmark example, a good similarity between "RL", "OC" and "iNMPC" was achieved. However, for "NMPC", the expected deviation from the optimal solution due to the tuning towards a real-time feasible controller is observed in Figure 3.4. We have to stress that this is a problem of the implementation and not of the approach. A speed-up of the implementation by using a multi-level real-time iteration scheme (Bock et al., 2007; Kirches et al., 2012), by using a state-of-the-art sequential quadratic programming method tailored for multiple-shooting (Janka et al., 2016) and replacing the quadratic program solver (Schork, 2015) could address the current time limitations. With a speed-up of the computations, a theoretical coverage of the area between the curves of "iNMPC" and "NMPC" is therefore possible. This will however not influence the already found break-even points, because the asymptotic behavior of the frozen NMPC methods is determined through "iNMPC" as the best possible outcome. Considering this, only an improvement for smaller variations of the friction coefficient is to be expected.

The adaptive methods successfully avoid constraint violations and substantially reduce regret compared to their frozen counterparts over the whole range of viscous friction coefficients. Interestingly, when the coefficient is not present in the system or has low values, "iNMPC-adapt" results in a higher regret than "iNMPC", and the same holds for the fast versions of NMPC. The reason for this is that the combination of NMPC and MHE in the form of "iNMPC-adapt" and "NMPC-adapt" starts with an initial guess of the parameter that is adapted during the actual run

of the system. Any mismatch between measurements and values predicted by the model will lead to adaption of the parameters in MHE. This is a crucial difference to RL, which adapts to the uncertainty through multiple trials prior to an assessment run, while NMPC is unaware of the mismatch at first. The cart-pendulum task is very sensitive to changes during movement initiation. Therefore, a wrongly identified parameter in the beginning can already cause substantial differences in terms of regret, which is seen in Figure 3.4. This effect is amplified for "NMPC-adapt" through the mentioned performance loss due to the real-time feasibility tuning. A speed-up of the computations would lead to a theoretical coverage of the area between the curves of "iNMPC-adapt" and "NMPC-adapt", boosting performance.

In the absence of uncertainties, "iNMPC-adapt" performs superior to both "RL" and "RL-adapt". This does not come as a surprise, as NMPC methods were running off-line, they were using the model of the correct system and, moreover, the uncertain parameter was defined explicitly. However, "iNMPC" outperforms RL methods only for small values of uncertainties. In case of medium and large uncertainties, there exist break-even points after which "RL" and "RL-adapt" obtain lower regret. We remark that the estimation of the difference between ideal and uncertain systems in terms of energy is ad hoc, and more generic measures for model uncertainties should be used in the future.

In the non-ideal setting, performance of NMPC becomes comparable to RL. Nonetheless, one cannot directly report similarity of "NMPC-adapt" and "RL-adapt"; while regret of "NMPC-adapt" is almost constant for the whole range of uncertainties, the regret of "RL-adapt" significantly increases. The explanation for this effect is twofold. First, for any value of uncertainty, "RL-adapt" was learning for a fixed additional 5 % of time. The larger the value of uncertainty, the more time "RL-adapt" requires to adapt to a new parameter value. This can be supported by the fact of an increasing variance of RL regret, which indicates that the actor-critic algorithm simply did not have enough time to converge in areas of high uncertainties. Second, for large uncertainties, it might be necessary to significantly change the control strategy, i.e., to learn a new policy rather than adapt an ideal one. This will probably require more learning efforts, to first unlearn the initial policy, and then to learn a realistic one.

Several issues were encountered while formulating the benchmark problem with the aim of obtaining identical results. These issues are known to OC and RL communities, but, to the best of our knowledge, they were never explained in the same context before.

OC-related issues:

1. In contrast to RL, the derivative-based methods of optimization used to solve the discretized OC problem require a continuously differentiable formulation of the problem.

2. The performance of the ideal NMPC-MHE combination ("iNMPC-adapt"), for which computational time was neglected, is the order of magnitude better in terms of regret than the corresponding real-time version, mainly caused by a shortened prediction horizon used in the latter.

Table 3.3: Summary of results.

Category	Findings
Achieved similarity of "iNMPC" and "RL" methods on the ideal system	more than $90.3\,\%$
Break-even point: the difference in energy consumed by ideal and noisy systems after which "RL-adapt" performance becomes better then "iNMPC"	$6.1\,\%$
Best performing algorithm under parametric uncertainties	"iNMPC-adapt"
Best performing algorithm under structural uncertainties	"iNMPC" before the break-even point and "RL-adapt" after the break-even point

RL-related issues:

1. In this chapter, we use a model-free RL method, which means that transition model of a system is unknown *a priori*.
2. Learning a solution with a quality comparable to OC takes many episodes.
3. Constraints in the original OC problem are included into the RL formulation by means of negative rewards received for violating the constraints.
4. For the benchmark example, the OC objective function has been modified. Formulating a reward function by simply negating the OC objective results in a) a very slow learning in cases when no negative reward is used, b) an inability to learn or even a divergence of the value function if $\gamma = 1$.
5. For symmetrical problems, RL can use state space reduction techniques. For example, for the cart-pendulum example it is possible to wrap the pendulum angle to the $[-\pi, \pi)$ interval, which results in two equally possible optimal trajectories under our objective function. OC generally does not allow implementation of such techniques if they violate the smoothness assumptions.
6. When learning with a quadratic objective function, which is often used in OC, it is useful to implement learning techniques that are able to reduce steady-state error while leaving the objective function unchanged, for example reward shaping.

The presented quantitative comparison is particularly important for our future plans of combining RL and NMPC to control a more complex system with a high number of degrees of freedom. One possible combination could be that RL learns a real model for NMPC, while NMPC provides a backup of an RL exploratory policy.

Another scheme could be that RL receives a control signal from NMPC as a suggestion. Initially RL passes this suggestion to the actuators, but at a later stage it takes over in state space areas where it is confident. Independently of the chosen combination strategy, for value function-based RL it is important to retain the Markov property, which may impose restrictions on the NMPC controller as well. For example, such RL methods usually avoid time as a state, hence, the trajectory-tracking NMPC should not be used in the suggestion-based scheme.

3.8 Conclusion

In this chapter, we provided an extensive comparison of model-free RL and model-based NMPC methods. We began with finding a proper formulation of NMPC and RL problems tackling the same task of a swing-up and balancing motion of a cart-pendulum system. The benchmark is standard and well-known in literature. To facilitate follow-up research, we provide the freely available source code of the benchmark online (Caarls, 2015).

We showed that both methods were capable of solving the benchmark problem and that the resulting trajectories for states and controls were similar in terms of the coefficient of determination and regret.

In our experiments considering uncertainties, we showed that ideal NMPC with MHE is superior to RL for the whole range of uncertainties, but the realistic NMPC with MHE is comparable to RL. The major achievement is a quantification of a break-even point after which learning in a model-free setting becomes more beneficial than nonlinear model predictive control with an inaccurate model.

We expect that a proper combination of these methods will open the door to novel control strategies. In particular, we plan to develop a hybrid NMPC-RL controller and test it on a real seven-degree-of-freedom walking robot, specifically designed for the purpose of learning with RL.

4

Model-plant mismatch compensation using reinforcement learning

This chapter further addresses research question 2, where the proposed combination of model-based optimal control (OC)/nonlinear model predictive control (NMPC) and model-free reinforcement learning (RL) is implemented. To compensate for the model-plant mismatch, two combination approaches are studied in simulation. The first approach learns a compensatory control action that minimizes the same performance measure as is minimized by NMPC. The second approach learns a compensatory signal from the difference of the state transition predicted by the NMPC internal model and the actual state transition. A theoretical justification of the approaches is provided. In simulated experiments, the second approach showed a better performance, and therefore it was verified on the real robot Leo. The exploration strategy was chosen according to the results obtained in Chapter 1. Even though safety is not guaranteed by the proposed approaches, the results demonstrate the usefulness of NMPC, which provides safety barriers to constrain RL exploration actions near dangerous state-space regions.

Published in: Koryakovskiy, I., Kudruss, M., Vallery, H., Babuska, R., and Caarls, W. (2018). Model-Plant Mismatch Compensation Using Reinforcement Learning. *IEEE Robotics and Automation Letters*, 3(3):2471–2477.

4.1 Introduction

Mechanically and electronically, robotics have advanced to the point where cognitive abilities have become the main limiting factor. While robots can flawlessly execute a set of commands to achieve a task, these commands are mostly encoded or tuned by hand. RL allows to find an optimal sequence of commands without any prior assumption about the world. However, the application of pure learning to real systems is very limited due to intrinsically damaging exploratory policies. For example, when learning from scratch the robot Leo depicted in Figure 4.1 can withstand only five minutes of operation due to large and rapidly changing motor torques and frequent falls (Schuitema, 2012).

Usually, the dynamics of physical systems are known, but various uncertainties do not allow achieving optimal performance with model-based control methods, see Chapter 3. Whereas for the estimation of parametric uncertainties moving horizon estimation (MHE) techniques (Kühl et al., 2011) can often be employed, for structural uncertainties, such as backlash, Coulomb friction or wear and tear, this is not easily possible. Nevertheless, model-based methods can predict the evolution of the system for a short horizon, thus enabling the implementation of safety barriers to limit risky exploration in dangerous state space regions.

By safety, we mean the prevention of actions that cause damage to the system. For example, in a bipedal robot, safety is particularly related to the robot not falling. Falls result in impact forces applied to the limbs and gearboxes, and some robots cannot withstand even a single fall. In model-based control, it is natural to constrain the angles and velocities to remain within an admissible range, and to enforce static stability constraints such that the center of mass projection is some distance away from the support polygon border. These constraints help prevent falls, but a momentary violation of them does not necessarily result in the one. In RL, it is possible to consider angle and velocity constraints by means of negative rewards. However, to learn avoiding such constraints, they need to be violated multiple times in different robot configurations. Random exploration exacerbates the problem and can lead to a very large number of falls.

Therefore, we propose to combine RL and NMPC in one framework that allows RL to gather the required experience without damaging a many-degree-of-freedom system. The experience is used by RL to compensate the difference between the internal model of the system and the real one. Any model-based nominal controller is suitable, but the choice of NMPC is particularly motivated by the complexity of the robot.

We propose two different approaches shown in Figure 4.2. Similarly to Bayiz and Babuska (2014), the first approach learns a compensatory control action, but instead of a proportional-derivative (PD) controller, we use NMPC, which introduces an additional optimization problem. Since both NMPC and RL optimize similar performance measures, the obtained policy is optimal with respect to the real system. The second approach learns a compensatory signal from the difference of transitions predicted by the internal model and the actual transition. In this case, RL uses a different optimization goal, which does not divert NMPC from reaching

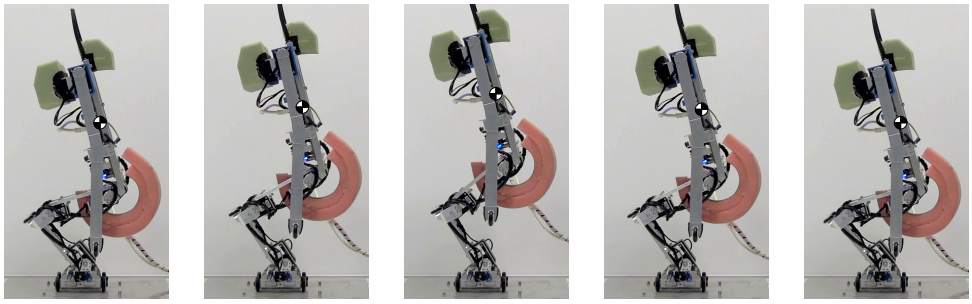


Figure 4.1: Robot Leo performs up and down motions. Root point is shown by the circle with black and white sectors.

its objective. As a result, the model-plant mismatch is eliminated by forcing the real system to behave as if it has no uncertainties.

We conduct simulated and real experiments with Leo and demonstrate the advantage of our proposal in the presence of temperature- and torque-dependent Coulomb friction.

4.2 Related work

From a control theoretic viewpoint, our approaches can be compared to adaptive internal model control (IMC) (Datta and Xing, 1998). The implementation requires an explicit model of the plant to be used as part of the controller. However, in adaptive IMC, the structure of the unknown system is determined offline, while its parameters can be inferred by online parameter estimation (Kühl et al., 2011). A particular shortcoming is that the structure needs to be identified precisely; otherwise a model-plant mismatch remains. The proposed approaches require neither precise identification of the structure nor of the parameters.

As a learning controller, we employ model-free on-policy deterministic policy gradient (DPG) by Silver et al. (2014). In principle, any model-free RL algorithm can be used, e.g. Kakade (2002); Schulman et al. (2015, 2016, 2017). However, lack of safety measures and sample complexity of the algorithms limits their application to real systems.

Learning the forward model of the system demonstrates the lowest number of interactions with it (Kamthe and Deisenroth, 2017; Gu et al., 2016). Learning the inverse model (Kawato, 1990; Christiano et al., 2016; Gamboa Higuera et al., 2017) assumes that the model can connect successive states prescribed by the nominal controller.

When the approximate model of the system is available, it is possible to pre-train the initial policy, which can speed up learning. The two-step sequential approach is proposed by Farshidian et al. (2014). First, an iterative linear-quadratic-Gaussian algorithm is used to design an initial policy. Then, the policy is refined using the PI² algorithm on the real system. Another approach is to iteratively learn the difference model of the measured state and the state obtained on the approximate model

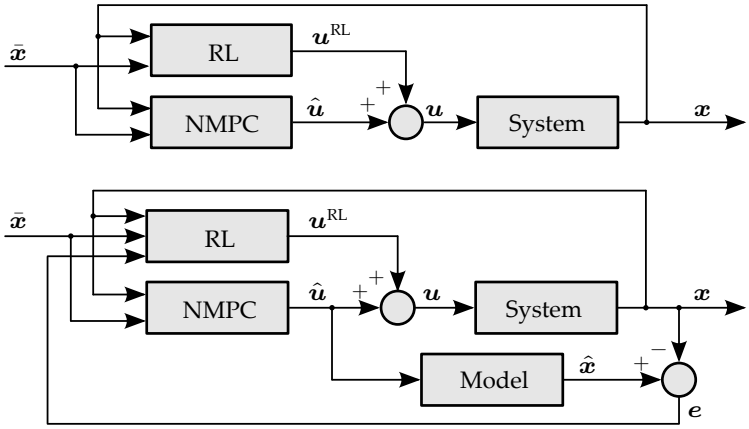


Figure 4.2: (*top*) Compensatory Action Learning, and (*bottom*) Model-Plant Mismatch Learning.

and adopt this difference model for improving the policy (Ha and Yamane, 2015; Saveriano et al., 2017). Finally, Rajeswaran et al. (2017) use an ensemble of slightly perturbed model parameters to learn a robust policy.

Learning involving off-line planning or human-expert demonstrations (Schuitema, 2012; Abbeel et al., 2010; Peters et al., 2010; Levine and Koltun, 2013; Zhang et al., 2016) constrains the problem space, thus reducing hardware damage. This option requires either a hand-coded suboptimal policy or a skilled human operator.

For a bipedal robot where any failure can be catastrophic, the above methods are not suited even given a good starting policy, because it is likely that RL will result in at least several failures during subsequent policy improvement episodes.

It is possible to guarantee safe learning when one can either predict repercussions of bad actions (Moldovan and Abbeel, 2012) or has a backup policy to lead the system back to safe states (Hans et al., 2008; Fisac et al., 2017). Here, we do not guarantee safe learning, though the proposed approaches in practice can be safe.

Our contribution is twofold. First, we propose approaches that can in principle compensate any type of uncertainty which preserves the Markov property, without a time-consuming structure identification process and expert-designed models of friction, backlash, etc. The approaches can be implemented on top of an existing model-based controller which makes it easier to integrate into the various fields of engineering, such as robotics, chemistry or computer science. Second, we demonstrate successful learning results on a real robot.

4.3 Background

4.3.1 Problem statement

Consider the nonlinear time-invariant system in the form of

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\rho}) \tag{4.1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ the system state at time t, $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$ is the control vector of joint motor voltages applied to the system at time t, and $\boldsymbol{\rho}$ is an unknown structural uncertainty. The presence of uncertainty causes the model-plant mismatch $\boldsymbol{e}$ which is formulated as the difference between the real system state $\boldsymbol{x}$ and the simulated state of the model $\hat{\boldsymbol{x}}$, see Figure 4.2. We do not make any assumption on how the uncertainty enters the equations. Thus it represents the general concept of mismatch.

4.3.2 Nonlinear model predictive control

The nominal feedback is provided by NMPC, a closed-loop control strategy in which the control action is computed from the current system state by solving an open-loop optimal control problem on a finite prediction horizon $[0, T]$ online,

$$\min_{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)} \quad \int_0^T L(\boldsymbol{x}(t), \boldsymbol{u}(t))\, \mathrm{d}t \tag{4.2a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), 0), \tag{4.2b}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0,$$

$$\boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \geq 0. \tag{4.2c}$$

Here we strive to find a control trajectory $\boldsymbol{u}(t)$ such that an objective function composed of a Lagrange term $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is minimized. The state trajectory $\boldsymbol{x}(t)$ is characterized by the dynamic system (4.1). In (4.2b), we assume the idealized model, $\boldsymbol{\rho} = 0$, because nothing is known about the uncertainties in the real system. In addition, we impose mixed state-control path constraints $\boldsymbol{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$ such as constraints on joint angles ensuring the static stability of the robot together with constraints formulating limits on the maximum motor input voltage.

In the simulated experiment, we use a nominal NMPC scheme that is based on direct multiple shooting (Bock et al., 2007). Controls $\boldsymbol{u}$ are approximated as piecewise constant functions. The cost of some discretized trajectory $\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{x}_1, \boldsymbol{u}_1, \ldots$ obtained using policy $\boldsymbol{u}_k = \pi(\boldsymbol{x}_k)$ is denoted as $\mathcal{L} = \sum_k L(\boldsymbol{x}_k, \boldsymbol{u}_k)$.

To achieve real-time control on the robot, we implement a parallelized version of the NMPC scheme proposed by Kudruss et al. (2018), where one controller provides a fast feedback by efficiently reusing control problem linearizations of the last iteration, while the second controller prepares the next nonlinear step.

4.3.3 Reinforcement learning

RL is a trial-and-error method which does not require an explicitly given model, and can naturally adapt to uncertainties in the real system (Sutton and Barto, 1998). RL assumes the system is stochastic, and thus it maximizes the expected discounted return

$$G_k^\gamma = \mathbb{E}\left\{ \sum_{i=0}^{\infty} \gamma^i r(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}, \boldsymbol{x}_{k+i+1}) \right\} \tag{4.3}$$

$$r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) = \begin{cases} -L(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) & \text{if } \boldsymbol{g}(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) \geq 0 \\ r^{\mathrm{a}} & \text{otherwise.} \end{cases} \tag{4.4}$$

Here $r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1})$ is the scalar reward given for a transition from $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1}$ caused by the control signal $\boldsymbol{u}_k = \pi(\boldsymbol{x}_k) + \boldsymbol{n}_k, \boldsymbol{n}_k \sim \mathcal{N}$ chosen from some policy π. Discount rate $\gamma \in [0, 1)$ is required for integrability of the infinite sum. Its role is similar to NMPC horizon T. Constraints (4.2c) are established by means of the large negative reward r[a]. Subsequently, the episode is terminated, and the system is restarted in state $\boldsymbol{x}_0$. Usually, RL requires at least several repetitions to estimate the return (4.3) correctly. These repetitions are obtained by adding exploration noise $\boldsymbol{n}_k$ to control signals $\boldsymbol{u}_k$ at every time step. The outcome of repetitions is not known in advance and therefore may be damaging to the system.

An important aspect of learning is the preservation of the Markov property, which assumes that the next state $\boldsymbol{x}_{k+1}$ depends only on the current state $\boldsymbol{x}_k$ and action $\boldsymbol{u}_k$, but not on previous states or actions (Sutton and Barto, 1998).

We solve problem (4.3) using DPG with linear function approximation and compatible features, chosen for its ability to optimize continuous control policies and fast convergence.

4.4 Proposed combination approaches

4.4.1 Compensatory Action Learning

In the proposed combination schemes shown in Figure 4.2, we use $\hat{\boldsymbol{u}}$ notation for the output of the NMPC controller and $\boldsymbol{u}^{\mathrm{RL}}$ notation for the output of the RL controller.

Compensatory Action Learning (CAL) approach learns a compensatory control action added to the control input computed by nominal NMPC. For learning, we use the NMPC-inspired reward (4.4), which establishes similar optimization goals for both controllers. Due to small differences in formulation and function approximations in RL, the obtained policy might be suboptimal compared to NMPC.

4.4.2 Model-Plant Mismatch Learning

In the Model-Plant Mismatch Learning (MPML) approach, RL maximizes return (4.3) where the reward is given by

$$r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) = -\left\|\boldsymbol{e}_{k+1}\right\|_2 = -\left\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1}\right\|_2 . \tag{4.5}$$

Cumulative model-plant mismatch along some trajectory is calculated as the negative value of the undiscounted return $\mathcal{E} = \sum_k \left\|\boldsymbol{e}_k\right\|_2$.

In the following, we prove two theorems. The first one explains the behavior of the system when the model-plant mismatch is minimized by RL. The subsequent corollary considers the case when the cumulative return ($\gamma > 0$) is useful for discovering a better control policy. The second theorem specifies conditions under which the system retains the Markov property. If the property is preserved, then RL will not diverge, and the mismatch can be minimized. In this case, the MPML performance depends on the RL learning capability.

Theorem 1. *The outcome of the control policy approaches the outcome of the optimal policy with respect to the idealized model iff the model-plant mismatch* $\boldsymbol{e}_k \to 0$ *when* $k \to \infty$.

Proof. Writing the mismatch as $\boldsymbol{e}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k \to 0$ results in $\boldsymbol{x}_k \to \hat{\boldsymbol{x}}_k$. Assuming the nominal controller can reach the setpoint on the model, $\hat{\boldsymbol{x}}_k \to \bar{\boldsymbol{x}}_k$, implies that the system state will also approach the setpoint, $\boldsymbol{x}_k \to \bar{\boldsymbol{x}}_k$. Since the mismatch $\boldsymbol{e}_k$ is minimized in every point of the reference trajectory $\bar{\boldsymbol{x}}_k$, we arrive at the proof of the theorem. The same logic holds for the reverse. □

Corollary 1. *If there is a time step k such that $\boldsymbol{e}_k \neq 0 \ \forall \boldsymbol{u}_k$, then $\min_\pi|_{\gamma=0} \mathcal{E} \geq \min_\pi|_{\gamma>0} \mathcal{E}$. When the mismatch is eliminated along the reference trajectory, strict equality holds.*

The corollary is based on the RL result that larger γ improves the quality of the policy (Jiang et al., 2015). However, if there exists a control action which achieves zero mismatch in every state, then maximizing immediate rewards ($\gamma = 0$) is desirable because the problem becomes computationally easier.

In the following theorem, we assume that the system (4.1) can be discretized as $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\rho})$ and the setpoint $\bar{\boldsymbol{x}}$ can be included into the state $\boldsymbol{x}$ for simplicity.

Theorem 2. *The system controlled by the nominal controller is Markov w.r.t. RL*[1] *if (a) the system itself is Markov w.r.t. RL, $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\rho})$; (b) the internal model is Markov w.r.t. the nominal controller, $\hat{\boldsymbol{x}}_{k+1} = f(\boldsymbol{x}_k, \hat{\boldsymbol{u}}_k, 0)$; and (c) the nominal controller response $\hat{\boldsymbol{u}}_k \equiv \hat{\boldsymbol{u}}(\boldsymbol{x}_k) + \boldsymbol{m}_k$, $\boldsymbol{m}_k \sim \mathcal{M}$ is stochastic with some stationary distribution $\mathcal{M}$.*

Proof. First, by looking at the bottom diagram of Figure 4.2 we write the condition (a) and show that the distribution of states $\boldsymbol{x}_{k+1}$ is defined by the current state $\boldsymbol{x}_k$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \hat{\boldsymbol{u}}_k + \boldsymbol{u}_k^{\mathrm{RL}}, \boldsymbol{\rho}) = f(\boldsymbol{x}_k, \hat{\boldsymbol{u}}(\boldsymbol{x}_k) + \pi(\boldsymbol{x}_k) + \boldsymbol{m}_k + \boldsymbol{n}_k, \boldsymbol{\rho}).$$

Next, we show that the reward is also defined by $\boldsymbol{x}_k$.

$$r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) = -\left\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1}\right\|_2 = -\left\|\boldsymbol{x}_{k+1} - f(\boldsymbol{x}_k, \hat{\boldsymbol{u}}(\boldsymbol{x}_k) + \boldsymbol{m}, 0)\right\|_2$$

The expected return averages the sum of discounted rewards over the distribution of states and controls. Since both the dynamics and the return are predictable from the current state $\boldsymbol{x}_k$, we conclude that the system controlled by the nominal controller is Markov with respect to RL. □

Note that the real system does not have to be Markov with respect to NMPC, which means that any uncertainty $\boldsymbol{\rho}$ can be compensated. We use this observation in the real example where $\boldsymbol{\rho}$ depends on motor temperature which does not enter the model, but is used in RL as an extra state variable.

[1] The notation of some system or its model being Markov with respect to a controller means here that both the next state $\boldsymbol{x}_{k+1}$ and the cost of this transition $\boldsymbol{x}_k \to \boldsymbol{x}_{k+1}$ do not depend on how the system arrived at state $\boldsymbol{x}_k$.

In the special case of an affine system (4.1) w.r.t. controls, $\boldsymbol{x}_{k+1} = f^x(\boldsymbol{x}_k, \boldsymbol{\rho}) + f^u(\boldsymbol{x}_k, \boldsymbol{\rho})\boldsymbol{u}_k$, a perfectly learned RL control, i.e., $\boldsymbol{e} = 0$, is explicitly given by

$$\begin{aligned}\boldsymbol{u}_k^{\mathrm{RL}} = &\left(f^u(\boldsymbol{x}_k, \boldsymbol{\rho})^\top f^u(\boldsymbol{x}_k, \boldsymbol{\rho})\right)^{-1} f^u(\boldsymbol{x}_k, \boldsymbol{\rho})^\top \\ &\times \left[f^x(\boldsymbol{x}_k, 0) - f^x(\boldsymbol{x}_k, \boldsymbol{\rho}) + \left(f^u(\boldsymbol{x}_k, 0) - f^u(\boldsymbol{x}_k, \boldsymbol{\rho})\right)\hat{\boldsymbol{u}}_k\right],\end{aligned}$$

where $f^x(\boldsymbol{x}_k, \boldsymbol{\rho}) \in \mathbb{R}^{n_x \times 1}$ and $f^u(\boldsymbol{x}_k, \boldsymbol{\rho}) \in \mathbb{R}^{n_x \times n_u}$ are terms independent of $\boldsymbol{u}_k$. As expected, the control $\boldsymbol{u}^{\mathrm{RL}}$ captures the model-plant mismatch caused by $\boldsymbol{\rho}$.

4.5 Experiments

4.5.1 Bipedal walking robot Leo

Robot Leo is depicted in Figure 4.1. We focus on the task of reaching upper and lower setpoints which together realize a squatting motion. For this purpose, the robot is fixed to the ground plate below its feet. Falling situation is recognized when absolute torso angle becomes larger than 57.3 °. Leo has seven degrees of freedom (DoFs) driven by Dynamixel XM430 servo motors, three in each leg at ankle, knee and hip and one motor in the shoulder. Gearboxes are subject to Coulomb friction dependent on motor temperature and torque.

The robot state $\boldsymbol{x} = [\boldsymbol{\phi}, \dot{\boldsymbol{\phi}}, \bar{h}_\phi, \tau_{\text{knee}}]^\top$ is defined as a vector of all but shoulder joint angles $\boldsymbol{\phi}$, corresponding angular velocities $\dot{\boldsymbol{\phi}}$, setpoint height $\bar{h}_\phi \in \{0.28\,\text{m}, 0.35\,\text{m}\}$ and mean temperature of knee motors τ_{knee}.

Exploiting the symmetry of Leo, we apply the same control voltages to both legs. The shoulder is actuated using a PD controller. The setpoints are switched over when the robot root point appears to be within ± 0.01 m away from it. In the simulated experiment, we add shoulder angle and velocity to the state, and the shoulder voltage is also learned. The robot is initialized in a setpoint chosen randomly at the beginning of every episode. The idealized model is made such that no friction in joints is present. For the realistic model, we add Coulomb friction $\boldsymbol{u}_{\text{fr}} = -0.2\tanh(2000\dot{\boldsymbol{\phi}})$ in all joints.

The control delay of 13.0 ± 1.7 ms comprises measurement, computation and actuation delays. A sampling period of 33.3 ms is chosen to be larger than the control delay.

4.5.2 Objective function and constraints

The NMPC objective function is defined by (4.2a) with

$$\begin{aligned}L(\boldsymbol{x}, \boldsymbol{u}) = &0.05\,(height(\boldsymbol{\phi}) - \bar{h}_\phi)^2 + 0.10\,(x_c(\boldsymbol{\phi}) - x_{c,0})^2 \\ &+ 0.05\,(pose(\boldsymbol{\phi}) - 0.3)^2 + 0.003\,\dot{\boldsymbol{\phi}}^\top\dot{\boldsymbol{\phi}}.\end{aligned}$$

Here, the first term accounts for the vertical distance $height(\boldsymbol{\phi})$ to a setpoint $\bar{h}_\phi = height(\bar{\boldsymbol{x}})$, the second term maintains the horizontal position of the center of mass $x_c(\boldsymbol{\phi})$ close to predefined value $x_{c,0}$, the third term containing $pose(\boldsymbol{\phi}) = \phi_{\text{ankle}} + \phi_{\text{knee}} + \phi_{\text{hip}}$ is used as a regularization term improving the stability of the robot, and the last term favors small velocities.

We formulate static stability as a constraint $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u}) = (x_t - x_c(\boldsymbol{\phi}), x_c(\boldsymbol{\phi}) - x_h)^\top$, where x_t, x_h denote the position of the tip and the heel of robot feet. Additionally, robot angles and controls are subject to constraints

$$\begin{bmatrix}-1.57\\-2.53\\-0.61\end{bmatrix} \leq \boldsymbol{\phi} \leq \begin{bmatrix}1.45\\-0.02\\2.53\end{bmatrix} \qquad \{|u_j|, |\widehat{u}_j|, |u_j^{\mathrm{RL}}|\} \leq 10.0\,\mathrm{V}$$
$$j \in \{\mathrm{ankle}, \mathrm{knee}, \mathrm{hip}\}.$$

For MPML approach the reward (4.5) is calculated based on the joint angles $r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) = -\left\|\boldsymbol{\phi}_{k+1} - \hat{\boldsymbol{\phi}}_{k+1}\right\|_2$.

4.5.3 Parameters

The time horizon T for NMPC optimization is selected to be 1 s. One learning or testing episode lasts for 15 s. Advantage function, state value function and actor learning rates are chosen to be 0.01, 0.10 and 0.01, respectively. Additional parameters include discount rate $\gamma = 0.97$, and an eligibility trace decay rate of 0.65. We rely on NMPC to avoid falls of the robot, therefore negative reward r^{a} is not used. Exploration is achieved by Ornstein-Uhlenbeck (OU) noise $\Delta\boldsymbol{u}_{k+1} = 0.5\Delta\boldsymbol{u}_k + \mathcal{N}(0,\sigma)$ with $\sigma = 0.005$. For the real experiment, we select a higher advantage learning rate of 0.06 and increase exploration by using $\sigma = 0.01$.

4.5.4 Evaluation

For quantitative assessment, we evaluate objective (4.2a) separately for reaching upper and lower setpoints $\mathcal{L}^{\{\mathrm{u,l}\}} = \sum L(\boldsymbol{x},\boldsymbol{u})$. Second, we evaluate the minimization of model-plant mismatch $\mathcal{E}^{\{\mathrm{u,l}\}} = \sum \|\boldsymbol{e}\|_2$ for reaching both setpoints separately. Third, we calculate root mean squared error (RMSE) between transitions obtained by both approaches and NMPC executed on the idealized model. Finally, to experimentally demonstrate safety barriers imposed by NMPC, we calculate the cumulative number of falls and violation of NMPC constraints at multiple levels of exploration noise σ for two proposed approaches and DPG.

For qualitative assessment, we calculate the number of squats the robot performs during the testing episode. This measure should be accounted only as a learning progress indicator since it is not included in the optimization objective.

4.5.5 Simulation results

In Figure 4.3 and Table 4.1, we demonstrate a significant difference in the performance of standalone NMPC on the idealized and realistic models. On the idealized model, NMPC realizes three squats. On the realistic model, NMPC can reach neither upper nor lower switching points, which results in the inability to squat and high costs $\mathcal{L}$. This result motivates the need for an adaptive component in the controller.

To compare the performance of the proposed approaches to the baseline performance of NMPC, we perform learning for 10^6 time steps. The time was enough for CAL and MPML to converge, while DPG required about hundred times more steps. Thus, its results were excluded from the comparison.

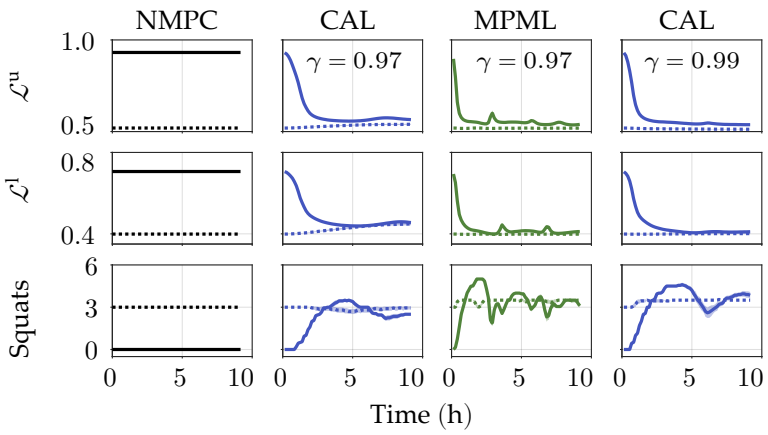


Figure 4.3: Learning to reach (*top row*) upper and (*middle row*) lower setpoints in simulation. Number of squats is given in the *bottom row*. *Dotted* and *solid* lines show learning on the idealized and realistic models, respectively. Means with upper and lower 95% confidence limits are shown for 10 runs.

We notice that on the idealized model the performance of both approaches becomes slightly worse than the baseline performance of NMPC. For CAL$^{\gamma=0.97}$, we observe deviation from the optimal policy which is seen in the increase of $\mathcal{L}$ cost. Yet, the approach is able to keep the number of squats close to the baseline value. For MPML, costs $\mathcal{L}$ do not change, but the number of squats increases by 0.5 which indicates that the approach reaches the upper setpoint right before the episode ends. Deviation of the learned trajectory from the idealized one is captured by RMSE which is nonzero for both approaches. For the CAL and MPML approaches the mean RMSE is $68.6\,\%$ and $91.7\,\%$ below the reference of 20.4 ± 0.2 which is RMSE of NMPC trajectory obtained on the realistic model.

Results of the realistic model experiment show that both approaches improve the performance of NMPC. The decrease of the $\mathcal{L}$ cost is at least $35.2\,\%$ and $41.9\,\%$ for CAL$^{\gamma=0.97}$ and MPML approaches, respectively. In terms of squats, both learning approaches overshoot the NMPC baseline of 3 squats and then continuously reduce the number towards the baseline. RMSE increases comparing to the idealized model experiment, but still remains significantly below the reference value.

To find the reason of CAL$^{\gamma=0.97}$ performance decrease on the idealized model, we test the approach with a discount rate of $\gamma = 0.99$. Increasing γ leads to a longer planning horizon which makes RL return (4.3) more similar to NMPC objective (4.2a). It turns out that CAL$^{\gamma=0.99}$ obtains lower $\mathcal{L}$ costs not only comparing to CAL$^{\gamma=0.97}$ but also comparing to the baseline NMPC. We believe the reason of this is due to the early switching of setpoints described above. While CAL$^{\gamma=0.99}$ can learn this fact, NMPC is not aware of it.

In Figure 4.4, we plot the number of falls and NMPC constraint violations accumulated over 10^6 time steps. Here, we prematurely stopped DPG for the sake of results comparability. The proposed approaches are almost identical. Both approaches prevent the robot from falling, while constraints get violated at $\sigma > 0.1$. A

Table 4.1: Final performance of methods. Significant width of the confidence interval is shown in brackets.

Method	$\mathcal{L}^{\mathrm{u}}$ $\times 10$	$\mathcal{L}^{\mathrm{l}}$ $\times 10$	Number of squats	RMSE $\times 10^4$
Idealized model				
NMPC	5.2	**4.0**	**3.0**	**0.0**
CAL$^{\gamma=0.97}$	5.4	4.4	3.0(0.1)	6.4(1.5)
MPML	5.2	**4.0**	3.5	1.7(0.1)
CAL$^{\gamma=0.99}$	**5.1**	**4.0**	3.5	11.8(4.0)
Realistic model				
NMPC	9.3	7.1	0	20.1
CAL$^{\gamma=0.97}$	5.6	4.6	2.5	10.4(0.5)
MPML	**5.4**	**4.1**	**3.2(0.1)**	**4.3(1.2)**
CAL$^{\gamma=0.99}$	**5.4**	**4.1**	3.9(0.2)	13.2(1.2)
Real robot				
NMPC 38 °C	22.0(2.1)	–	0	
MPML	**7.9(2.9)**	**4.4(1.8)**	**3.3(1.1)**	**94.9(26.5)**

different picture is seen in DPG results. The smallest number of falls and constraint violations is achieved for the value of $\sigma = 0.02$. Smaller σ reduces the learning pace, while larger values increase chances of fall.

The results of the influence of the discount rate γ on the ability of MPML to minimize the mismatch is given in Appendix B.2.

MPML learns almost twice as fast as CAL and does not exhibit deviating behavior, which are the main reasons for testing the approach on the real robot.

4.5.6 Results on the real robot

Results of standalone NMPC on Leo are shown in Figure 4.5. While on the idealized model NMPC successfully reaches switching points, on the real robot the controller is not able to do so. The reason is due to Coulomb friction in gearboxes, which depends on motor temperature and the applied torque. Modeling these effects is possible but requires a precise identification of the underlying physical processes.

To circumvent this problem, we apply the proposed MPML approach. In Figure 4.5, results of three independent runs are shown. MPML successfully realizes squatting by learning the compensation signal. The variation of motor temperature leads to noticeable differences in the trajectories. The trajectory obtained in the 3$^{\mathrm{rd}}$ run is less noisy and squatting is faster than the one achieved in the 1$^{\mathrm{st}}$ and 2$^{\mathrm{nd}}$ runs. In particular, the gradient of the downwards motion in the 3$^{\mathrm{rd}}$ run is very similar to the idealized NMPC run, except for the later part where the slow approach towards the setpoint diminishes.

Variation of motor temperature also leads to differences in the learning progress, see Figure 4.6. The 1$^{\mathrm{st}}$ and 2$^{\mathrm{nd}}$ runs require substantially longer time before the

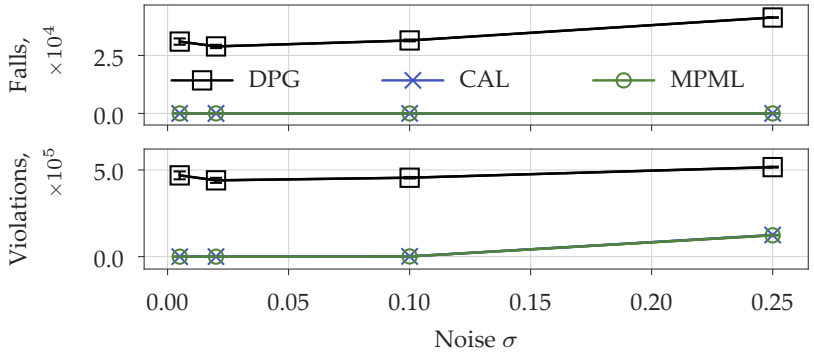


Figure 4.4: (*top*) The number of falls and (*bottom*) NMPC constraint violations as functions of σ. Means with upper and lower 95% confidence limits are shown for 10 runs.

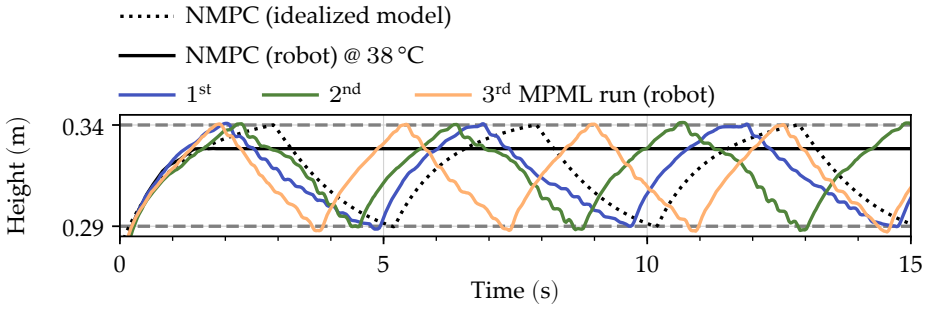


Figure 4.5: Robot root point trajectories obtained after learning.

squatting cycle is observed. This is due to the increase of motor temperature above 40.0 °C which requires additional exploration of the state space. Nevertheless, all runs successfully attain a stable squatting cycle after 7.25 h.

Model-plant mismatch $\mathcal{E}^{\mathrm{u}}$ and $\mathcal{E}^{\mathrm{l}}$ is minimized to about 0.5 and 0.8 for reaching upper and lower setpoints, respectively. As it was expected, minimization of model-plant mismatch leads to minimization of the nominal controller objective (4.2a) shown by plots $\mathcal{L}^{\mathrm{u}}$ and $\mathcal{L}^{\mathrm{l}}$. The smallest final costs are incurred by the 3rd run because it was stuck the least due to temperature fluctuations, while the largest costs are incurred by the 1st run which was stuck the most.

RMSE after learning is calculated in Table 4.1. RMSE of MPML trajectories is much higher than for the realistic model, and it also exhibits more variability. Unfortunately, it is not possible to obtain the reference RMSE value of the real robot.

Figure 4.7 shows the MPML knee control signal and the RL compensation component of it. For reference, NMPC control on the idealized model is also shown. MPML controls are very oscillatory comparing to NMPC. Nonetheless, the robot neither fell down, nor were its motors damaged, which is a significant result, cf. Schuitema (2012). As is expected with Coulomb friction compensation,

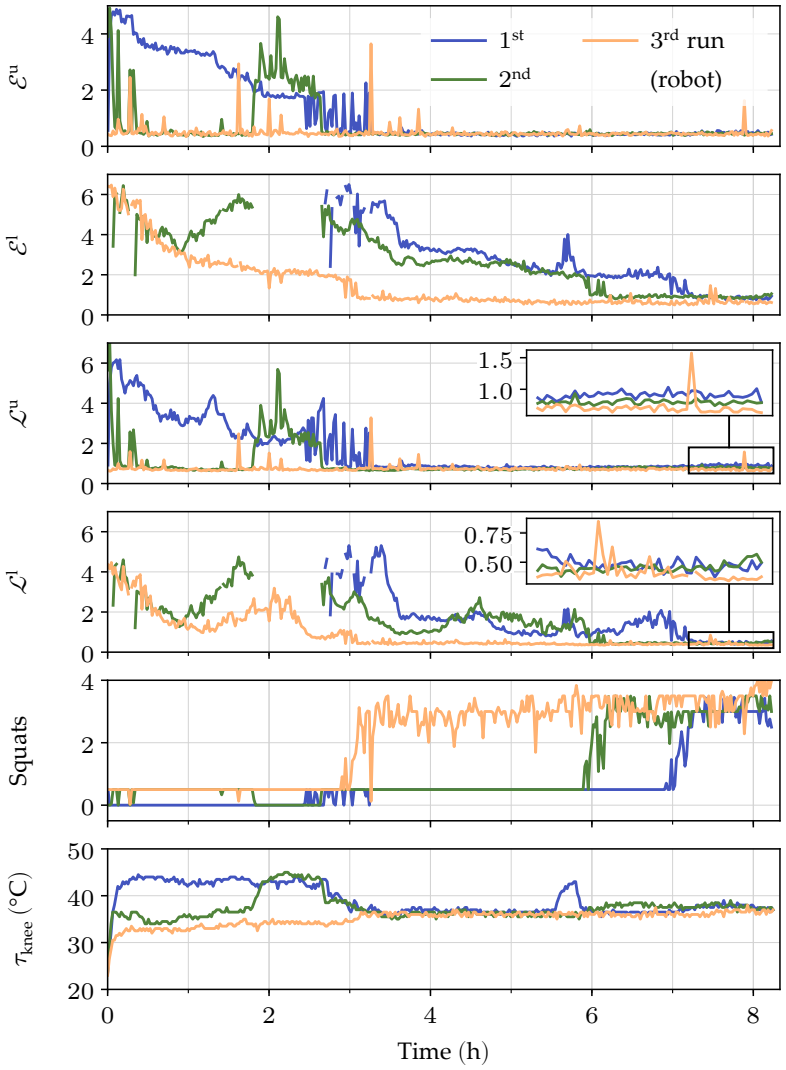


Figure 4.6: Learning progress of three independent runs. (*upper two*) Model-plant mismatch, (*middle two*) nominal controller objective, number of squats and the mean temperature of knee motors obtained during three real learning experiments. Measurements of $\mathcal{E}^{\mathrm{l}}$ and $\mathcal{L}^{\mathrm{l}}$ are missing when the upper switching point is not reached.

RL learned to apply positive and negative controls for the upward and downward motions, respectively.

4.6 Discussion

Even though the final CAL policy is optimal with respect to the real system, the policy is suboptimal with respect to the objective of the nominal controller (4.2a). This result can be explained by the fact that RL and NMPC objectives are not exactly same. While NMPC optimizes the undiscounted cost up to horizon T, RL optimizes γ-discounted reward on the infinite horizon. All in all, RL views the system and the nominal controller as a hybrid entity and the obtained policy becomes optimal with respect to the RL objective. This observation also explains the inability of CAL$^{\gamma=0.97}$ to reach the optimal performance on the realistic model, even though it can significantly improve the performance of the nominal controller. The longer prediction horizon used by CAL$^{\gamma=0.99}$ attains a better performance.

Another problem of CAL is the slow convergence which is probably caused by the fact that the reward constructed from the quadratic objective function of the nominal controller results in small gradients, see Chapter 3 for details. This hypothesis is supported by the fact that DPG with a quadratic cost function learns the task extremely slowly. To mitigate this, the RL cost function can be modified. The downside of it can be the difficulty of predicting the outcome of such modification, e.g. robot velocity may change drastically.

The MPML approach is free from these complications. However, it should be emphasized that MPML optimizes policy with respect to the internal model, that is RL forces the system to behave like the idealized model. In principle, a large mismatch may pose a problem because the obtained policy will be less optimal with respect to the real system and control constraints may prevent the necessary compensation to be applied. However, in our experiments, this is not a problem. Minimization of the model-plant mismatch $\mathcal{E}$ closely follows minimization of the nominal controller cost $\mathcal{L}$. MPML successfully learns to compensate the unknown Coulomb friction as well as its dependency on motor temperature and torque.

The MPML approach obtains the lowest RMSE. This does not come as a surprise, as MPML directly minimizes the mismatch by the specifically constructed reward function. CAL also minimizes RMSE, even though its primary goal is not defined in terms of such minimization. Arguably, in order for NMPC to successfully complete the task, the realistic model should resemble the idealized one which is achieved by learning with RL in both approaches.

For both proposed approaches, a little deviation caused by RL exploration leads to an immediate setback reaction from NMPC. Our simulated experiment reveals a wide range of admissible exploration noise σ for which the number of NMPC constraint violations and robot falls is zero. This result demonstrates the role of NMPC which provides safety barriers to constrain RL exploratory actions near dangerous state space regions. However, there are disadvantages. First, the formulated task demands deliberate control learning which is difficult in the presence of Coulomb friction. If the robot starts moving after a slight overshoot caused by RL explo-

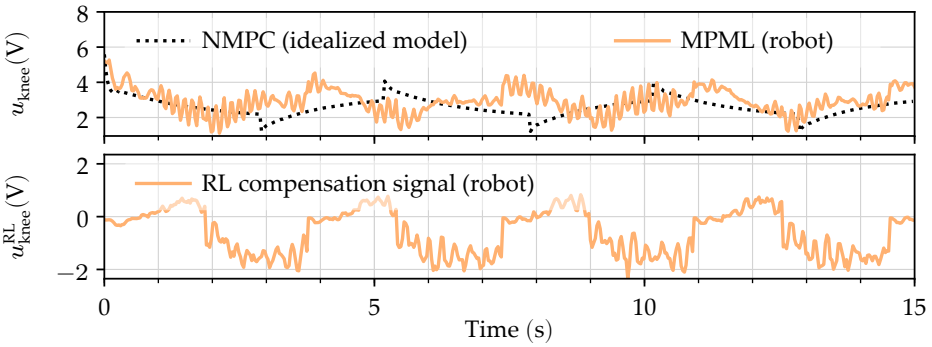


Figure 4.7: (*top*) Knee control signal of NMPC applied to the idealized model and of MPML applied to the real robot after learning. (*bottom*) Compensation signal learned by MPML.

ration, this immediately causes the decrease of friction (Stribeck effect), and at the next sampling moment, the system displacement appears to be too large. NMPC counteracts, so that resulting trajectories appear to be oscillatory. The other reason of oscillations is due to the large control delay. Given the results, it is hard to assess the role of NMPC counter-reaction in the oscillatory trajectories, but we expect that reduction of sampling time and control delay will reduce oscillations. Second, NMPC can drive the system very close to constraint boundaries. For some systems violation of constraints can be very critical, however, this is not true in our case.

We note that the success of model-plant mismatch compensation depends on the learning capabilities of RL on the hybrid system mentioned above. Whether there is a decrease or increase of computational complexity of that system against the original system remains an open problem.

It is common to compensate for a steady-state error in a task completion by adding an integral term to the objective that is tuned by experimental data. Learning the actual model-plant mismatch with MPML goes far beyond cost tuning. It allows to predict the outcome of executed actions since the learned trajectory is expected to be optimal with respect to the idealized model.

4.7 Conclusion

We proposed two learning approaches to compensate model-plant mismatch. Our simulation results demonstrated the feasibility of both approaches. We implemented the better one on a real robot affected by torque and temperature dependent friction and autonomously learned a squatting task. Trying to achieve a similar performance with the standalone nominal controller would require tedious identification of the law of such a dependency. During learning, the robot did not fall.

Several avenues can be explored in future. First, one may reduce learning time by using different function approximators or RL algorithms. A wide range of recent model-free RL can be utilized in a straightforward way. Second, it is important to validate MPML on more challenging tasks. It is also a relatively straightforward im-

plementation if the nominal controller is already set up, and RL actions are bounded so that the nominal controller can always compensate them.

5

Sample-efficient reinforcement learning via difference models

Ivan Koryakovskiy, Divyam Rastogi[1]

The combination of the nonlinear model predictive control (NMPC) and reinforcement learning (RL) methods studied in the previous chapter is currently not applicable to walking tasks because it requires a different type of the online NMPC controller that can properly handle hybrid dynamics. To the best of our knowledge, such a controller does not exist yet. Therefore, this chapter suggests another way of compensating model-plant mismatch by leveraging only the approximate forward model of Leo. This chapter provides: (1) an iterative RL approach that can scale to systems with highly non-linear and contact-rich dynamics with continuous state and action spaces, and (2) sample complexity analysis. The approach reduces the demand for real samples by only learning the difference in regions of the state space that are essential for completing the task. This chapter concludes the pursuit towards attaining the answer to research question 2.

Shortened version published in: Rastogi, D., Koryakovskiy, I., Kober, J. (2018). Sample-Efficient Reinforcement Learning via Difference Models, *The Third Machine Learning in Planning and Control of Robot Motion Workshop at ICRA 2018*, Brisbane, Australia.

[1] D.Rastogi is currently with Robotics and Biology Lab, Department of Computer Engineering and Microelectronics, TU Berlin, Marchstraße 23, 10587 Berlin, Germany.

5.1 Introduction

Learning with no or little prior knowledge about the environment is becoming more popular among control researchers. The reason is that learning provides a generic toolbox for solving complex high dimensional tasks possibly subject to discontinuities. A good example of such task is bipedal locomotion. The intrinsic vulnerability of bipedal walking platforms limits the amount of possible real experience, thereby making it very expensive to obtain. Therefore, data-driven methods require the development of measures which reduce their sample complexity and speed up the learning. The consequence is that these methods lack generality and hence become applicable to specific types of robots (Morimoto et al., 2004; Mori et al., 2004; Matsubara et al., 2006; Tedrake et al., 2004; Schuitema et al., 2005; Endo et al., 2008).

In the light of the recent success of deep RL methods for control of high dimensional systems (Schulman et al., 2016; Lillicrap et al., 2015; Schulman et al., 2015), we study the applicability of neural network representations to learning on the real bipedal robot Leo (Schuitema et al., 2010). Deep RL methods require more than 10^5 samples for their convergence (Duan et al., 2016). This limits the application of such methods only to simulated systems. Due to the discrepancy between the model and the real system, the *reality gap*, policies that perform optimally in simulation fail to perform adequately in the real world, particularly when applied to high degree of freedom (DoF) systems like bipedal robots. Therefore, a possible solution is to pre-train a policy in a simulator, and then warm-start learning on the real system.

A substantial amount of work can be done to better match the simulated system with the real-world system. This involves improving the contact models, friction and also improving the physical identification of quantities like masses, joint lengths and friction coefficients (Ljung, 1999). Apart from being a tedious and time-consuming process, it also has a drawback of complicating and over-describing the model, making the simulation slow and finding a feasible policy more challenging.

In this chapter, our main contribution is the proposal of learning a difference model using a deep neural network (DNN). We assume that an idealized simulator of Leo is provided and that the real system resembles the model to some extent. We expect that the accurate representation of the difference between systems can mitigate the reality gap. Since the reality gap only matters in regions of the state space explored by the policy, the difference model only needs to be accurate in those parts of the state space. We compare the performance of our method against warm-started and cold-started (from scratch) learning on two simulated examples: 1-DoF inverted pendulum and 7-DoF bipedal robot Leo.

5.2 Related work

RL is an established technique and has already been successfully applied to learn a variety of robotic tasks (Kober et al., 2013). However, a limited number of approaches tackle the bipedal walking problem. Most of these approaches try to incorporate a certain level of prior knowledge into training. For example, they do

this by using a pre-structured policy (Mori et al., 2004; Matsubara et al., 2006), or by reducing the dimensionality of the controlled system using the symmetry of the robot (Endo et al., 2008; Schuitema et al., 2010).

More recent model-free approaches focus on generic solutions achieved by training neural network representations (Heess et al., 2016; Schulman et al., 2016). The methods demonstrate remarkable performance on highly challenging 3D locomotion tasks but require a large number of training samples and are therefore limited to simulated environments.

There has been a large body of work which belong to the class of model-based RL methods which tackle the issue of high sample complexity. In these methods, the policy training is either interleaved with a forward model learning (Deisenroth and Rasmussen, 2011; Hester et al., 2012) or it is done concurrently (Caarls and Schuitema, 2016). These methods start with no knowledge about the model and aim for learning it by interaction with the real system.

However, usually an idealized model of the physical system is known but various uncertainties do not allow achieving optimal performance of the policy trained on this idealized model. To correct the discrepancy between the system and the model, Christiano et al. (2016), Gamboa Higuera et al. (2017) and Hanna and Stone (2017) learn the inverse model directly from the physical system. These methods assume that the inverse model can connect successive states prescribed by the nominal controller. The approach taken by Abbeel et al. (2006) is free from this assumption because it learns an additive component of the forward model. The approach allows to compensate both parametric and structural uncertainties, but cannot scale to systems with discontinuities. A similar approach was proposed by Ha and Yamane (2015). Unfortunately, it uses Gaussian processes for modeling the reality gap which limits its application to high DoF systems (Lillicrap et al., 2015). The approach proposed by Farchy et al. (2013) scales to high DoF systems and environmental discontinuities but requires expert guidance to select parameter sets to investigate after each model update. The approach proposed in Chapter 4 works in combination with NMPC which allows it to provide safety barriers on RL exploration depending on the severity of the mismatch. The approach can scale to high DoF systems with parametric and structural uncertainties but requires an additional controller at hand.

In this chapter, our contribution is twofold. First, we propose an iterative model learning approach which scales to high DoF contact-rich systems and does not require human supervision. We assume that an idealized model of a system is provided and that the real system resembles the model to some extent. We approximate the difference with a DNN which allows us to apply our method to high dimensional continuous systems. We expect that the accurate representation of the difference between systems can mitigate the reality gap. Second, we provide a detailed sample complexity analysis by comparing the performance of our method against warm-started and cold-started (from scratch) learning on two simulated examples: 1-DoF inverted pendulum and 7-DoF bipedal robot Leo. To our knowledge, such analysis is currently missing in the literature. Since the reality gap only matters in regions of the state space explored by the policy, the difference model only needs

to be accurate in those parts of the state space. This helps us to reduce sample complexity further since the model only needs samples in regions which are important for the task.

5.3 Reinforcement learning

We consider a standard reinforcement learning problem formulation. The learning is formalized by Markov decision process which is the tuple $\langle \mathcal{X}, \mathcal{U}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{X}$ is a set of n_x-dimensional states $\boldsymbol{x} \in \mathbb{R}^{n_x}$, $\mathcal{U}$ is a set of n_u-dimensional actions $\boldsymbol{u} \in \mathbb{R}^{n_u}$, $\mathcal{P} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ is a transition function which defines a probability of ending up in state $\boldsymbol{x}_{k+1}$ after taking action $\boldsymbol{u}_k$ in state $\boldsymbol{x}_k$. Here k denotes discrete time steps of the system. Reward function $\mathcal{R} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ gives a scalar reward $r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1})$ for particular transitions between states.

The goal of the learning agent is to maximize the discounted expected return

$$G_k^\gamma = \mathbb{E}\left\{ \sum_{i=0}^{\infty} \gamma^i r(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}, \boldsymbol{x}_{k+i+1}) \right\} \tag{5.1}$$

where $\gamma \in [0; 1)$ is the discount factor which ensures integrability of the infinite sum.

The control policy, or the actor, $\pi : \mathcal{X} \to \mathcal{U}$ is a deterministic function which selects action $\boldsymbol{u}_k$ in state $\boldsymbol{x}_k$. Exploration is achieved by adding random noise sampled from some process $\mathcal{N}$. When solving continuous control problems, it is convenient to evaluate the quality of the policy by the action-value function $Q^\pi : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, or critic. It describes the expected return after taking an action $\boldsymbol{u}_k$ in state $\boldsymbol{x}_k$ and thereafter following policy π.

We use a particular realization of RL, the off-policy Deep Deterministic Policy Gradient (DDPG) algorithm of Lillicrap et al. (2015) with DNN function approximation and compatible features, chosen for its ability to optimize continuous control policies for high DoF systems. The value function and policy are parametrized by the network weights denoted as $\boldsymbol{\theta}^{\mathrm{c}}$ and $\boldsymbol{\theta}^{\mathrm{a}}$.

5.4 Proposed method

5.4.1 Notation

In this chapter, we work only with simulated models. Therefore, we create two models called *idealized* and *true*. Our method minimizes the gap between these models by learning the *difference model*. Schematically, the proposed method is shown in Figure 5.1.

The *idealized model*

$$f^{\mathrm{idl}}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{x}_{k+1}^{\mathrm{idl}}$$

is a forward dynamics (usually first principles) model based on a certain idealized configuration of the robot. We denote the policy trained on this model as π^{idl}.

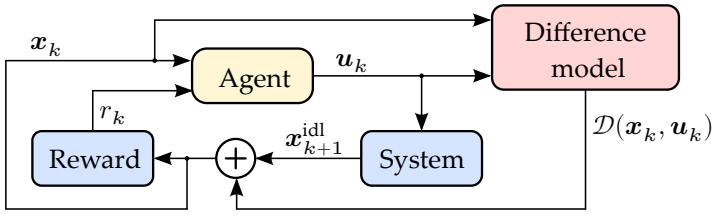


Figure 5.1: Proposed framework.

The *true model*

$$f^{\text{true}}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{x}_{k+1}^{\text{true}}$$

is a forward dynamics model based on the real configuration of the robot. The corresponding policy trained on this model is denoted as π^{true}.

The *difference model*

$$\mathcal{D}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{x}_{k+1}^{\text{true}} - \boldsymbol{x}_{k+1}^{\text{idl}} \tag{5.2}$$

predicts the difference in the transition states given an initial state and action. The policy trained using the difference model is denoted as $\pi^{\mathcal{D}}$.

5.4.2 Algorithm

We begin with the algorithm overview presented in Figure 5.2. The algorithm starts by pre-training the initial policy π^{idl} and the corresponding value function using the idealized model of the robot. The policy behaves sub-optimally when applied to the true model due to the reality gap. To eliminate the gap, we propose an iterative approach which consists of three distinctive steps. In the first step, we collect a number of real samples through interaction with the true model and save them in buffer $\mathbb{B}$. In the second step, we use the whole buffer to learn the difference model represented by the DNN. The difference model predicts the difference in states between the idealized model and the true model. In the third step, DDPG is used to learn a new policy $\pi^{\mathcal{D}}$ with this difference model. The newly obtained policy is likely to perform better on the true model. However, if the initial reality gap is large, the new policy may still not perform satisfactory. Thus, we iterate the process of updating the difference model and the policy until a certain number of iterations is reached. To keep the successive DDPG training time short, we bootstrap parameters of the policy and the value function from the ones used in the previous iteration.

The complete algorithm is given in Algorithm 1. In line 5, N trajectories $\{\zeta^n\}_{n=1}^N$ are collected by executing the obtained policy on the true model. Each trajectory $\zeta^n = (\boldsymbol{x}_1^n, \boldsymbol{u}_1^n, \dots, \boldsymbol{x}_K^n, \boldsymbol{u}_K^n, \boldsymbol{x}_{K+1}^n)$ records the sequence of states and actions for some predefined time T or until the system fails the task. While obtaining the trajectories, a certain amount of random exploratory noise $\boldsymbol{n} \sim \mathcal{N}$ is added to the policy to ensure a good exploration of the state space around the previously learned policy. Obtained trajectories are saved into the buffer $\mathbb{B} = \{(\boldsymbol{x}_{1:K}^{\text{true},n}, \boldsymbol{u}_{1:K}^{\text{true},n}, \boldsymbol{x}_{2:K+1}^{\text{true},n})\}_{n=1}^N$.

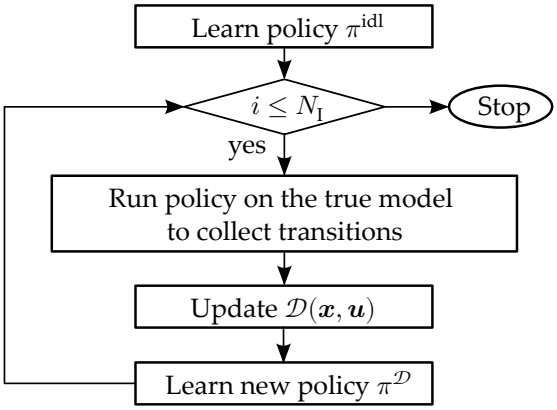


Figure 5.2: Algorithm overview.

In lines 6 to 9, the $\{(\boldsymbol{x}_{1:K}^{\text{true},n}, \boldsymbol{u}_{1:K}^{\text{true},n})\}_{n=1}^{N}$ pairs are applied to the idealized model in order to calculate the difference (5.2). In line 10, supervised learning (Duchi et al., 2011) is used for training the difference model $\mathcal{D}(\boldsymbol{x}_k, \boldsymbol{u}_k)$. Finally, DDPG learns a new policy in line 12.

5.4.3 Training data

The data used for training the difference model is obtained by running the previously obtained policy on the true model several times. Let this policy be denoted by $\pi(\boldsymbol{x}; \boldsymbol{\theta}_{i-1}^{\text{a}})$. The action to be taken is calculated as

$$\boldsymbol{u} = \pi(\boldsymbol{x}; \boldsymbol{\theta}_{i-1}^{\text{a}}) + \boldsymbol{n}.$$

The particular advantage of using policy from the previous iteration is that it progressively matches the distribution of training transitions to those required for the successful completion of the task. Since DDPG is used to learn a new policy after every update of the difference model, the replay buffer from the first update of the policy is saved and used in every following update. This helps to reduce the number of iterations required to find a good policy at each step and increases the diversity of samples.

5.5 Experiment details

5.5.1 Inverted pendulum

Figure 5.3 shows the inverted pendulum of mass $m = 0.055\,\text{kg}$ which rotates around the fixed point O located $l = 0.042\,\text{m}$ away from the center of mass of the pendulum. The pendulum state $\boldsymbol{x} = [\phi, \dot{\phi}]^\top$ is composed of pendulum angle ϕ and angular velocity $\dot{\phi}$. The control input $u \in [-3\,\text{V}, 3\,\text{V}]$ is the voltage applied to the motor located at O. The voltage is bounded to prevent the pendulum from performing a swing-up in one go. The pendulum is initialized in state $\boldsymbol{x}_0 = [\pi, 0]^\top$,

Algorithm 1 Proposed algorithm

1: Initialize training data buffer $\mathbb{B} = \{\emptyset\}$ and the difference model $\mathcal{D}(\boldsymbol{x}, \boldsymbol{u}) = 0$
2: Learn the Q-function and the policy parametrized by $\boldsymbol{\theta}_0^{\mathrm{c}}$ and $\boldsymbol{\theta}_0^{\mathrm{a}}$ using DDPG on the idealized model
3: **for** $i = 1, N_{\mathrm{I}}$ **do**
4: Initialize random process $\mathcal{N}$
5: Execute policy $\pi^{\mathcal{D}}(\boldsymbol{x}; \boldsymbol{\theta}_{i-1}^{\mathrm{a}}) + \mathcal{N}$ on the true model to obtain transitions $(\boldsymbol{x}_k^{\text{true}}, \boldsymbol{u}_k^{\text{true}}, \boldsymbol{x}_{k+1}^{\text{true}})$
6: **for** each $(\boldsymbol{x}_k^{\text{true}}, \boldsymbol{u}_k^{\text{true}})$ pair **do**
7: Calculate $\boldsymbol{x}_{k+1}^{\text{idl}} = f^{\text{idl}}(\boldsymbol{x}_k^{\text{true}}, \boldsymbol{u}_k^{\text{true}})$
8: Add training data to the buffer $\mathbb{B} \leftarrow \mathbb{B} \bigcup \left(\boldsymbol{x}_k^{\text{true}}, \boldsymbol{u}_k^{\text{true}}, \boldsymbol{x}_{k+1}^{\text{true}} - \boldsymbol{x}_{k+1}^{\text{idl}}\right)$
9: **end for**
10: Update $\mathcal{D}(\boldsymbol{x}, \boldsymbol{u})$ using $\mathbb{B}$
11: Initialize critic and actor networks with weights $\boldsymbol{\theta}_{i-1}^{\mathrm{c}}$ and $\boldsymbol{\theta}_{i-1}^{\mathrm{a}}$
12: Learn a new Q-function and a policy parametrized by $\boldsymbol{\theta}_i^{\mathrm{c}}$ and $\boldsymbol{\theta}_i^{\mathrm{a}}$ where $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_{k+1}^{\text{idl}} + \mathcal{D}(\boldsymbol{x}_k, \boldsymbol{u}_k)$
13: **end for**

and the agent has to learn to swing up the pendulum and balance it until the end of the 20 s episode. The reward function is given by

$$r = -5\phi^2 - 0.1\dot{\phi}^2 - 0.01u^2.$$

Sampling period is $T_s = 0.05\,\mathrm{s}$.

The true model has a higher pendulum mass shown in Table 5.1.

5.5.2 Bipedal walking robot Leo

Leo (Schuitema et al., 2010) is a 2D bipedal robot developed by the Delft BioRobotics Lab shown in Figure 5.4. The robot is attached to a boom which prevents it from lateral falls. Leo is modeled using Rigid Body Dynamics Library (RBDL) (Felis, 2017b) with the boom mass added to the torso.

Leo has 7 actuators, two for each hip, knee, and ankle, and the last one for the shoulder. All joints and the torso-to-boom connection are equipped with encoders which provide real-time measurements. The learning state space of Leo comprises of 18 dimensions which consist of the angles ϕ and the angular velocities $\dot{\phi}$ of all but shoulder joints, and the torso linear positions and velocities. The action space of Leo consist of voltages applied to each actuator except the shoulder which is actuated using a proportional-derivative (PD) controller.

The reward function awards the agent $300\,\mathrm{m}^{-1}$ for every meter of forward movement of the robot. The agent is penalized by a -1.5 additional reward for every time step and by a $-2\,\mathrm{J}^{-1}$ reward for every Joule of electrical work done. Premature termination of the 20 s episode due to a fall is punished by the negative reward of -75. Sampling period is $T_s = 0.03\,\mathrm{s}$.

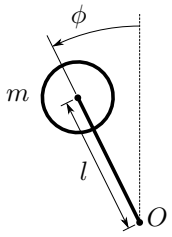


Figure 5.3: Inverted pendulum.

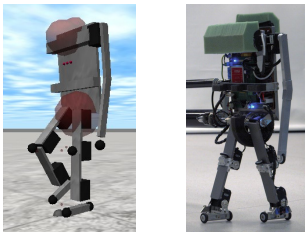


Figure 5.4: Leo: the model and the real robot.

The true Leo model has higher torso mass and viscous friction added to all actuators. These differences affect the performance of the idealized policy when applied to the true model, occasionally leading to oscillations and eventual fall of the robot. The change in parameters is shown in Table 5.1.

5.5.3 Training data and parameters

All parameters are kept the same for both systems. The DDPG critic and actor networks consist of 2 hidden layers with 300 and 400 neurons, respectively. Learning rates of 0.001 and 0.0001 are chosen for the critic and the actor. Additional parameters include a target network update rate of 0.001 and the discount rate $\gamma = 0.99$. Exploration is achieved by Ornstein-Uhlenbeck (OU) noise with parameters $\sigma = 0.12$ and $\theta = 0.15$ used during warm-start learning and learning with the difference model. For cold-start learning noise parameters are slightly higher.

The difference model consists of 3 hidden layers with 400 neurons in each layer. The activation function for each of the hidden layers is ReLU while the output layer has a linear activation function. To reduce over-fitting to the training data, we use dropout (Srivastava et al., 2014) with probability 0.3. Dropout is only applied to the hidden layers and is the same for all layers. Before each model update, 3750 datapoints are collected by running the previously obtained policy on the true model which are saved to a buffer that has not limits on size. We train the network using AdaGrad (Duchi et al., 2011) which is a variant of stochastic gradient descent. The advantage of using AdaGrad over normal stochastic gradient is the ability to adapt the learning rates based on the training data. This implies that the initial learning rate does not have much impact on the performance of the algorithm.

5.5.4 Evaluation measures

For quantitative assessment, we evaluate performance of the proposed method in terms of the undiscounted return (5.1). Additionally, for the Leo robot we evaluate the walked distance S and the motor work E normalized to the walked distance given by

$$E = \frac{T_s}{S} \sum_{j=1}^{n_u} U_j \frac{U_j - K_\tau \dot{\phi}_j}{R}$$

Table 5.1: Difference in physical parameters between idealized and true model.

Parameter	Idealized model	True model
Inverted pendulum		
Pendulum mass m (kg)	0.055	0.090
Robot Leo		
Torso mass (kg)	0.942	1.250
Viscous friction coefficient ($\mathrm{N\,s\,m^{-2}}$)	0.000	0.030

where U_j is the voltage applied to the motor j, K_τ is the motor's torque constant, and R is the winding resistance.

We evaluate the quality of the difference model by calculating the mean squared error (MSE) between the transitions obtained on the true model and the idealized model enhanced by the difference model

$$\frac{1}{|\mathbb{B}|}\sum_{k=1}^{|\mathbb{B}|}\left\|f^{\text{true}}(\boldsymbol{x}_k,\boldsymbol{u}_k)-(f^{\text{idl}}(\boldsymbol{x}_k,\boldsymbol{u}_k)+\mathcal{D}(\boldsymbol{x}_k,\boldsymbol{u}_k))\right\|_2^2.$$

For qualitative assessment, we also show system trajectories obtained by each method.

5.6 Results

5.6.1 Inverted pendulum

Figures 5.5 and 5.6 show the return and final trajectories obtained by the policies evaluated on the true model. Policy trained on the idealized model is not capable of balancing the pendulum and therefore obtains the lowest return of -15508. On the contrary, the proposed method of learning the difference model can cross the reality gap. After one model update, the policy reaches the return of -660.0 which is similar to the one obtained by the policy π^{true} trained on the true model. This number of updates corresponds to 3.125 min of interaction with the true model. The cold-started method requires around 75 min to converge.

Comparing the final state and control trajectories in Figure 5.6, we notice the similarity between policies π^{true} and $\pi^{\mathcal{D}}$. Although policies are not exactly same, the difference model captures the mismatch which is crucial for the successful pendulum balancing.

Table 5.2 compares results of learning on the true model obtained by DDPG learning from scratch (cold start), initializing the value function and the policy by learning on the idealized model (warm start) and by the proposed method. Given the budget of 15000 true model samples, the proposed method obtains the highest return. Alternatively, it requires 6 times fewer samples than the warm-started learning to reach the target return of -1000.0.

Finally, we evaluate the quality of the difference model by calculating MSE. Re-

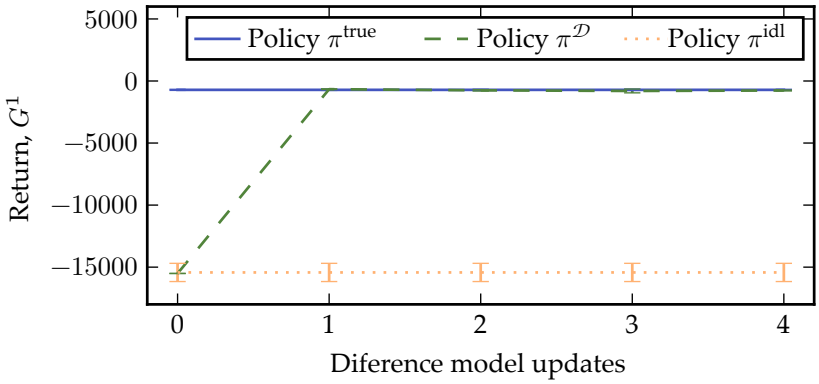


Figure 5.5: Comparison of results by evaluating the performance of different policies on the true model for the inverted pendulum. The results with the difference model are averaged over 5 runs and plotted with 95% confidence interval.

Table 5.2: Pendulum sample complexity and performance comparison for the proposed method versus cold and warm starts.

Method	Sample budget	Return, G^1
Fixed budget		
Cold start	15000	-20507.6 ± 2801.9
Warm start	15000	-14087.2 ± 235.6
Difference model	15000	-657.2 ± 22.9
Fixed return		
Cold start	82164 ± 7498	-1000.0
Warm start	23383 ± 2357	-1000.0
Difference model	3750	-1000.0

sults in Table 5.3 show that the error between the true and the idealized model enhanced by the difference model is negligible compared to the case when the difference model is not used.

5.6.2 Robot Leo

The results of learning the policy with the difference model are presented in Figure 5.7. Here all policies are evaluated on the true model. Initial updates of the difference model show a high standard error, which is reduced in later updates. After 7 updates, the proposed method reaches the mean return of 2297.46 which is within 10 % of the return obtained by learning directly from the true model. This number of updates corresponds to about 13 min of interaction with the true system, while the cold-started method requires 200 min to converge.

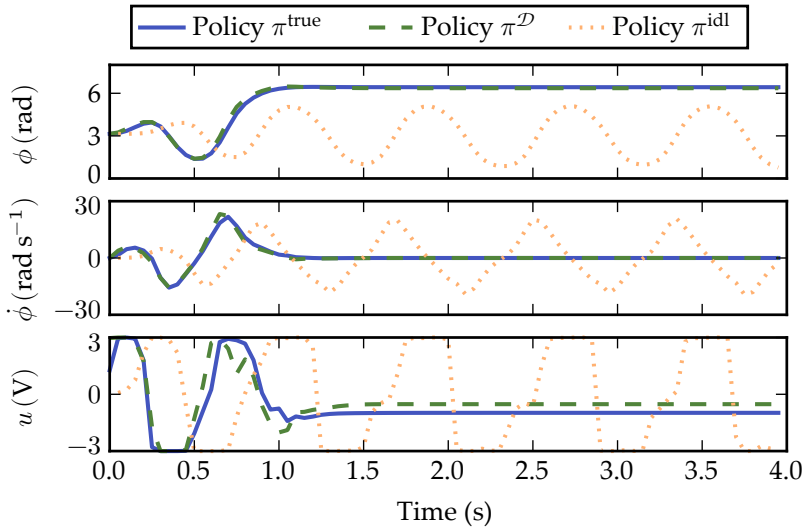


Figure 5.6: State trajectories and control input for policies learned on different models and evaluated on the true model.

Table 5.3: MSE without the difference model and with the difference model.

System	Without the model	With the model	Decrease (%)
Inverted pendulum	5.9276	0.0118	99.80
Robot Leo	1.3734	0.8064	41.28

The corresponding final performance of policies evaluated on the true model is shown in Table 5.4. In both benchmarks, the policy trained on the idealized model performs the worst. Enhancing the idealized model by learning the difference model improves the performance. In particular, this policy in the mean provides 38.3 % improvement in the walked distance (see Figure 5.8) and 4.8 % reduction in the energy consumption compared to the policy π^{idl}.

Figure 5.9 shows gait examples the robot learns with the policies. Visually, all trajectories display some irregularity of the walking cycle. As expected, the π^{idl} policy performs worst, and its gait exhibits small step sizes and a tendency to lift the swing knee very high. This results in a slow walking gait. The difference model improves the idealized model. Therefore, the $\pi^{\mathcal{D}}$ policy leads to a higher walking speed due to the reduction of the swing knee lifts.

Table 5.5 compares results of learning on the true model from scratch (cold start), initializing the value function and the policy by learning on the idealized model (warm start) and by the proposed method. Compared to the inverted pendulum, we increase the samples budget to 33750 true model samples. The proposed method obtains the highest return. Alternatively, it requires about 5 times fewer samples

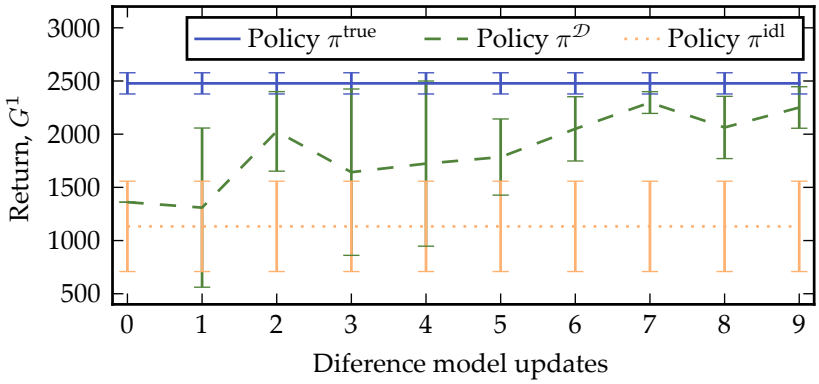


Figure 5.7: Comparison of results by evaluating the performance of different policies on the true model for Leo. The results with the difference model are averaged over 5 runs and plotted with 95% confidence interval.

Table 5.4: The comparison of different policies performance evaluated on the true model.

Policy	Distance walked, S (m)	Motor work, E ($\mathrm{J\,m^{-1}}$)
π^{idl} (idealized)	9.2 ± 2.4	111.4 ± 11.1
$\pi^{\mathcal{D}}$ (difference)	14.9 ± 0.5	106.0 ± 3.8
π^{true} (true)	14.8 ± 0.5	98.2 ± 6.7

than the warm-started learning to reach the target return of 2300. Compared to the cold-started learning, the proposed method requires 20 times fewer samples to reach the target.

The quality of the learned difference model is evaluated in Table 5.3. It can be seen that with the difference model MSE decreases by 41.28 % which is less compared to the inverted pendulum.

5.7 Discussion

The presented results demonstrate that learning the difference model for the inverted pendulum is easier than for Leo. This is an expected result because the dimensionality of the pendulum is much lower, and there are no discontinuities in trajectories. For the pendulum, a single model update is enough to generalize well in the unseen parts of the state space. However, for Leo, this is not the case. The mismatch between the true and idealized models affects the whole state space of the robot, and we need more than one model update to compensate for it. The very high standard error for the initial updates of the difference model is due to the occasional failures of $\pi^{\mathcal{D}}$ on the true model. A possible reason for this could be that the $\pi^{\mathcal{D}}$ policy explores regions of the state space where the difference model is not yet

Figure 5.8: This figure illustrates the performance of π^{idl} (*yellow*), $\pi^{\mathcal{D}}$ (*green*), and π^{true} (*blue*) on the true model. The policy learned with the difference model achieves almost the same walking distance while only requiring 6.5% of interaction time compared to the cold-started method. Also see supplementary video.

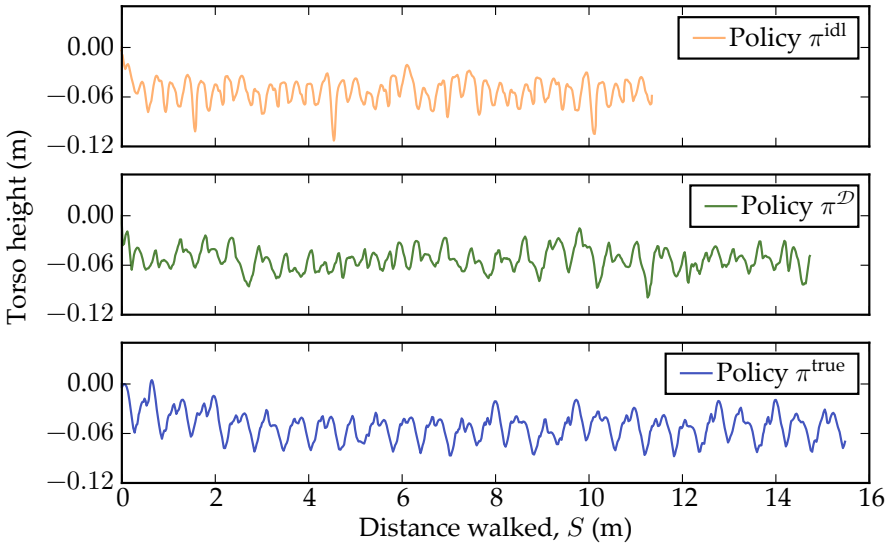


Figure 5.9: Torso trajectory for policies π^{true}, π^{idl} and $\pi^{\mathcal{D}}$ evaluated on the true model. Each walking trial lasts for 20 s.

accurate enough. This can happen because there are no constraints on the policy update, and the policy is free to divert into the regions of the state space where little or no data is available.

To further investigate the reason for the failure, we show in Figure 5.10a the distributions of states with the policy evaluated on the difference model and the true model. Initially, these distributions are entirely different, which implies that the difference model makes errors in predictions. Therefore, when the same policy is applied to the true model, the policy diverts into regions with little training data available, see hip trajectories in Figure 5.10a. The problem is further aggravated by the fact that the torso is heavier than the legs. This reduces the stability region for the robot and implies that there is a very small difference between successful and unsuccessful policies. However, the difference model gets more accurate with the number of updates, as new samples are collected in previously unexplored regions.

Table 5.5: Leo sample complexity and performance comparison for the proposed method versus cold and warm starts.

Method	Sample budget	Return, G^1
Fixed budget		
Cold start	33750	-42.9 ± 12.1
Warm start	33750	1782.1 ± 247.4
Difference model	33750	2434.1 ± 111.1
Fixed return		
Cold start	309766 ± 35282	2300.0
Warm start	84326 ± 6287	2300.0
Difference model	15750 ± 7648	2300.0

The corrected model predictions reduce the probability of the learned policy failing on the true model, thus reducing the standard error of the return. This hypothesis is supported by Figure 5.10b, where the distribution of data for the difference model and the true model is roughly the same. The resulting hip trajectories are slightly misaligned in time which implies that the walking speed is different, but the overall pattern looks similar.

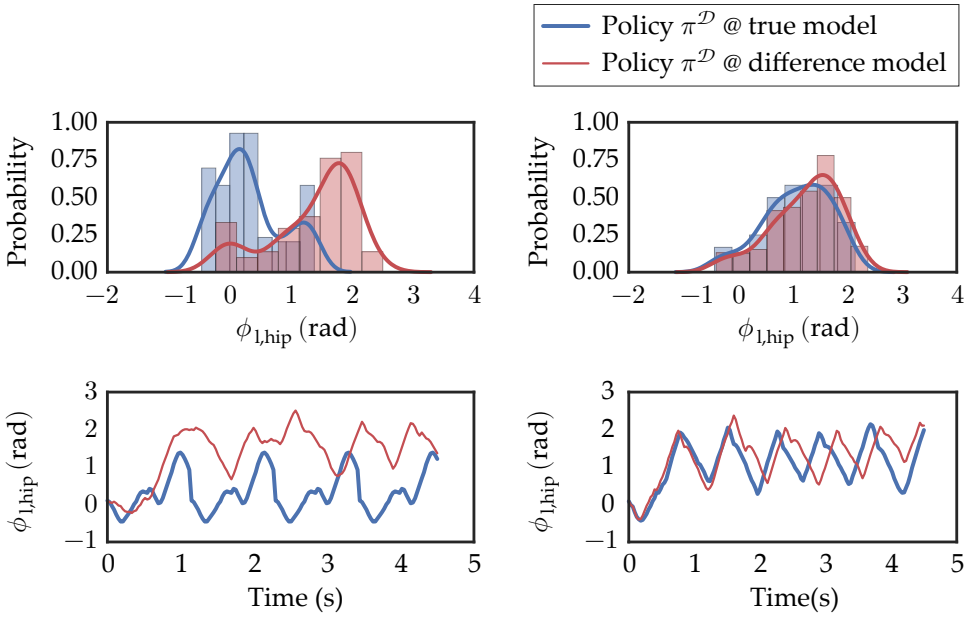
Ideally, the difference model should eliminate the error between the $\boldsymbol{x}^{\text{true}}$ and $\boldsymbol{x}^{\text{idl}} + \mathcal{D}(\boldsymbol{x}_k, \boldsymbol{u}_k)$. Experiments with the inverted pendulum demonstrate that it is possible to greatly reduce MSE. The remaining inaccuracies are due to function approximations and regularization. In Leo, these inaccuracies are aggravated by the larger state-action space and discontinuities.

Finally, we note that learning with the difference model allows to reduce exploration noise on the true model compared to the cold-start learning. This is an important property for learning on real robots as lower noise can potentially reduce system damage.

5.8 Conclusion

In this chapter, we develop a method of reducing the sample complexity of learning on a real robot by iterative updates of the difference model. We compare the performance of our method with cold-started and warm-started learning on two simulated platforms: 1-DoF inverted pendulum and 7-DoF robot Leo. The proposed method achieves significantly higher returns given the same exploration budget on the true model, or it achieves the same return but with a much smaller number of true model samples.

Sample diversity is a known bottleneck of neural networks, which also holds true for learning the difference model. However, our method is successful due to the fact that the difference model needs to be accurate only in the region of the state space which is essential for walking.



(a) Update 1 (b) Update 9

Figure 5.10: Distributions of left hip angles after (a) the first and (b) the last update of the difference model is shown in *top* plots. Note that policy $\pi^{\mathcal{D}}$ is evaluated on the true model (*blue*) and on the idealized model enhanced by the difference model (*red*). Histogram height is scaled to fit the ordinate limit. (*bottom*) Trajectories of the left hip.

The proposed method is generic and can be utilized for a wide variety of robotic systems. A next possible step is to learn the difference model on the real robot Leo.

6

Multitask reinforcement learning for safer acquisition of locomotion skills

Inspired by nature, this chapter revisits the reduction of robot falls in the case when the approximate model of Leo is not available, thereby providing the answer to research question 3. The chapter begins by drawing attention to the records of children learning to walk and clinical rehabilitation studies, which indicate that people tend to learn balancing before walking. Applied to robots, this suggests that the particular arrangement of learning tasks can reduce the number of falls compared to the cases when the final task is learned directly. This hypothesis is verified in a set of experiments performed on four robot models and two dynamic simulators. Provided analysis reveals that (1) learning the balancing task is quicker and safer than learning the walking task, and (2) the acquired balancing skill averts unsafe exploration actions during the walking task. For the model of the real robot Leo, scheduling several tasks results in a 31.8 % reduction in falls compared to the direct learning. Apart from learning the control policy, the proposed curriculum learning strategy can also learn the model of Leo. This model can be combined with the approaches presented in Chapters 4 and 5 to further reduce the risks of hardware damage.

Prepared for journal submission

6.1 Introduction

Motivation. The fundamental goal of reinforcement learning (RL) in the field of robotics is an algorithm that learns to perform a wide range of tasks with the least human involvement. The acquisition of a new skill that solves a task should be reasonably quick, safe for the robot and people, and the learned skill should be robust to perturbations and environmental uncertainties. Modern RL algorithms require more than 10^5 samples for their convergence (Duan et al., 2016), and the number increases with system complexity. Therefore, ensuring safety during such a long learning time is the highest priority when one is interested in model-free learning on real robots.

By safety, we mean the prevention of actions that cause damage to the robot. Ensuring safety is particularly important for legged robots, for which possibly unstable locomotion phases can lead to severe damage due to unexpected robot falls. The falls result in impact forces applied to the limbs and gearboxes, and some robots cannot withstand even a single fall. In RL, to prevent the robots from falling, it is common to punish contacts of the robot body parts other than feet with the ground by means of negative rewards. However, to learn recognizing such situations, they need to be repeated multiple times in different robot configurations. Moreover, in the real world, it is hard to predict the damage severity of the falling robot because it largely depends on the structure of the environment. Random exploration exacerbates the problem and can lead to a very large number of falls. Therefore, reducing robot falls is an important objective.

To reduce the number of robot falls during learning, we draw inspiration from infants learning to walk and post-stroke individuals who re-establish their impaired locomotion skills. In the following, we review RL approaches towards safer learning, curriculum learning in the context of RL, and the neuromechanical studies of infants and post-stroke individuals.

Reinforcement learning approaches towards safer learning. A small number of approaches consider damage reduction during model-free learning. Existing techniques include modification of state or action value functions (Mihatsch and Neuneier, 2002; Shen et al., 2014), or the modification of the exploration process based on some risk measure (Gehring and Precup, 2013; Meijdam et al., 2013; Koryakovskiy et al., 2017). Unfortunately, these approaches are usually sensitive to hyperparameters and may have undesirable effects such as the distortion of a long-term utility and overly pessimistic policies (Garcia and Fernandez, 2015). To avoid such modifications, Schuitema (2012) proposed to identify dangerous states based on reward values and learn two value functions with different resolutions. This approach requires tuning the parameters of the function approximators and the proper selection of the reward threshold.

Curriculum learning for control. Table 6.1 summarizes applications of curriculum learning to control problems in artificial intelligence. Each approach either improves the learning performance in terms of a learning speed or a quality of the

Table 6.1: The summary of the control-related curriculum learning variations found in literature. We denote everything not related to the actuated system embodiment as *environment*.

Category	Reference articles
Changes in the starting or goal distributions	*Improve learning*: Andrychowicz et al. (2017), Asada et al. (1996), Zaremba and Sutskever (2015)
Changes in the environment	*Improve learning*: Yin et al. (2008), Wu and Tian (2017), Sukhbaatar et al. (2015), Heess et al. (2017) *Improve robustness*: Tesauro (1995), Silver et al. (2017), Bansal et al. (2017),
Changes of tasks (reward functions)	*Improve learning*: Karpathy and van de Panne (2012), Tessler et al. (2017)

final policy, or improves the policy robustness in terms of their ability to counteract stronger opponents or external perturbations.

Improve learning. The ability to learn from sparse or binary rewards is an attractive property of RL because it allows avoiding reward engineering efforts. Curriculum learning was found useful in such situations. Asada et al. (1996) suggested a method called Learning from Easy Missions that reduced the learning time from exponential to almost linear order in the size of the state space. The authors divided the task of shooting a ball into the gates from an arbitrary position into subtasks depending on the distance to the goal. The robot was trained to shoot the ball from the closest positions first, and the transition to further positions was made according to some human-selected performance threshold. A similar approach was recently applied to robotic manipulator tasks (Andrychowicz et al., 2017). Experimental results demonstrated the acquisition of complicated control policies in a simulator and the successful policy transfer to a physical robot. First-person shooter games are another example of environments with sparse rewards, for which Wu and Tian (2017) developed a state-of-the-art learning framework.

Tasks that usually require non-sparse rewards may also benefit from curriculum learning. Yin et al. (2008) and Heess et al. (2017) demonstrated quicker learning of sophisticated locomotion skills by gradually increasing the terrain difficulty. At first, agents solved a basic locomotion task on flat terrains or terrains with easy obstacles such as low hurdles, stairs or short gaps in the ground. Once agents learned a feasible gait, they proceeded to more difficult terrains with very tall hurdles and stairs or very long gaps.

Finally, life-long learning benefits from curriculum learning of multiple related skills. Karpathy and van de Panne (2012) tuned the parameters of several separate skills which they progressively merged at later stages. Similarly, Tessler et al. (2017) developed a framework that solved composite tasks after learning to solve relatively simple sub-tasks.

Improve robustness. While in the referenced articles the curriculum is hand-

crafted[1], self-play provides an automatic and natural way of increasing environment difficulty (Tesauro, 1995). In the self-play setting, one realization of RL competes with another realization of a similar level. In the discrete domain, Silver et al. (2017) trained a Go player that achieved performance beyond human capabilities. In the continuous domains, Bansal et al. (2017) demonstrated learning robust control policies in very simple environments.

The proposed hypothesis in relation to the curriculum learning literature. We hypothesize that there is another benefit of curriculum learning – namely that it can increase the safety of learning by reducing the number of robot falls. Before explaining the proposed approach and contributions, we briefly overview the literature related to the locomotion development of infants and post-stroke individuals.

Neuromechanical studies of infants and post-stroke individuals. People start learning locomotion skills from the early days of their lives and attain desired skills after a considerable amount of practice. A remarkable feature of locomotor development is that walking is not practiced immediately but is learned by changing tasks and gradually increasing their difficulty (Adolph and Robinson, 2013). Long before infants can walk, they spontaneously alternate legs in a cycling pattern when lying on their backs or held in mid-air. Later, infants practice rhythmic crawling in a wide variety of forms as well as upright supported standing, which is later exchanged for upright locomotion forms such as walking on knees, or walking with the support of furniture, or their parents. Finally, from the age of 8 to 17 months infants start to enjoy walking on their own (Berger et al., 2007). However, there is no universal development pattern because in infants some stages can precede the others (Adolph et al., 2011). Nevertheless, individual stages are important because they focus on different aspects of walking (Adolph and Robinson, 2013). For example, crawling builds up leg and arm muscles. Upright walking with support begins with much weight applied to the arms. Later, improved body balance allows infants to face forward and use only one hand for their support. Interestingly, parenting that involves practices like massaging, sitting and standing, bouncing, and gravity resisting sling carrying reduce the time until infants can do their first steps. Some infants can even skip the crawling stage due to these nursing activities (Hopkins and Westra, 1990).

Similar but more methodical approaches are used for rehabilitation of individuals post-stroke with the goal of reducing recovery time (Veerbeek et al., 2014). Commonly practiced tasks during rehabilitation include sitting balance combined with reaching tasks, standing up and sitting down, standing balance, and walking. A physical therapist decides proper recovery strategies for patients, monitors their performance, and adjusts the strategies so that the difficulty of tasks increases with patients' progress (Winstein and Kay, 2015). From the literature, it is difficult to give a definite answer to the question of whether one task influences the success rate of the other tasks. For example, the sitting or standing balance tasks are practiced so that the patient can independently perform daily activities such as eating, reaching and dressing, and it is not known if they reduce the walking recovery time.

[1] Andrychowicz et al. (2017) work is a noticeable exception.

Nevertheless, we hypothesize that from the patients' safety perspective learning to balance before learning to walk is less detrimental than learning to walk immediately.

Proposed experiments and contribution. To validate the proposed hypothesis, we conduct a set of locomotion learning experiments on four physical systems using Rigid Body Dynamics Library (RBDL) (Felis, 2017b) and OpenAI Roboschool simulator[2]. For all systems, falls are recognized by the torso angle or center of mass exceeding some thresholds, or by the contacts of the body parts other than feet with the ground. Figure 6.1 shows the experiments conducted. Although the connection between the balancing and walking skills was discovered earlier in the research of passive dynamic walkers (Mochon and McMahon, 1980; Wisse, 2004), is not obvious that learning to balance should help to learn walking. One possible pitfall can be that the well-trained balancing policy will avoid actions that imbalance the robot, thereby obstructing the walking controller from learning.

Our contribution is twofold. First, using a manually-defined number of time steps for each task, we investigate if 2-task and 3-task curricula reduce the number of falls while keeping the policy performance similar to a direct learning. The upper-body balancing task in the 3-task curriculum is motivated by its relevance to post-stroke gait recovery (Veerbeek et al., 2011). For practical purposes, switching at a manually defined number of time steps may lack generality, e.g. the policy can perform worse when this strategy is applied to a real robot. Therefore, our second contribution is the comparison of the time-based switching strategy to another two strategies that switch the tasks automatically. One of the strategies uses a success rate of balancing to determine the task-switching moment. Another strategy incorporates two more performance indicators and uses trained recurrent neural network (RNN) to recognize the next task to practice.

6.2 Background

6.2.1 Reinforcement learning

The goal of a RL agent is to find a control policy π which maximizes the expected discounted return,

$$G_k^\gamma = \mathbb{E}\left\{\sum_{i=0}^{\infty}\gamma^i r(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}, \boldsymbol{x}_{k+i+1})\right\}, \tag{6.1}$$

where k is a discrete time step, $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ is a continuous n_x-dimensional state of the system, $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ is a continuous n_u-dimensional control action and $r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1})$ is a scalar reward the agent receives upon transition. The discount factor $\gamma \in [0, 1)$ conveys the increasing uncertainty about the future (Sutton and Barto, 1998). Walking and balancing are *continuing* tasks for which $\gamma < 1$ ensures the integrability of the infinite sum.

[2] https://github.com/openai/roboschool

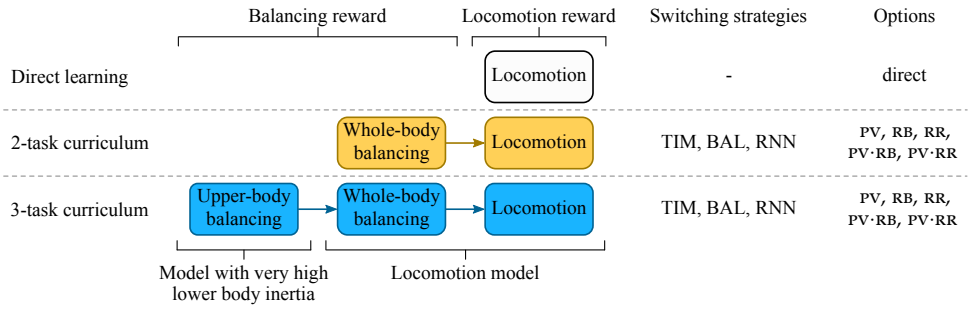


Figure 6.1: Illustration of locomotion learning experiments with direct learning, the 2-task curriculum that learns whole-body balancing first and then locomotion, and the 3-task curriculum that learns upper-body balancing, then whole-body balancing and only then it learns locomotion. Curriculum switching strategies include (1) the time-step-based switching strategy, TIM, (2) the successful-balancing-based switching strategy, BAL, and (3) the recurrent-neural-network-based switching strategy, RNN. Corresponding options applied at the switching moments are listed on the right-hand side of the figure and explained later in the article.

We consider learning the deterministic control policy $\boldsymbol{u} = \pi(\boldsymbol{x})$. The value function $Q^\pi(\boldsymbol{x}, \boldsymbol{u})$ of the policy π denotes the expected return after taking action $\boldsymbol{u}$ in state $\boldsymbol{x}$ and then following π. The optimal control policy π^* maximizes the value function for each state. Therefore, the optimization of the control policy is tightly coupled with the maximization of the value function.

The Bellman equation defines a recursive relation between the value function at the current time step and the value function at the next time step (Sutton et al., 2011),

$$Q^\pi(\boldsymbol{x}_k, \boldsymbol{u}_k) = \mathbb{E}\Big\{ r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) + \gamma Q^\pi(\boldsymbol{x}_{k+1}, \pi(\boldsymbol{x}_{k+1})) \Big\}. \tag{6.2}$$

For real-world systems, continuous control is usually preferred. This requires a parametrization of the policy $\pi(\boldsymbol{x}; \boldsymbol{\theta}^{\mathrm{a}})$ using neural networks or a set of basis functions, and associated weights $\boldsymbol{\theta}^{\mathrm{a}}$. The weights are usually optimized by gradient descent methods (Williams, 1992; Bhatnagar et al., 2009; Grondman et al., 2012), or by global gradient-free methods (Hansen and Ostermeier, 2001; Botev et al., 2013). Because the estimation of policy gradients results in a high variance, the policy update is often coupled with an explicit estimation of the state-action value function $Q^\pi(\boldsymbol{x}, \boldsymbol{u}; \boldsymbol{\theta}^{\mathrm{c}})$ parametrized by its own set of parameters $\boldsymbol{\theta}^{\mathrm{c}}$. This combination is known as the actor-critic method, where the policy is referred to as the actor, and the value function is referred to as the critic.

6.2.2 Model-free deep reinforcement learning

Linear-in-parameters approximators of complex control policies and corresponding value functions, e.g. tile-coding (Sutton and Barto, 1998), require millions of parameters. A deep neural network (DNN), which is a global nonlinear function approximator, is used in this work to parametrize the control policy and the value function. It allows to significantly reduce the number of parameters but poses training difficulties such as value function divergence, the inherent difference of physi-

cal units in the state vector, correlated data and non-stationary distributions. In the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al., 2015), which we use in this thesis, these problems are solved by soft updates of the critic and actor network copies called target networks, batch normalization (Ioffe and Szegedy, 2015) and experience replay (Lin, 1993).

Every time step, a minibatch of N transitions is sampled from the experience replay buffer $\mathbb{B}$. Then DDPG calculates the predictions of the value function q_n for each minibatch transition n using the right-hand side of the Bellman equation (6.2),

$$q_n = r_{n+1} + \gamma \hat{Q}(\boldsymbol{x}_{n+1}, \hat{\pi}(\boldsymbol{x}_{n+1}; \hat{\boldsymbol{\theta}}^{\mathrm{a}}); \hat{\boldsymbol{\theta}}^{\mathrm{c}}), \tag{6.3}$$

where target networks $\hat{Q}$ and $\hat{\pi}$ are parametrized by their own parameters $\hat{\boldsymbol{\theta}}^{\mathrm{c}}$ and $\hat{\boldsymbol{\theta}}^{\mathrm{a}}$. The Q-value function is then updated via stochastic gradient descent on the loss function,

$$\sum_{n}^{N} \left(q_n - Q(\boldsymbol{x}_n, \boldsymbol{u}_n; \boldsymbol{\theta}^{\mathrm{c}})\right)^2 + \|\boldsymbol{\theta}^{\mathrm{c}}\|^2_{\mathbf{W}}, \tag{6.4}$$

where $\|\boldsymbol{\theta}^{\mathrm{c}}\|_{\mathbf{W}}$ is the ℓ_2 critic regularization term with positive definite weighting matrix $\mathbf{W}$. Next, the policy π is updated using the sampled policy gradient (Silver et al., 2014),

$$\nabla_{\boldsymbol{\theta}^{\mathrm{a}}} G = \frac{1}{N} \sum_{n}^{N} \nabla_{\boldsymbol{u}} Q(\boldsymbol{x}, \boldsymbol{u}; \boldsymbol{\theta}^{\mathrm{c}})|_{\boldsymbol{x}=\boldsymbol{x}_n, \boldsymbol{u}=\pi(\boldsymbol{x}_n)} \nabla_{\boldsymbol{\theta}^{\mathrm{a}}} \pi(\boldsymbol{x}; \boldsymbol{\theta}^{\mathrm{a}})|_{\boldsymbol{x}=\boldsymbol{x}_n}.$$

Finally, the target network parameters are updated by the target network update weight $\lambda \ll 1$,

$$\hat{\boldsymbol{\theta}}^{\mathrm{c}} \leftarrow \lambda \boldsymbol{\theta}^{\mathrm{c}} + (1-\lambda)\hat{\boldsymbol{\theta}}^{\mathrm{c}}, \qquad \hat{\boldsymbol{\theta}}^{\mathrm{a}} \leftarrow \lambda \boldsymbol{\theta}^{\mathrm{a}} + (1-\lambda)\hat{\boldsymbol{\theta}}^{\mathrm{a}}.$$

Algorithm 2 shows the DDPG loop. The termination indicator $\mathcal{J}$ initialized in line 3 identifies the moment when the learning loop is terminated. In the original version of DDPG, the indicator is simply the difference between the current time step and the number of time steps until termination. Other indicators are proposed in the next section. For the rest of the article, it is important to notice that in line 5 all encountered experience is stored in the experience replay buffer $\mathbb{B}$, while policies and value functions are updated using the minibatches of experience. The network parameters and the replay buffer are returned in line 8.

6.3 Proposed method

6.3.1 Curriculum learning

Even though the ultimate goal for the learning agent is to acquire the locomotion skill, we propose to learn the skill by sequencing several tasks. Algorithm 3 shows the proposed curriculum learning setup. In the beginning, we provide boolean variables PV, RB, and RR that specify the task-to-task transfer options.

Algorithm 2 Deep Deterministic Policy Gradient loop

1: **function** DDPG($\hat{\boldsymbol{\theta}}^{\mathrm{c}}$, $\hat{\boldsymbol{\theta}}^{\mathrm{a}}$, $\boldsymbol{\theta}^{\mathrm{c}}$, $\boldsymbol{\theta}^{\mathrm{a}}$, $\mathbb{B}$, τ)
2: Setup environment according to task τ
3: Initialize termination indicator $\mathcal{J} = 0$
4: **while** $\mathcal{J} \leq 0$ **do**
5: Execute a single DDPG step // Update the replay buffer and network parameters
6: Update indicator $\mathcal{J}$
7: **end while**
8: **return** $\hat{\boldsymbol{\theta}}^{\mathrm{c}}$, $\hat{\boldsymbol{\theta}}^{\mathrm{a}}$, $\boldsymbol{\theta}^{\mathrm{c}}$, $\boldsymbol{\theta}^{\mathrm{a}}$, $\mathbb{B}$
9: **end function**

- PV denotes the transfer of the policy and value function parameters, $\hat{\boldsymbol{\theta}}^{\mathrm{c}}$, $\hat{\boldsymbol{\theta}}^{\mathrm{a}}$, $\boldsymbol{\theta}^{\mathrm{c}}$, $\boldsymbol{\theta}^{\mathrm{a}}$.
- RB denotes the transfer of the replay buffer $\mathbb{B}$.
- RR denotes the transfer of the replay buffer with samples reevaluated according to the current task τ_i.

Options PV and RB/RR can be jointly and independently applied to each task. The combination of several options is denoted by the $\cdot$ symbol. In the case of the 3-task curriculum, two independent sets of options are provided; and notation $+$ separates them in the text.

In line 4, the algorithm picks the task τ_i from the enumerated list of tasks $\mathbb{T}$. Depending on the executed curriculum, starting task is either the upper-body balancing or the whole-body balancing, see Figure 6.1. The difference in the task rewards, if available, is obtained in line 5. Option RR assumes that we know the difference between the rewards of tasks τ_{i-1} and τ_i, which is not available in the model-free learning. In the experiments, we study both possible situations – when the difference is available and when it is not.

Note that keeping samples in the replay buffer without reevaluation does not pose a serious problem to DDPG. According to equation (6.3), old predictions q_n for the new task are wrong. However, the value function update in (6.4) is based on the minibatch of samples. The proportion of wrong samples in the minibatch diminishes with more correct samples added. Therefore, the RB option provides a smooth transition of gradient from the previous task to the new one.

After applying the provided options, we execute DDPG on the picked task τ_i. The DDPG algorithm monitors indicator $\mathcal{J}$ to decide when to terminate the current task. The switching moment is determined by the curriculum.

- TIM denotes the strategy that forces DDPG to return based on the number of manually defined time steps.
- BAL denotes the strategy that monitors the number of successfully accomplished balancing episodes and forces DDPG to return when a predefined

Algorithm 3 Curriculum learning

Input: Boolean variables PV, RB, RR
Initialize a collection of tasks $\mathbb{T}$
Initialize experience replay buffer $\mathbb{B} = \{\emptyset\}$
for $i = 1, N_\mathrm{I}$ **do**
    Pick a task $\tau_i \in \mathbb{T}$ according to some curriculum
    If available, pick the difference Δr_i of tasks τ_i and τ_{i-1} rewards
    if not PV **or** $i = 1$ **then**
        Initialize critic and actor weights $\boldsymbol{\theta}^\mathrm{c}$ and $\boldsymbol{\theta}^\mathrm{a}$ randomly
        Initialize target weights $\hat{\boldsymbol{\theta}}^\mathrm{c} \leftarrow \boldsymbol{\theta}^\mathrm{c}$ and $\hat{\boldsymbol{\theta}}^\mathrm{a} \leftarrow \boldsymbol{\theta}^\mathrm{a}$
    end if
    if RR **then**
        $\tilde{\mathbb{B}} = \{\emptyset\}$
        for each $(\boldsymbol{x}_n, \boldsymbol{u}_n, r_{n+1}, \boldsymbol{x}_{n+1}) \in \mathbb{B}$ **do**
            $r_{n+1} \leftarrow r_{n+1} + \Delta r_i(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{x}_{n+1})$
            Store transition $\tilde{\mathbb{B}} \leftarrow \tilde{\mathbb{B}} \cup (\boldsymbol{x}_n, \boldsymbol{u}_n, r_{n+1}, \boldsymbol{x}_{n+1})$
        end for
        $\mathbb{B} \leftarrow \tilde{\mathbb{B}}$
    else if not RB **then**
        Initialize experience replay buffer $\mathbb{B} = \{\emptyset\}$
    end if
    Call DDPG($\hat{\boldsymbol{\theta}}^\mathrm{c}$, $\hat{\boldsymbol{\theta}}^\mathrm{a}$, $\boldsymbol{\theta}^\mathrm{c}$, $\boldsymbol{\theta}^\mathrm{a}$, $\mathbb{B}$, τ_i) from Algorithm 2
    Collect new parameters $\hat{\boldsymbol{\theta}}^\mathrm{c}$, $\hat{\boldsymbol{\theta}}^\mathrm{a}$, $\boldsymbol{\theta}^\mathrm{c}$, $\boldsymbol{\theta}^\mathrm{a}$, and the replay buffer $\mathbb{B}$
end for

number is reached. BAL1 denotes a single successful episode, while BAL2, BAL5 and BAL10 denote two, five and ten successful episodes, respectively.

- RNN denotes the strategy that monitors several performance indicators and forces DDPG to return when a new task has been proposed by the task-switching RNN.

Performance indicators are evaluated every testing episode during which no exploration noise is applied to the policy. After the task switching, the next predefined task is picked by the curriculum, except for the RNN strategy which executes the task proposed by RNN.

6.3.2 Supervised learning of the task-switching network

Dataset collection

A naive approach to dataset collection could be to randomly sample durations of each task τ, perform a single TIM learning run, and collect the number of falls incurred by each run. Finally, runs with the number of falls smaller than some percentile could be added to the dataset. However, this approach quickly becomes inefficient with more tasks in the curriculum, because optimal task durations are

unlikely to appear during the random sampling. Therefore, we employ a combinatorial rare event optimization method known as a Cross-Entropy (CE) method (Rubinstein, 1997), see Figure 6.2a. The method is particularly suitable for noisy samples and time-consuming learning runs.

At each iteration, the CE method estimates the importance sampling density of rare events. In our case, the rare event corresponds to a well-performing run, i.e., the well-performing order of tasks and their durations which is illustrated in Figure 6.2a by yellow, blue and green color. The performance of each run is first evaluated, and then the best ϵ^{CE}-percentile of them is selected to update the sampling density, which is parametrized by some parameter vector $\boldsymbol{p}^{\mathrm{CE}}$. Here, we calculate the performance as the negative number of falls occurred during each run when the run reached the $(1-1/e) \approx 63\,\%$ of the total return obtained without the curriculum. Described steps are iterated multiple times, see the work of Rubinstein (1997) for more details. The sampling density, which is modeled as the multivariate Bernoulli density, allows updating the parameter vector $\boldsymbol{p}^{\mathrm{CE}}$ analytically. The final dataset is composed of the same ϵ^{CE} of all learning runs that achieved the minimum number of falls and $63\,\%$ of the total return of a run that learns to walk directly.

While collecting the dataset, we assume a monotonic succession of tasks such as "upper-body balancing $\rightarrow$ whole-body balancing $\rightarrow$ walking". Depending on the sampled task durations, the actual curriculum can skip one or even both tasks. By definition of the task reward functions given in the next section, the curriculum without walking cannot reach a high return. For this reason, it does not participate in the sampling density update.

Training the network

The choice of RNN is motivated by the fact that the history of the performance indicators expresses more than their single observation. This property is particularly relevant to the bipedal walking task, where policy performance is very uneven. For example, several updates of a working policy can result in large negative rewards due to robot falls or due to the robot being trapped in a double support phase.

Figure 6.2b shows the training process. Using the CE method, we collect a training dataset of learning runs, which records the performance indicators $\boldsymbol{I}$ and the corresponding tasks τ. The recurrent network $f(\boldsymbol{I}; \boldsymbol{\theta}^{\mathrm{s}})$ takes the set of indicators and outputs the estimated task probabilities $\boldsymbol{p} = [p_1, p_2, \dots, p_{|\mathbb{T}|}]$. RNN parameters $\boldsymbol{\theta}^{\mathrm{s}}$ are learned by gradient-based minimization of the CE loss $-\sum \boldsymbol{\tau}^{\top} \log f(\boldsymbol{I}; \boldsymbol{\theta}^{\mathrm{s}})$, where $\boldsymbol{\tau}$ is the one-hot vector transformation of task τ, and the logarithm is taken for each output of the network. For the current task τ, the switching moment is then detected by testing the termination indicator $\mathcal{J} = \max_{\tau'} p_{\tau'} - p_{\tau}$ in Algorithm 2.

Performance indicators

The performance of each task can be evaluated by various performance indicators $\boldsymbol{I}$, not necessarily the same for all systems and tasks. However, to make our results applicable to many tasks and systems, we suggest three universal indicators independently proposed in literature.

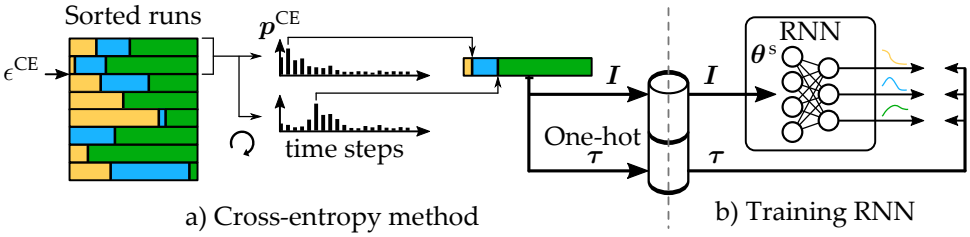


Figure 6.2: a) Visualization of the CE method for database collection. b) Supervised training of RNN. Yellow, blue and green color corresponds to the upper-body, whole-body balancing and walking tasks.

1. Testing episode length normalized to the maximum episode length. This indicator is related to the confidence of a patient performing some task (Veerbeek et al., 2014) and is also used by the BAL switching strategy.

2. Mean absolute temporal-difference error as a measure of a state-action value uncertainty proposed by Gehring and Precup (2013),

$$\mathbb{E}_{(\boldsymbol{x}_n, \boldsymbol{u}_n, r_{n+1}, \boldsymbol{x}_{n+1}) \sim \mathbb{B}} \left\{ \left| r_{n+1} + \gamma \hat{Q}(\boldsymbol{x}_{n+1}, \hat{\pi}(\boldsymbol{x}_{n+1}; \hat{\boldsymbol{\theta}}^{\text{a}}); \hat{\boldsymbol{\theta}}^{\text{c}}) - \hat{Q}(\boldsymbol{x}_n, \boldsymbol{u}_n; \hat{\boldsymbol{\theta}}^{\text{c}}) \right| \right\}.$$

   The indicator is usually high at the beginning of learning a new task but reduces to smaller values during convergence.

3. The network complexity estimated as the critic regularization term $\|\boldsymbol{\theta}^{\text{c}}\|^2_{\mathbf{W}}$. This indicator is inspired by the ℓ_2 regularization gain proposed by Graves et al. (2017), which is the difference between two regularization terms before and after training. We found that for our purpose the regularization term works better than the regularization gain of Graves et al. (2017).

The indicators are evaluated every testing episode and then concatenated into a single indicator $\boldsymbol{I}$. Later $\boldsymbol{I}$ is used for supervised training of the task-switching RNN $f(\boldsymbol{I}; \boldsymbol{\theta}^{\text{s}})$. Alternatively, the trained RNN is used for classifying the next task to practice based on the current run performance described by $\boldsymbol{I}$.

6.4 Experiment details

6.4.1 Systems

We use four systems to experiment with the curriculum learning, see Figure 6.3. The systems are initialized with the torso being upright except Halfcheetah-v1[3] which is initialized with the torso being horizontal to the ground. Small random noise is added to the initial joint angles. The goal for each system is to find the locomotion strategy which maximizes (6.1) with locomotion reward function r^{l} defined in Table 6.2. In the case of the 2- and 3-task curriculum, balancing rewards r^{b} are used. Episodes are terminated either due to timeout, which is 20 s for Leo and 16.5 s for

[3] For brevity, we omit -v1 suffix for the rest of the paper.

other systems, or due to the robot falling. Falls are recognized by certain configurations of the body height, pitch and limb contacts specified in Table 6.3 as an absorbing set $\mathcal{X}^{\mathrm{a}}$. Upon termination, the systems are reset to their initial positions.

Bipedal robot Leo is the real 7-degree of freedom (DoF) 2D walking robot specifically developed for conducting RL experiments (Schuitema, 2012). Pictures of the robot and its model are shown in Figure 6.3a. The robot is small (50 cm) in size and light-weight (1.7 kg). Foam bumpers are attached on both sides of the torso top and between the hip motors to secure the robot from damage due to falls in a wide range of configurations. Elastic couplings are added for the built-in protection of motor gearboxes. Leo is connected to the boom at the hip with power provided through a freely rotating joint. The on-board embedded computer communicates with 7 servomotors, three in each leg at the ankle, knee, and hip as well as one motor in the shoulder joint. A single arm facilitates preprogrammed stand up motion thereby delivering a completely autonomous learning platform. Motors actuate the joints in voltage control mode (max. voltage 10.8 V) and communicate back their position and temperature. The joint velocities are calculated using a finite difference and a subsequent filtering of the angular positions.

In the experiments, we use four models of Leo created using RBDL (Felis, 2017b) and simulated using Generic Reinforcement Learning Library (Caarls, 2015). The first model we call *idealized* because it has a normal torso weight. The second model, which we call *perturbed*, has an added torso mass within the range of ± 60 %. It is used for testing the generalization properties of the switching strategies. Alternatively, both models are referred to as the *whole-body* models. In the case of the 3-task curriculum, we additionally use *upper-body* models that are derived from the whole-body models by increasing the inertia of the lower body such that ankle and knee joints become immovable. The upper-body models allow to keep the state and action spaces the same for balancing and walking tasks. In all models, the arm joint is actuated by a proportional-derivative (PD) controller, and the actuation of the other joints is learned using DDPG.

Hopper, **Halfcheetah** and **Walker2d** are modeled using the Bullet Physics Engine[4]. These systems are commonly-used RL benchmarks but are not based on any existing hardware prototype. The systems use torque-based actuation and more complex reward functions. Duan et al. (2016) have previously shown poor DDPG performance on Hopper and Walker2d systems. We modified the original rewards to encourage more exploration by giving small negative penalties every time step, see Table 6.2. The large negative penalty is given at the absorbing states $\mathcal{X}^{\mathrm{a}}$ when the systems fall on the ground. With these modifications, DDPG regularly succeeds in learning locomotion.

Additionally, we found that when Hopper and Walker2d learn to balance, they tend to tilt the torso so much that its pitch almost exceeds the threshold. This happens because the models are not realistic: their torso mass is too small compared to the lower-body mass, see Appendix B.3 for a comparison with human anatomy. To avoid excessive torso tilting, we accommodate an extra term that penalizes postures

[4] https://github.com/bulletphysics/bullet3

that deviate too much from the upright posture. Near-absorbing sets $\mathcal{X}^{s}$ implement these penalties, see Table 6.3 for details.

6.4.2 Learning parameters

For each system, we selected a fixed total number of learning time steps which are equal to 300000, 600000, 600000, and 700000 for Leo, Hopper, Halfcheetah, and Walker2d, respectively.

We keep algorithm-related parameters the same for all systems and experiments. The DDPG[5] critic and actor networks consist of 2 hidden layers with 300 and 400 neurons, respectively. Hidden layers use ReLU activation functions, while the output layers use linear and tanh functions for critic and actor, respectively. Learning rates of $\alpha^{c} = 0.001$ and $\alpha^{a} = 0.0001$ are chosen for the critic and actor. Additional parameters include the target network update weight $\lambda = 0.001$, the discount rate $\gamma = 0.99$, the minibatch size $N = 64$, and the critic regularization weight $\mathbf{W}$, which is an identity matrix multiplied by 0.001. Before the first policy update by DDPG, it is preferable to collect an initial batch of samples. Preliminary hyperparameter sensitivity test suggested that 1000 samples in the replay buffer is an optimal number. After collecting the initial batch of samples, single gradient decent step is performed every time step. Exploration is achieved by an Ornstein-Uhlenbeck (OU) noise $\mathcal{N}$ with parameters $\sigma = 0.15$ and $\theta = 0.20$. Every thirty-first episode is the *testing* episode, during which the noise is not applied.

The CE method iterates 6 times, each time recording data of 96 runs. The best $\epsilon^{\text{CE}} = 10\,\%$ of runs are used to update the CE parameters $\boldsymbol{p}^{\text{CE}}$ with smoothing of 0.8. The final dataset consists of 10 % of the best runs, 70 % of which are used for training and the rest is used for validation. The network consists of one hidden gated recurrent unit layer with 6 neurons and tanh activation function. The softmax output layer with 3 neurons indicates the task probabilities: upper-body balancing, whole-body balancing, or walking. The network is trained with learning rate of 0.01, and a dropout with a keep rate of 0.7 is applied to the hidden layer.

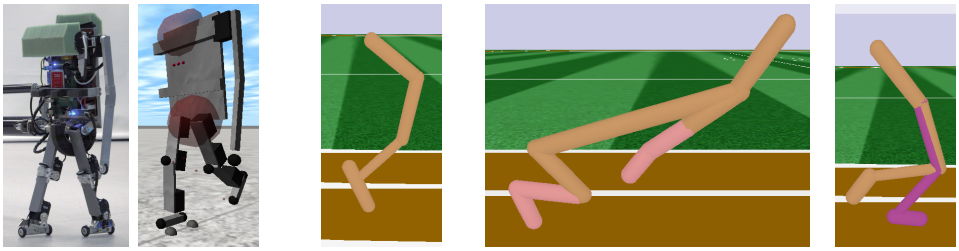
6.4.3 Evaluation methodology

We begin experimenting with the TIM strategy for which we manually define the number of time steps to practice. For the 2-task curriculum, balancing time equals to 17 % of the total learning time for all systems except Walker2d for which balancing time equals to the 29 % of the total time.

We hypothesize that Roboschool systems will not perform differently on the 2- and 3-task curricula due to their light upper body. For this reason, we perform the 3-task curriculum experiment for Leo only. Leo learns the upper-body and the whole-body balancing for 7 % and 10 % of total time allocated for learning. The 2- and 3-task curriculum parameters were selected after several pilot experiments with the systems, and may not be the optimal parameters for damage reduction.

In the 2-task curriculum, reevaluation is used to fix the discrepancy between the samples originating from different tasks but same systems. In the 3-task curricu-

[5] Our implementation of DDPG and RNN is available online at `https://github.com/ikoryakovskiy/curriculum_learning.git`



(a) Leo and its model (b) Hopper (c) Halfcheetah (d) Walker2d

Figure 6.3: Illustration of systems used in the experiments.

Table 6.2: Reward functions used for learning. SI metric system with omitted units is used for all variables. Control signal $\boldsymbol{u}$ is voltage for Leo and torque for other systems, $\dot{s}$ and $\dot{\phi}$ are the forward velocities of the models and angular velocities of their motors, respectively. The number of DoFs is denoted as n (3 for Hopper and 6 for Halfcheetah/Walker2d) with n^{lim} being the number of DoFs locked at their limits. $\mathcal{X}^{\text{a}}$ and $\mathcal{X}^{\text{s}}$ are the absorbing and near-absorbing sets described in Table 6.3. Notation $[\cdot]$ means the Iverson brackets. The difference between the locomotion and balancing rewards is shown in blue.

System	Task	Reward
Robot Leo	Balancing	$r^{\text{b}} = -0.014 \left\|\boldsymbol{u}\right\|_2^2 - 75 \left[\boldsymbol{x} \in \mathcal{X}^{\text{a}}\right]$
	Locomotion	$r^{\text{l}} = r^{\text{b}} + \Delta r = r^{\text{b}} + 10\dot{s} - 1.5$
Hopper Halfcheetah Walker2d	Balancing	$r^{\text{b}} = -\frac{2}{n}\boldsymbol{u}^\top\dot{\boldsymbol{\phi}} - \frac{0.1}{n}\left\|\boldsymbol{u}\right\|_2^2 - 0.2n^{\text{lim}} - 5\left[\boldsymbol{x} \in \mathcal{X}^{\text{s}}\right] - 75\left[\boldsymbol{x} \in \mathcal{X}^{\text{a}}\right]$
	Locomotion	$r^{\text{l}} = r^{\text{b}} + \Delta r = r^{\text{b}} + 3\dot{s} - 1.5$

lum, the upper-body balancing task is performed on the upper-body model of Leo with high inertia of the lower body. Hence, the discrepancy between the samples cannot be eliminated by the buffer reevaluation. For this reason, and also to limit the number of options tested during the transfer from the whole-body balancing to walking, we exclude the RR option in the 3-task curriculum experiments.

The experience replay buffer is organized as a queue of a fixed size. Because we know the duration of each task, the size of the queue is set to the number of time steps in the current task. This ensures that samples from the previous task are completely removed from the queue at the end of the current task. Note that the standalone PV option potentially may increase the number of falls encountered in a new task because DDPG waits for 1000 replay buffer samples before the first policy update. However, the previously learned and transferred policy can prevent a robot from falling.

Based on the TIM strategy outcome, we narrow down the number of investigated options in the subsequent experiments with BAL and RNN strategies. Since

Table 6.3: Definition of the absorbing $\mathcal{X}^{\mathrm{a}}$ and near-absorbing $\mathcal{X}^{\mathrm{s}}$ state sets. Note that $\mathcal{X}^{\mathrm{s}}$ is used only by Hopper and Walker2d systems. The body height above the ground is calculated by the $height(\boldsymbol{x})$ function, and the body pitch relative to the vertical axis is calculated by the $pitch(\boldsymbol{x})$ function. Indicators $contact\_shin$ and $contact\_thigh$ prevent ground contacts of body parts other than feet.

System	Definition
Robot Leo	$\mathcal{X}^{\mathrm{a}} = \{\boldsymbol{x} \mid height(\boldsymbol{x}) \leq 0.15\,\mathrm{m} \vee \lvert pitch(\boldsymbol{x})\rvert \geq 1.0\,\mathrm{rad} \vee \lvert\boldsymbol{\phi}_{\mathrm{ankle}}\rvert \geq 1.1\,\mathrm{rad}\}$
Hopper Walker2d	$\mathcal{X}^{\mathrm{a}} = \{\boldsymbol{x} \mid height(\boldsymbol{x}) \leq 0.60\,\mathrm{m} \vee \lvert pitch(\boldsymbol{x})\rvert \geq 1.5\,\mathrm{rad} \vee contact\_thigh\}$ $\mathcal{X}^{\mathrm{s}} = \{\boldsymbol{x} \mid height(\boldsymbol{x}) \leq 0.80\,\mathrm{m} \vee \lvert pitch(\boldsymbol{x})\rvert \geq 1.0\,\mathrm{rad}\}$
Halfcheetah	$\mathcal{X}^{\mathrm{a}} = \{\boldsymbol{x} \mid \lvert pitch(\boldsymbol{x})\rvert \geq 1.0\,\mathrm{rad} \vee contact\_shin \vee contact\_thigh\}$

for both strategies the switching time is not known in advance, we simply keep all samples in the replay buffer.

6.4.4 Evaluation metrics

Throughout the experiments, we compare the options and strategies listed in Figure 6.1 in terms of the undiscounted return G^1, the cumulative number of falls, and locomotion distance S. We use ANOVA test to determine a statistically significant differences between the means of options and post-hoc TukeyHSD test to label options not significantly different from the options with the minimum number of falls (family-wise error rate $\alpha > 0.05$). For each curriculum strategy or option 16 independent runs are used unless we mention a different number.

To investigate the reasons behind the success or failure of a certain curriculum, we additionally calculate the number of falls until a certain value of a cumulative mean exploration intensity Ξ is reached. Calculating the number of falls in this way allows comparing the falls-averting properties of the current control policy π with respect to the amount of exploration conducted by this policy. As a measure of the exploration intensity, we take the average distance to a previously observed nearest-neighbor for each learning time step k,

$$\xi_k = \frac{1}{k_{\mathrm{w}}} \sum_{i=k}^{k+k_{\mathrm{w}}} \min_{j=0..k-1} \left\| (\boldsymbol{x}_i - \boldsymbol{x}_j)\boldsymbol{\Sigma}^{-1} \right\|_2 ,$$

where $k_{\mathrm{w}} = 30$ is a sliding window and $\boldsymbol{\Sigma}$ is a diagonal matrix with the estimated standard deviations of corresponding state elements. Essentially, $\boldsymbol{\Sigma}^{-1}$ ensures that angles and angular velocities are scaled to approximately similar range. At the beginning of every task, the exploration intensity is usually very high. After a reasonable policy is acquired, the exploration intensity goes down, which indicates the policy fine-tuning with a random noise applied.

The falls-averting property analysis is conducted in two distinctive periods that correspond to the balancing and walking periods of the 2-task curriculum learning.

Each period starts at the beginning of each task and runs until the time step K at which the cumulative mean exploration intensity $\Xi = \sum_k^K \xi_k$ reaches a predefined threshold. In the case of the direct learning, each period starts at the precisely same moments as in the case of the 2-task curriculum learning and runs until the same threshold. Note that the cumulative mean exploration intensity and the cumulative number of falls are reset to zero at the beginning of each period.

Finally, we investigate the difference between the locomotion and balancing tasks by evaluating the difference between controls applied by corresponding policies in every state of a single state trajectory $(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, ...)$. For this comparison, the locomotion and balancing policies are learned independently from each other.

6.5 Results

6.5.1 TIM: curriculum learning with manual selection of time steps to practice

Two-task curriculum

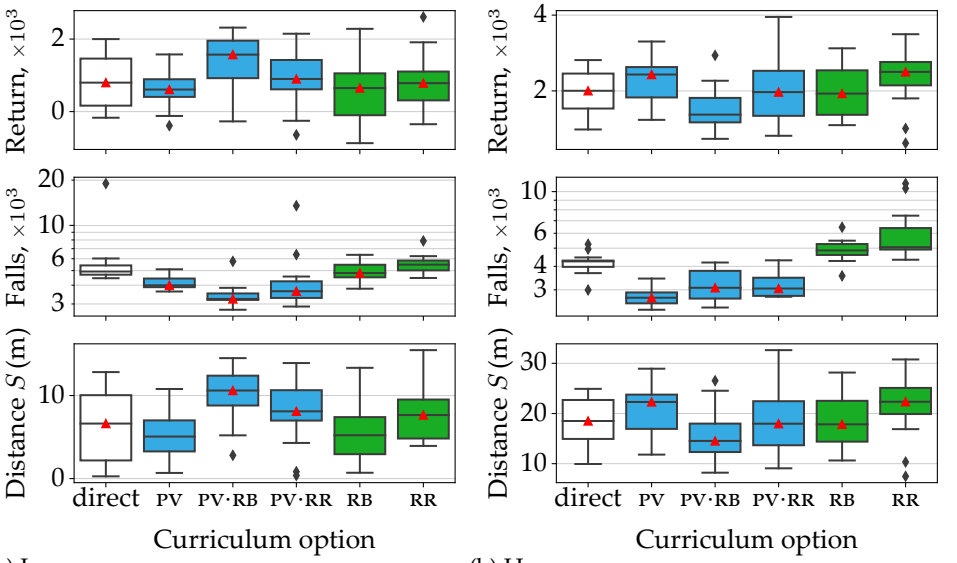
Figure 6.4 shows the locomotion learning results by the TIM strategy and various curriculum options. The common deep RL pitfall is that learning does not always succeed even when the starting parameters are held the same across independent runs (Henderson et al., 2017). The only difference between the runs in our experiments is the random-seed value, which we sample uniformly before learning. The results of the failed runs often appear as outliers in figures, but they are not excluded from the performance calculation unless we state the opposite.

By looking at the number of falls, we notice that for all systems except Walker2d there are curriculum options that significantly reduce the number of falls compared to direct locomotion learning. In particular, transferring the policy and the value function using the PV option is essential for reducing the number of falls. Possible reasons for why curriculum learning does not significantly improve the Walker2d results will be discussed in Section 6.6.

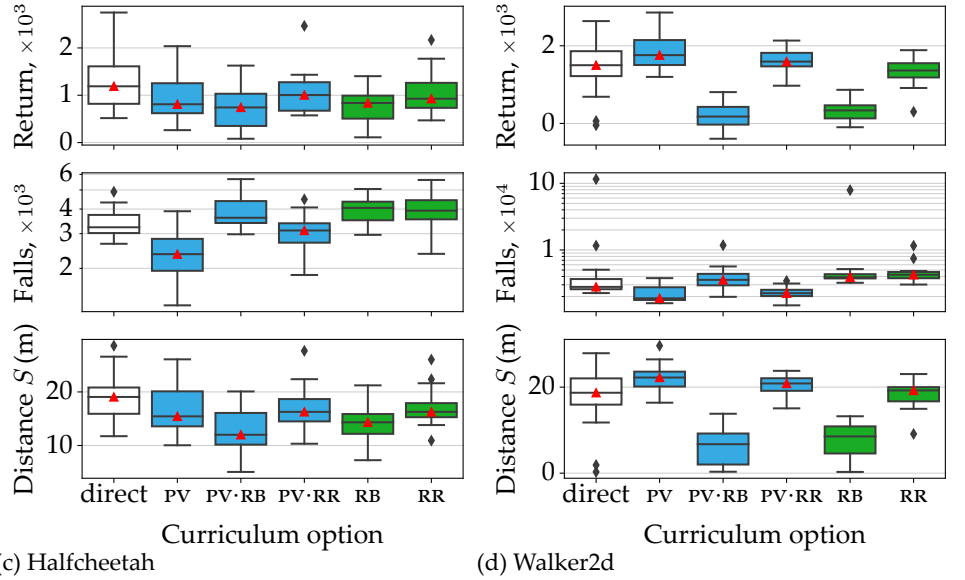
The results vary across systems if both PV and RB options are applied. For instance, the minimum number of falls for Leo is incurred by the PV·RB option. Interestingly, while standalone PV and RB options perform somewhat poorly on Leo, their combination performs exceptionally well.[6] However, the PV·RB option performs worse than PV on Hopper and Halfcheetah. While significantly reducing the number of falls, corresponding PV·RB and PV options maintain the total return and locomotion distance similar to direct learning.

The worst result in this series of experiments is attained by transferring only the replay buffer. For all four systems, the single RB option results in either unsatisfactory reward, falls or locomotion distance. The reevaluation of the replay buffer with option RR improves the performance compared to RB, but does not result in a significant advantage. Notably, reevaluation of the replay buffer with option PV·RR bridges the gap between PV·RB and PV by always attaining not significantly different

[6] Although not revealed by TukeyHSD test, the PV·RB option consistently outperformed all other options in our experiments with Leo.



(a) Leo (b) Hopper



(c) Halfcheetah (d) Walker2d

Figure 6.4: Results of the 2-task curriculum learning on four systems. Outliers are plotted as individual diamond-shaped points. *Red triangles* label median performance of options not significantly different ($\alpha > 0.05$) from the options with the minimum number of falls (PV·RB for Leo, PV for other systems).

results from the best option for each system.

Table 6.4 compares the best curriculum options with direct learning. Here, we record the time required to reach $\approx 63\,\%$ of the total return obtained without a curriculum, and the corresponding number of falls incurred during this time. Learning runs that could not reach the specified return were excluded from the calculations. Since Leo is the only model of a real robot, it is particularly useful to note that learning time has increased by $47.9\,\%$ in exchange for a $26.4\,\%$ reduction in falls compared to learning without the curriculum. For Roboschool systems, learning time has increased by at least $19.1\,\%$ while the number of falls has reduced by at least $21.1\,\%$.

Sensitivity to the number of balancing samples

Figure 6.7 on page 98 shows the performance of the 2-task curriculum with a varied number of balancing samples applied to Leo. Undertraining, which happens when the number of balancing samples is less than 50×10^3, results in an increased number of falls. However, it does not influence the total return or the walking distance. Overtraining, which happens when the number of balancing samples is more than 50×10^3, results in a slightly increasing trend in falls and slightly decreasing trend in the total return and the walking distance. This result means that Leo continues to balance more often in spite of the new walking reward being applied. Figure 6.7 suggests that 50×10^3 balancing samples before walking is the optimal training duration.

Three-task curriculum

Figure 6.5 shows results of the 3-task curriculum used for Leo to learn walking. Option PV·RB+PV·RB attains a number of falls that is significantly lower compared to direct learning. The total return and the walking distance appear to be at least as good as for the direct learning. However, the majority of options is not significantly different from this option. Therefore, we group the results by color. The results in the *blue* group transfer the policy and the value function every time the task changes. The results in the *yellow* group transfer the policy and the value function only from the second to the third task. In the similar situation, the final *green* group of results transfers only the replay buffer.

We observe that the PV option applied before the walking task has a major positive influence on reducing the number of falls. The number of falls obtained in the *blue* and *yellow* groups are mostly lower compared to the *green* group. The RB option also appears to reduce the number of falls, e.g. compare PV+PV with PV+PV·RB, PV·RB+PV with PV·RB+PV·RB, RB+PV with RB+PV·RB, PV+PV·RB with PV·RB+PV·RB, and finally PV+RB with PV·RB+RB.

Table 6.4 shows the number of robot falls in comparison to the direct and 2-task curriculum learning. Although not significantly, the number of falls incurred by the 3-task curriculum is lower compared to the 2-task curriculum. Compared to learning without the curriculum, learning time has increased by $40.5\,\%$ in exchange for a $31.8\,\%$ reduction in falls. A better performance could be potentially achieved with a better tuning of the task durations.

Table 6.4: Learning time and the cumulative number of falls the systems require to reach $(1-1/e) \approx 63\,\%$ of their total return obtained without the curriculum. The combination of options with the least number of falls is selected for the 2- and 3-task curricula. Runs that could not reach the target return are excluded from the calculations, and their success rate is adjusted accordingly.

System	Curriculum	Return	Time (h)	# of falls	Success (%)
Robot Leo	Direct learning	> 526	$\mathbf{1.21 \pm 0.19}$	4132 ± 337	93.75
	2-task curriculum	> 526	1.79 ± 0.13	$\mathbf{3017 \pm 344}$	93.75
	3-task curriculum	> 526	1.70 ± 0.08	$\mathbf{2820 \pm 192}$	90.63
Hopper	Direct learning	> 1253	$\mathbf{1.31 \pm 0.21}$	2407 ± 300	100.00
	2-task curriculum	> 1253	1.56 ± 0.17	$\mathbf{1243 \pm 174}$	100.00
Halfcheetah	Direct learning	> 807	$\mathbf{1.69 \pm 0.23}$	2830 ± 277	100.00
	2-task curriculum	> 807	2.10 ± 0.23	$\mathbf{2027 \pm 337}$	100.00
Walker2d	Direct learning	> 925	$\mathbf{2.07 \pm 0.43}$	2345 ± 493	87.50
	2-task curriculum	> 925	3.38 ± 0.23	$\mathbf{1850 \pm 276}$	87.50

In-depth study of the curriculum and the direct locomotion learning

In this section, we show additional results for the 2-task curriculum learning and the direct locomotion learning. These results reveal the reasons of the damage reduction when curriculum learning is used.

The number of falls incurred by the direct and 2-task curriculum learning while reaching the same cumulative mean exploration intensity in the aforementioned periods (cf. Section 6.4.4) is shown in Table 6.5. During the first period, the balancing task of the 2-task curriculum always incurs a smaller number of falls compared to the walking task of the direct learning. The number of falls is reduced by at least 22.5 % in this period. During the second period, for all systems except Walker2d, curriculum learning achieves a similar or lower number of falls compared to direct learning.

Finally, we investigate the similarity of the balancing and walking policies. Figure 6.6 shows the exemplar trajectories of a single leg. The black color of trajectories demonstrates that Leo, Hopper, and Halfcheetah use similar actuation while walking and balancing, particularly at the moments of touchdown. In contrast, Walker2d does not use the same actuation. In fact, the opposite hip and knee actuation patterns are observed in Leo and Walker2d.

Selection of options

Reevaluation of the replay buffer fails to outperform all other options, even though the reevaluated samples correspond to the true reward function. Because of the balancing task, the samples originate from a small region of the state space. However, learning locomotion requires samples from a much wider region. We hypothesize

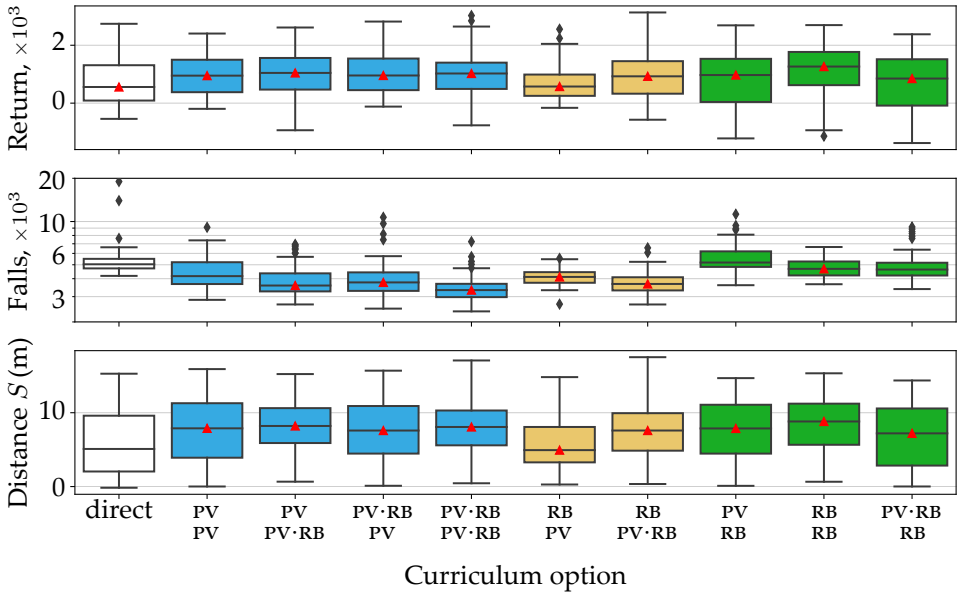


Figure 6.5: Results of the three-task curriculum on the idealized model of Leo. Outliers are plotted as individual diamond-shaped points. *Red triangles* label options not significantly different ($\alpha > 0.05$) from the PV·RB+PV·RB option with the minimum number of falls. The first and second rows of x-axis labels specify options used for switching from the upper-body balancing to the whole-body balancing and then to walking. Means with upper and lower 95% confidence limits are shown for 64 samples.

that any system that is able to balance can explore this wider region, thereby reaching a total return and locomotion distance similar to the options with the buffer reevaluation. We exclude the RR option from further consideration given the fact that reevaluation 1) requires the knowledge of the task difference Δr, 2) is impractical for modified system dynamics, and 3) results in an unsatisfactory improvement.

Experiments in the next sections are applied to Leo only. Both 2- and 3-task curricula use the PV·RB option. Note that while in this section at the end of a current task the replay buffer transitions from a previous task are completely discarded, in the next sections we keep all transitions in the replay buffer. This simplifies the automated task-switching strategies since the task-switching moments are not known in advance.

6.5.2 BAL: automated curriculum learning with the duration of balancing as a task-switching indicator

Figure 6.8 shows the results of the curriculum, in which the switching moment is detected based on the number of episodes that achieved a successful balancing duration over 5 s. Because balancing requires a static pose, 5 s is usually enough for determining successful task completion. The red dashed line shows the median performance of the *optimal* run taken from Figure 6.7. Here, neither the 2- nor 3-task curriculum could reach a number of falls as low as the TIM strategy with 50×10^3

Table 6.5: Comparison of the cumulative number of falls incurred by the direct and 2-task curriculum learning. *Two-task curriculum*: the number of falls is calculated since the beginning of each task and until the cumulative mean exploration intensity Ξ reaches a predefined threshold. *Direct learning*: the number of falls is calculated until the cumulative mean exploration intensity Ξ reaches the same threshold twice to match with corresponding balancing and walking tasks of the 2-task curriculum. For all systems except Walker2d, curriculum learning achieves the similar or lower number of falls compared to the direct learning. The threshold is calculated as the minimum value of maximum cumulative exploration intensities calculated during the described periods.

System	Period	Strategy	Curriculum task	Threshold intensity Ξ	Cumulative # of falls
Robot Leo	1	Direct learning	–	536.4	2084 ± 267
		2-task curriculum	balancing		$\mathbf{1615 \pm 238}$
	2	Direct learning	–	1853.7	1708 ± 356
		2-task curriculum	walking		$\mathbf{1123 \pm 203}$
Hopper	1	Direct learning	–	391.3	1387 ± 255
		2-task curriculum	balancing		$\mathbf{808 \pm 128}$
	2	Direct learning	–	1304.1	$\mathbf{1150 \pm 129}$
		2-task curriculum	walking		$\mathbf{1122 \pm 102}$
Half-cheetah	1	Direct learning	–	695.0	1194 ± 95
		2-task curriculum	balancing		$\mathbf{909 \pm 102}$
	2	Direct learning	–	2423.0	$\mathbf{994 \pm 314}$
		2-task curriculum	walking		$\mathbf{1185 \pm 269}$
Walker2d	1	Direct learning	–	1296.7	1406 ± 151
		2-task curriculum	balancing		$\mathbf{852 \pm 53}$
	2	Direct learning	–	1739.1	$\mathbf{370 \pm 294}$
		2-task curriculum	walking		722 ± 127

time steps. Finding the optimal number of successful balancing episodes for the 3-task curriculum is particularly difficult, and this is reflected in its low performance in terms of the return and the walking distance.

Out of all tested strategies, the 2-task BAL10 strategy incurs the minimum number of falls. However, the low total return suggests that long balancing results in overtraining. BAL2 and BAL5 strategies show a balanced performance with a high return and walking distance, but also a higher number of falls. Finally, the BAL1 strategy results in undertraining, which leads to the worst performance compared to the other 2-task options.

6.5.3 RNN: automated curriculum learning with RNN-based identification of task-switching moments

Figure 6.9a shows the obtained CE distributions after six iterations. The previous experiments showed that the upper-body balancing task is very easy. For this reason, a logarithmic scale of time steps is used for the moments when the upper-body bal-

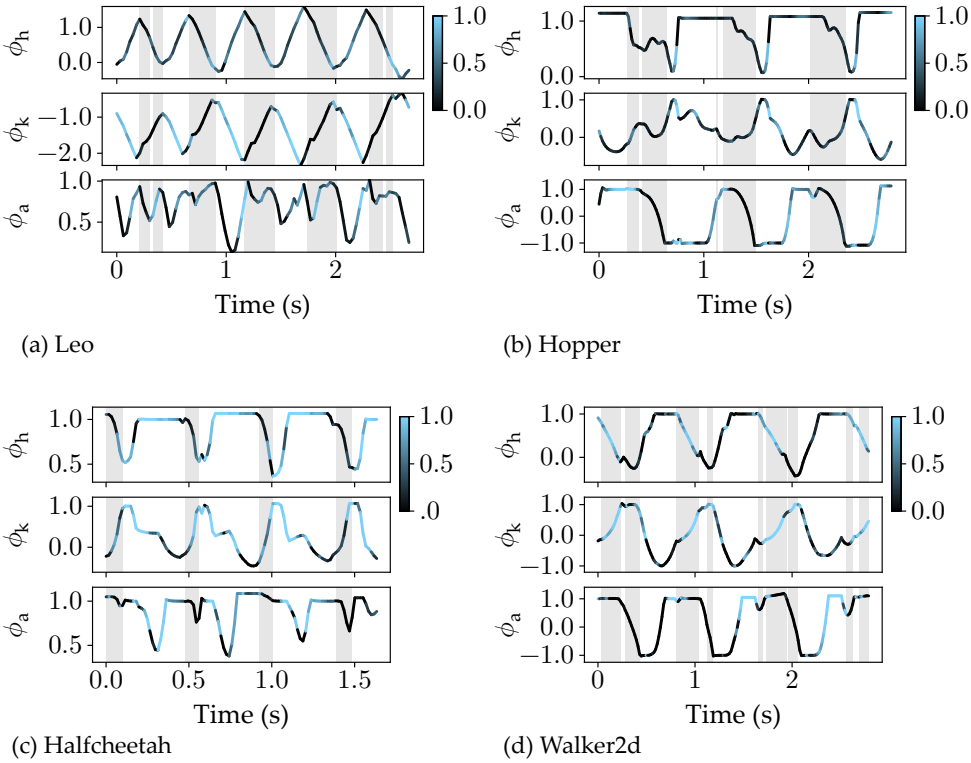


Figure 6.6: Angle trajectories of the left (or front for Halfcheetah) hip ϕ_h, knee ϕ_k and ankle ϕ_a during locomotion. Plots (*a*), (*b*) and (*c*) illustrate that when the leg is in contact with the ground (*shaded area*), controls are similar to those applied during balancing (*black line*). Less similarity with balancing controls (*blue line*) is observed when the leg is swinging in the air. Comparable similarity is not observed in plot (*d*). Leo angles are given in radians, while Roboschool angles are normalized to the interval $[-1.0, 1.0]$.

ancing task switches to the whole-body one (upper plot in Figure 6.9a). From these results, we see that (1) the optimal switching to the whole-body balancing task is at the 1833 time steps since the beginning of learning, and (2) the optimal switching to the walking task is at the 45000 time steps since the beginning of the upper-body balancing task. Switching the tasks at these moments realizes the 3-task curriculum that incurs the minimum number of falls on average.

The validation result of RNN training is shown in Figure 6.9b. The chosen indicators do not allow to reliably distinguish the upper-body balancing task, resulting in a true positive rate (sensitivity) at only 56.5 %. However, the confusion matrix shows that indicators of the upper-body balancing task are hardly misclassified as belonging to the walking task. Hence, the whole-body balancing stage is unlikely to be skipped. Classification of the other two tasks is much more reliable, reaching at least 93.5 % sensitivity. Specificity of the classifier is also relatively high, reaching at least 86.0 %.

Figure 6.9c presents the learning results with the switching moments detected

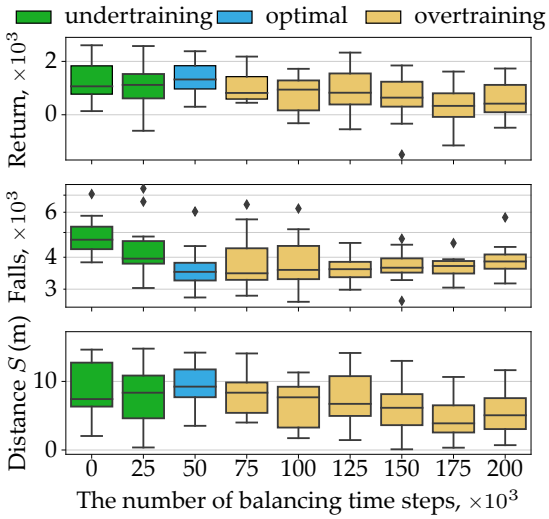


Figure 6.7: Two-task curriculum learning results with the TIM switching strategy applied to Leo. The performance of the strategy varies with the number of balancing time steps and reaches the optimum at 50000 steps. The number of walking time steps is fixed. Direct learning corresponds to a 0 number of balancing steps. The PV·RB option is applied at the task-switching moments.

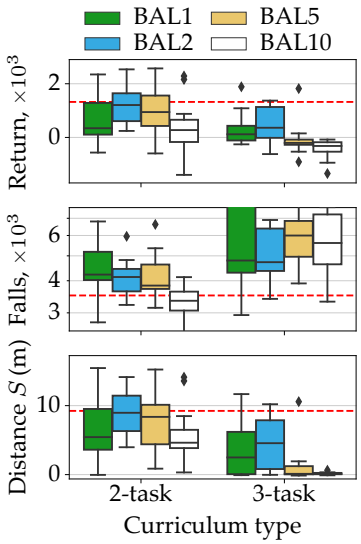


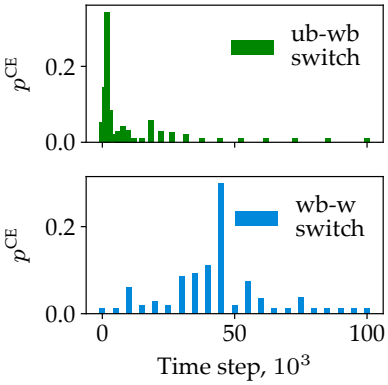
Figure 6.8: Two- and 3-task curriculum learning results with BAL switching strategies applied to Leo. Neither strategy could reach the optimal performance of the TIM strategy shown by the *red dashed* line, cf. Figure 6.7. The PV·RB option is applied at the task-switching moments.

using RNN on the perturbed model of Leo subject to $\pm 60\,\%$ added torso mass. For all strategies, a heavier torso leads to a greater number of falls. However, the TIM, BAL2 and RNN strategies consistently outperform the direct learning in terms of Leo falls. Moreover, the curriculum strategies outperform the direct learning in terms of the total return for the added torso mass of $\leq -40\,\%$, which is a surprising result. For the heavier torso (added mass $\geq 40\,\%$) the direct learning attains a greater return.

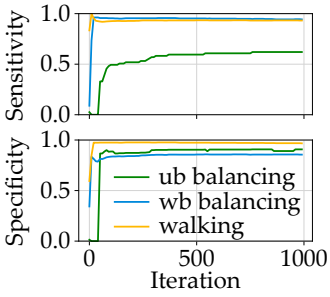
Table 6.6 shows the decrease of falls compared to direct learning. Although not entirely consistent, the TIM strategy performs the best among the other curriculum strategies. The RNN strategy often reaches similar or slightly greater number of falls, but lags behind in terms of the total return and the walking distance, which is reflected in the learning success rate. The TIM and BAL2 strategies perform similar in terms of the return. However, the latter consistently attains a greater number of falls compared to the other strategies.

6.6 Discussion

Our results demonstrate that learning to balance before learning locomotion can significantly reduce the number of falls. For DDPG, transferring the policy and the

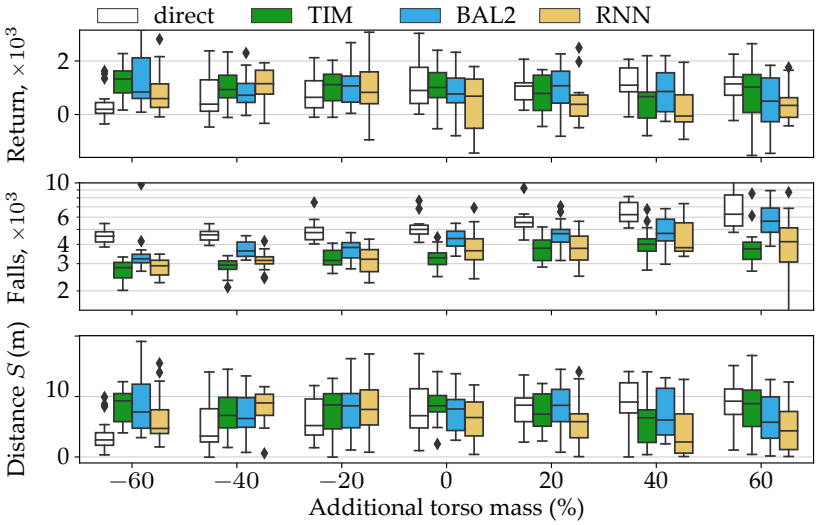


(a) Rare-event probability distribution estimated using cross-entropy method after six iterations. Non-zero and non-coinciding peaks demonstrate that the upper- and whole-body balancing tasks reduce damage while learning to walk.



Actual \ Predicted	ub	wb	w
ub	45	33	1
wb	7	497	23
w	0	43	749

(b) *(top)* Supervised training of RNN task classifier. *(bottom)* Confusion matrix of the validation dataset.



(c) Comparison of the generalization of the direct learning and the TIM, BAL2 and RNN strategies subject to the torso mass perturbations. All strategies keep samples in the replay buffer. The TIM and BAL2 strategies use the most optimal moments of switching identified on the idealized model. The PV·RB option is applied at the task-switching moments.

Figure 6.9: CE dataset collection, RNN training, and performance evaluation. Labels "ub", "wb" and "w" refer to upper-body balancing, whole-body balancing and walking, respectively.

Table 6.6: Average success rate and the decrease of falls compared to direct learning while reaching a return of 526, cf. Table 6.4. All strategies keep samples in the replay buffer. The TIM and BAL2 strategies use the optimal moments of switching identified on the idealized model. The PV·RB option is applied at the task-switching moments. Runs that could not reach the target return are excluded from fall calculations, and the success rate is adjusted accordingly.

Criterion	Added torso mass, %	TIM	BAL2	RNN
Success rate, %	Average	$\mathbf{94.6 \pm 5.2}$	$\mathbf{97.3 \pm 4.5}$	83.0 ± 14.0
Decrease of falls, %	-60	$\mathbf{-32.8 \pm 5.8}$	$\mathbf{-14.7 \pm 25.4}$	$\mathbf{-28.3 \pm 5.0}$
	-40	$\mathbf{-34.4 \pm 4.1}$	-15.4 ± 7.8	$\mathbf{-30.1 \pm 5.3}$
	-20	$\mathbf{-30.3 \pm 5.9}$	$\mathbf{-27.4 \pm 7.9}$	$\mathbf{-25.3 \pm 9.2}$
	0	$\mathbf{-29.4 \pm 6.8}$	-10.4 ± 10.8	$\mathbf{-23.1 \pm 8.1}$
	20	$\mathbf{-29.1 \pm 8.7}$	$\mathbf{-16.0 \pm 15.3}$	$\mathbf{-27.4 \pm 11.9}$
	40	$\mathbf{-35.4 \pm 12.3}$	-21.1 ± 12.3	$\mathbf{-39.6 \pm 9.9}$
	60	$\mathbf{-48.3 \pm 8.0}$	-19.0 ± 16.0	$\mathbf{-41.7 \pm 12.7}$

value function using the PV option is necessary for achieving this result. Transferring the replay buffer additionally to the PV option further reduces the number of Leo falls, although the difference with standalone PV is not significant. The downside of the curriculum learning is the increasing learning time. However, we believe that in many cases it is more acceptable than the repair costs, which in case of Leo usually involve the time of disassembling the broken leg, changing the motor gearbox and calibrating the encoder.

Our detailed study of the exploration intensity and the cumulative number of falls reveals two important facts. First, the robots accumulate less falls at the end of the balancing task compared to the same period of the direct locomotion learning. Since this result holds for all systems, the most probable explanation is that learning the statically-stable balancing task is easier than the walking task. Second, the learned balancing policy averts unsafe exploration actions during the walking task. This fact applies only to Leo. We hypothesize that the balancing task is particularly important for supporting the heavy torso of Leo, while the Roboschool systems do not benefit from the balancing task due to their light-weight upper bodies. This hypothesis is supported by the difference in the number of falls observed for higher torso mass in Figure 6.9c and Table 6.6.

The knowledge of balancing allows Leo to learn the walking task more safely. During the walking task, the preconditioned policy parameters are gradually revised, therefore ensuring a slow transition from one policy to another. This policy is attracted to local maxima different from the direct learning maxima. The statement is supported by the contrasting performance of the policies learned using direct and curriculum learning on the perturbed model of Leo. For the lighter torso, curriculum learning finds better local maxima, while direct learning gets stuck in a suboptimal behavior, which leads to frequent falls. For the heavier torso, the direct learning finds better local maxima. Although curriculum learning occasionally gets

stuck in a suboptimal behavior, this behavior prefers balancing to falling, which is reflected in the lower number of robot falls. Similar observations of the difference between the local maxima were also found in supervised learning (Bengio et al., 2009; Erhan et al., 2010; Kumar et al., 2010).

Curriculum learning does not achieve a significant fall reduction for Walker2d. A careful study of the Walker2d trajectories reveals that the balancing policy is not reused during walking the same way as it is reused by other systems, notably by Leo. We believe that the foot construction and the unrealistically lightweight upper body cause this difference. Our preliminary experiment with the Walker2d ankle placed in the middle of the foot as it is in Leo showed a significant reduction of falls.

Both TIM and BAL strategies require the selection of various parameters that identify the switching moments. If not selected correctly, undertraining or overtraining occurs. However, undertraining results in a stronger negative influence on the number of falls compared to overtraining. Thus, a larger number of balancing time steps is less detrimental than the smaller number.

Although the BAL strategy demonstrates a reduction of falls compared to direct walking, it suffers from an innate pitfall. Occasionally, it happens that RL cannot learn a feasible balancing policy. In such cases, the BAL strategy does not switch over to practice the walking task, therefore failing in terms of the total return and the walking distance.

The RNN switching strategy avoids the pitfall of the BAL strategy and always switches over to the walking task. Even though the idealized model of Leo is used for RNN training, our results demonstrate the reduction of falls for the whole $\pm 60\,\%$ range of torso mass perturbations. However, Leo appears to be less successful in walking, which is reflected in the smaller walking distance and return. There are two possible explanations of this effect which need to be studied in future. The first explanation is that the policy and the value function become slightly undertrained or overtrained as a result of applying the RNN. This effect is also observed in both TIM and BAL strategies with suboptimal task durations. The second explanation supported by the additional result presented in Appendix B.4 suggests that mixing transitions sampled from two different models of Leo – the upper-body model and the whole-body model – is detrimental for learning. Similarly, such incorrect samples affect TIM and BAL strategies.

Our studies do not reveal a significant difference between the 2- and 3-task curriculum learning, although the results in Table 6.4 and the cross-entropy method results suggest that both the upper-body balancing and the whole-body balancing tasks are useful. Also note that we use a conservative damage evaluation because damage due to the robot falls is treated equally across all tasks. In practice, the upper-body balancing task requires us to fix the robot in a standing position. Since the robot is fixed in its hips, it is possible to eliminate the damage due to falls by placing safety cushions around the robot at the height of its hips. In the 2-task curriculum learning, placing cushions is cumbersome because the robot is constantly moving. By neglecting falls during the upper-body balancing task, we expect to further reduce the number of falls by $3.2\,\%$ compared to the conservative evaluation presented in Table 6.4.

6.7 Conclusion

Inspired by infants learning to walk and by the rehabilitation process, we hypothesized that curriculum learning is a practical approach towards increasing the safety of locomotion learning by robots. We set up experiments on several dynamical systems and in several simulators, and demonstrate the correctness of this hypotheses.

The results suggest that the reduction of robot falls happens due to two reasons. First, the balancing task is easier than the walking task. Consequently, learning the balancing task is quicker and safer than learning the walking task. Second, the balancing task appears to precondition the control policy and to avert unsafe exploration actions during the walking task. This aspect was found useful only for Leo, which has a significantly heavier torso mass among the studied systems.

There are multiple ways of improving the current work. First, for the automated switching strategies, BAL and RNN, we kept all transitions in the replay buffer. We hypothesize that further improvement of their performance is possible by a smart bookkeeping and filtering of the transitions, cf. De Bruin et al. (2016).

Second, additional or new performance indicators can improve the RNN strategy to achieve a higher return and a more robust policy, e.g. Graves et al. (2017).

Third, it is possible to prevent forgetting the balancing policy when the walking task is being practiced, e.g. by using progressive networks (Rusu et al., 2017). By predicting states that are likely to lead to falls, one can actively switch between the already-acquired balancing policy and the immature walking policy, thereby providing a backup and constraining risky exploration. Increased training stability could be an additional benefit of using progressive networks because for each new task the capacity of the network is extended with untrained neurons. Training these neurons in the direction of the new task can be easier than the already pretrained ones.

Fourth, it is possible to learn the robot model concurrently with learning the policy and use it for directing the exploration of a new task. The starting task should be very safe to learn, e.g. in the case of Leo, learning of a sitting position can be initiated from the lying position. In the next task – possibly more complex and risky – the model can suggest safer exploration actions. Chaining several tasks in this manner can potentially lead to acquiring a precarious skill with less damage compared to direct learning.

Finally, for the simplicity of this work we used a trimmed version of the walking reward function for learning to balance. However, this approach is not generic for a larger set of tasks, and a new methodology should be developed.

7

Conclusions and future directions

This thesis presented several reinforcement learning approaches that use various forms of prior knowledge to increase the safety of learning on the real hardware. The improvement of safety was illustrated by means of numerical simulations and real experiments. This chapter presents the conclusions in reply to the research questions posed in Chapter 1 and provides possible directions for future research.

7.1 Conclusions

7.1.1 Influence of exploration strategies

In search for a better understanding of how exploration during learning influences the robot performance and the incurred damage , the following research question was addressed in Chapter 2:

How do exploration strategies affect the damage of a real robot?

Conflicting influence of exploration strategies was observed in multiple experiments. Learning the high-return policy with Ornstein-Uhlenbeck (OU) noise resulted in a large number of robot falls and substantial gearbox fatigue due to gear re-engagements. The high foot velocities exhibited by the policy right before heel-strike could be another potential source of gearbox damage. This damage was not taken into account by the reward function, therefore Leo would continue to suffer from high-energy heel-strike impacts even after learning.

Learning lower-return policies with other exploration strategies such as ϵ-greedy and ϵ-OU resulted in lower velocities before heel-strike. Such policies have less chances of damaging the gears. The gearbox fatigue was also reduced by the PADA strategy, which took only an action similar to the previous one. Unfortunately, this strategy could not react to the quickly changing states of the robot, and consequently Leo suffered from a large number of falls. The ϵ-OU strategy, which applied greedy actions distorted by occasional OU noise, reduced the number of falls but significantly increased gearbox fatigue. Interestingly, the number of falls was smaller compared to the popular ε-greedy strategy, while the other performance measures were similar. For this reason, it is recommended to use the ϵ-OU strategy instead of the ε-greedy one.

As in the presented results, the importance of the temporal exploration coherence was recently studied by Van Hoof et al. (2017b). The authors demonstrated a trade-off between the time-step-based and episode-based exploration in policy search RL. Although the authors did not provide a formal damage analysis, their results are comparable to the results in Chapter 2. For example, episode-based exploration, as with PADA, did not explore well, which resulted in a suboptimal performance and low actuation jerk. On the other hand, similarly to ϵ-PADA, the balanced exploration strategy proposed by the authors improved the performance and also increased the actuation jerk. Unfortunately, the authors did not experiment with complex unstable systems such as walking robots, which prevents a comparison of their exploration strategies with ours in terms of robot falls.

The important conclusion is the following: although the studied exploration strategies could reduce either gearbox fatigue or the number of robot falls, it is not possible to reduce both factors concurrently. Faster learning can be a viable solution. However, since learning from experience involves making mistakes, it is unreasonable to believe that faster learning can completely eliminate the damage. Therefore, a composite approach towards damage minimization is recommended in Section 7.2.1.

7.1.2 Safer learning with an approximate dynamical model

Chapters 3 to 5 addressed the following research question:

> How can an approximate model of the robot help to mitigate the risks of robot damage?

Chapter 3 established the ground basis for using the learning algorithms. In particular, it showed that RL outperformed optimal control (OC) and nonlinear model predictive control (NMPC) in robustness to large structural uncertainties. In Chapter 4, the instability in the sequential quadratic programming solver made it difficult to augment the existing model of Leo with a Coulomb friction model. Two learning approaches were proposed to mitigate the reality gap.

Combination of RL and NMPC

The first approach called Model-Plant Mismatch Learning (MPML) is described in Chapter 4. MPML uses model-free RL to correct the difference between the expected outcome of the NMPC, which uses an approximate system model, and the actual outcome of the real system. While standalone NMPC was not able to accomplish the task, in combination with RL it provided safety barriers to constrain dangerous exploration actions taken by MPML.

In this thesis, the combination of RL with the prior knowledge of the dynamical model, the constraints, and the model-based controller resulted in the safest learning approach. The success of the real experiment was largely due to the reduced hardware strain, even though the learning time was long. The presented result demonstrates that, in general, slower learning can be more attractive, as long as learning goals are achieved without damaging a robot. This observation creates a new insight and a practical solution towards learning on real systems, where stochastic observations and rewards often do not allow the use of learning rates as high as those used in simulated experiments. In non-stationary environments, however, the learning should be quicker than the changes of the system dynamics.

The benefits of the MPML approach are summarized below:

1. Depending on the severity of the model-plant mismatch, safety barriers allow learning with little risk.

2. The nominal controller does not need to be Markov[1], as long as RL is Markov with respect to the real system.

3. MPML provides a practical and useful adaptation mechanism to slowly changing non-stationary environments and life-long learning.

4. The learning outcome is predictable, i.e., when the mismatch is minimized, the trajectory of the controlled system becomes similar to the trajectory of the internal model controlled by the nominal controller.

[1] Nominal controller should be Markov with respect to the model.

One shortcoming of the MPML formulation is that it does not establish safety guarantees neither during nor after the learning. In contrast to MPML, the recent work of Ostafew et al. (2016) improves the internal model of NMPC by updating it using machine learning techniques. Their approach guarantees safety after learning if the unknown disturbance can be modeled as a Gaussian process. In addition, the necessity for the real-time computation of constraints limits the application of the approach to low-dimensional systems, although faster processors will mitigate this problem in the future.

Learning the difference model with RL

The model-based RL-only approach to compensating parametric and structural uncertainties was proposed in Chapter 5. The approach iteratively improves the model of Leo by learning the difference model, which reduces the reality gap. The approach does not require continuity of the system dynamics, objective function, or constraints , and is therefore very suitable for hybrid systems. However, there is a higher risk of damage compared to MPML due to two factors. First, there are no safety barriers that constrain the exploration of potentially dangerous states. Second, there are no restrictions imposed on the difference model in the parts of the state space where supervised-learning data is unavailable.

Independently from the work presented in Chapter 5, Chatzilygeroudis and Mouret (2017) proposed not only to learn the difference model, but also to update the parameters of the idealized model using the data from the reality gap. When idealized model is far from the true one, their experimental results demonstrate an improved performance compared to learning using the difference model only.

Recently, several other approaches of using the approximate model were proposed in literature. These approaches try to improve the robustness of policies using an ensemble of perturbed models. For example, Rajeswaran et al. (2017) maximizes the return associated with the worst-case model parameters. In contrast, Yu et al. (2017) and Clavera et al. (2018) learn a universal policy that performs optimally across all model parameters, therefore avoiding a robustness vs. performance trade-off. These approaches demonstrate robustness even to model parameters not included in the ensemble of models. Finally, the recent extension of the model-based PILCO method developed by Gal et al. (2016) and improved by Higuera et al. (2018) demonstrated the exceptionally quick learning properties for medium-sized control problems.

In conclusion, for a substantial damage minimization during learning, it is important either to have an approximate model of the robot and its surroundings or to learn these models concurrently with learning the control policy. This conclusion follows from the comparison of the model-based and model-free learning results presented in this thesis, and from the similar observations found in the aforementioned articles.

7.1.3 Safer learning without the approximate dynamical model

The third research question was formulated to find alternative solutions for safer learning when the approximate dynamical model is unavailable:

> Without having the model at hand, is it possible to reduce robot damage by sequencing learning tasks similar to the way humans learn gradually?

As a representative example, Chapter 6 studied whether learning to balance before learning to walk could reduce the number of robot falls. Curriculum learning demonstrated a reduced number of falls and a good performance of the final policy. To achieve similar performance, direct learning required less time, but incurred a significantly larger number of falls.

This longer-learning outcome is contradictory to other curriculum learning works, in which the curriculum speeds up learning, e.g. works by Asada et al. (1996); Heess et al. (2017); Andrychowicz et al. (2017). This contradiction arises from the nature of RL which implies that learning can succeed only when there is a chance of solving the task by random actions. Curriculum learning implemented in the cited articles increases the chances of solving a very difficult task because it starts with an easier task and shapes the control policy such that it can later solve a more complex task. In our case, the balancing task is auxiliary because walking skill does not directly follow from balancing skill, although a connection between the skills is known in the research field of passive dynamic walkers (Mochon and McMahon, 1980; Wisse, 2004). Learning this auxiliary balancing task required additional effort, and thus longer learning time. In the meanwhile, learning first to balance preconditioned policy parameters, which subsequently protected the robots against risky actions while learning to walk. In other words, the balancing policy can be viewed as a backup controller that, in the beginning of learning to walk, provides safety barriers similar to NMPC in Chapter 4. The difference with a NMPC-based backup controller is that the balancing policy is gradually substituted by the walking policy when more walking experience is added to the replay buffer.

The prior knowledge provided in the form of the tasks and their sequence did not hamper the final policy performance. This can be attributed to the choice of the tasks and their sequence.

In conclusion, the presented results reinforce the statement regarding the learning time made in the previous section. A longer learning time does not necessarily result in more damage. Even without a system model available, curriculum learning reduces damage to the robot and does not hamper the final policy performance. Note that the potential for robot damage becomes substantially lower when the approximate robot model is available, cf. Chapters 4 and 5.

7.2 Directions for future research

A number of open issues related to algorithms presented in this thesis are identified and listed below.

7.2.1 Composite approach towards damage minimization

The process of the robot Leo learning to walk resulted in frequent damage to hardware components (Schuitema, 2012). Since none of the studied exploration strate-

gies could reduce both the number of falls and the severity of gearbox damage, the composite approach towards damage minimization is recommended. The list of recommendations below is sorted with the most effective recommendation first:

- If the approximate model of Leo is available, the combination of the model-based and model-free controllers proposed in Chapter 4 can be used to speed up learning, reduce the number of robot falls, and potentially reduce gearbox fatigue. Since online model predictive control approaches towards hybrid dynamics are not available yet, a gradient-free optimization method can be used for deriving a suboptimal control from an approximate model of Leo, e.g. Nagabandi et al. (2017). Special care should be taken to ensure that the derived controller is Markov with respect to the model.

- Alternatively, a mental rehearsal can reduce the number of interactions with the real world, e.g. Chapter 5. The ensembles of slightly perturbed models can improve the robustness of a trained policy during the mental rehearsal (Yu et al., 2017; Rajeswaran et al., 2017; Clavera et al., 2018).

- To reduce the damage associated with the gearbox re-engagement, actions similar to the previous ones should be selected, e.g. using the PADA exploration strategy from Chapter 2. Another possibility of increasing the number of times the robot can withstand stress before failure is to use more durable materials, such as hardened steel instead of aluminum. Finally, compliant actuators integrated into the robot construction can reduce generic impact forces (Albu-Schaeffer et al., 2008).

- To reduce the severity of heel-strike impacts, high velocities of the feet before the contact should be penalized by a specifically designed reward function. An accelerometer or a lower-body model can be used for the velocity estimation purpose.

- Curriculum learning proposed in Chapter 6 can be used to reduce the number of robot falls that increase due to the PADA exploration strategy.

- To further reduce the number of robot falls, knowledge of the critical states that lead to falls can be used to constrain exploration or dynamically modify the exploration strategy, e.g. Mannucci et al. (2018). For walking robots with statically unstable poses, the information about the critical states should be propagated, for example by learning a separate failure function similar to the value function, but at a higher learning rate and at a coarser scale (Schuitema, 2012).

7.2.2 Future directions for safer reinforcement learning research

The following list contains suggestions for improving the presented work. Some points directly arise from the developed approaches, while the others suggest more fundamental research directions.

- It is important to establish guarantees on the performance and stability of the MPML approach. A possible direction towards stability can be to guarantee that NMPC can recover from any RL disturbance in a single step.

- It is equally important to establish guarantees on the learning performance of RL, e.g. of the difference model approach. The possible number of ways a physical system may fail increases with the growth of its degrees of freedom (DoFs). This problem becomes especially pronounced with deep neural network (DNN)-based representations, because they are highly sensitive to an intentionally manipulated inputs. In computer vision, Papernot et al. (2017) demonstrated that DNN are vulnerable to black-box adversarial attacks. Behzadan and Munir (2017) demonstrated similar results in RL. DNN are easily fooled by these attacks because they do not understand the nature of tasks they are solving. While there exist a number of brute-force methods to prevent such attacks, these attacks clearly show that DNN could not yet yield the level of abstraction required to understand the concepts behind the training data.

- In terms of safety, it would be useful to compare model-free RL approaches to those proposed in Chapters 4 and 5 but with a model learned from the experience, e.g. Deisenroth and Rasmussen (2011); Caarls and Schuitema (2016); Nagabandi et al. (2017). Since learning the model may result in longer learning times, it is important to consider if the benefits of safer – but longer – learning outweigh the repair costs of the hardware.

- A particularly interesting connection can be drawn between models and curriculum learning in the context of generalization. It is well-known that generalization improves when one uses models of the "right" capacity (Shalev-Shwartz and Ben-David, 2014). On the other hand, the model with a particular capacity is appropriate for solving a particular task for a particular system. Curriculum learning provides a framework for building richer and more abstract concepts based on those that are already known. As such, curriculum learning seems to be a suitable tool for linking the "right" model capacity with the difficulty of tasks and environments. If the system model is not available, curriculum learning coupled with the model-driven exploration can gradually learn the model of a system or policies resulting in damage, and possibly reduce the risk of damage.

- The first step in the direction proposed in the previous step, and the direct continuation of Chapter 6, is to gain safety-related information from the approximate model of Leo to achieve fail-safe learning. Can Leo learn walking with just a few falls, if the approximate model of the robot is provided? One promising solution to this problem is to learn state and action representations that would comprehend the structure imposed by physics.

  The state representation is usually learned from redundant visual or lidar observations (Jonschkowski and Brock, 2015). However, it would also be useful

to discover the state representation of denser observations such as measurements of angle and velocity.

Learning the action representations can facilitate transfer between robotic systems and tasks as long as these systems operate under the laws of physics.[2] Such representations may serve as an abstraction layer between the controller and the robot actuators. Here, one can consider possible parallels with computer vision, where the layers of DNN learn local features that merge into more complex structures at deeper layers. These features are universal and can be reused for training on new datasets or problems (Shelhamer et al., 2017). The pre-trained network imposes implicit constraints on neuron weights, thereby guiding the learning towards minima that support better generalization and faster convergence (Erhan et al., 2010).

The proposed state and action representations would facilitate safer learning even when applied to generic problems. Finding such representations would be a major scientific break-through in the upcoming years.

- This thesis showed that curriculum learning is very promising for reducing the damage of walking robots. This hypothesis was inspired by knowledge of infants' development and post-stroke rehabilitation. The important issue to consider is whether curriculum learning can be applied to other robots for the purpose of damage reduction. Selection of the relevant tasks – that is, reward functions – is another open issue, which needs to be addressed in the future.

- To address the previous point, one can simplify the problem by first automating the specification of a single reward function for solving a single task. This is itself a challenging problem, because the proper reward function encodes the essential understanding of the task and the system. This knowledge is usually provided either by human (Popov et al., 2017) or obtained from demonstrations using inverse RL (Abbeel and Ng, 2004). However, similar to the previous points, the laws of physics and spacial rules of geometry are uniform for all real systems. Therefore, robotic systems and tasks can share similarity beyond the currently considered penalties on the control signals and jerk. Developing approaches to account for such similarity in the reward function could be an exciting avenue to explore. These approaches initialized with physics-based knowledge could discover platform- and task-specific reward functions through self-learning, e.g. guided solely by the intrinsic motivation. In such a setting, human would provide only a goal-directed part of the reward.

- The behavior of the exploration strategies is generic and should hold for model-free or model-based, discrete or continuous RL algorithms as well as other physical systems. For example, results in Chapter 2 are obtained for the SARSA algorithm with discretized states and control actions. To implement this strategy for continuous states and actions, the actor-critic algorithm can

[2] This idea is related to dynamic movement primitives or hierarchical RL, although these representations are usually learned for a specific system and task.

learn the delta-action that is added to the previously applied action. Similarly to the PADA results, these bounded delta-actions will affect systems with unstable dynamics such as generic bipedal robots. It is expected that this strategy will increase the number of robot falls but reduce the damage due to the gearbox re-engagement. Van Hoof et al. (2017a) already explored this idea, although the reduction of damage by this strategy was neither calculated nor evaluated experimentally.

- Memorizing a backup controller learned from solving each task in the curriculum can provide an attractive enhancement to the basic approach described in this thesis. Such enhancement, e.g. achieved by means of progressive networks (Rusu et al., 2017), can provide a backup solution during the whole training time and not only in the beginning of learning. Potentially, this can further reduce the number of robot falls incurred during the learning.

- Suppressing the difference model in the parts of the state space, where supervised-learning data is unavailable, may yield steadier policy improvement by the algorithm proposed in Chapter 5.

Working with hardware is challenging. However, working with it is essential if one wants not only to solve practical robotic problems but also to gain new insights that might not be possible to perceive in simulated environments.

Acknowledgements

First and foremost, I would like to thank my promoters, Heike Vallery and Robert Babuška, for their continued support throughout my Ph.D. studies. Both Heike and Robert provided invaluable advice on many topics. They were always open for discussion due to which the research steadily progressed forward. No matter how busy they were, they always found the time to read the articles I was writing, provide constructive suggestions and keep a critical attitude towards the results. All this helped me enormously with sharpening ideas, developing new ones and finally publishing them. I would also like to thank Wouter Caarls whose expertize and knowledge of reinforcement learning has proven to be priceless. Although officially he was not my supervisor, he has committed to Skype meetings every other week, and we were communicating like that for four years almost without intermissions. I truly appreciate this mindset of Wouter. Wouter also provided many advise on the articles, taught me a structured and analytical approach towards science. This work would not have been possible without the encouragement and enthusiasm of Heike, Robert and Wouter. Also, I would like to thank Jens Kober and Tim de Bruin for their keen insights, ideas and help with fellowship proposal writing.

Second, I would like to thank all my colleagues – who also became my good friends – at TU Delft BioRobotics Lab. I really liked their curious minds and open hearts that welcomed me in the Netherlands and made my life there very pleasant. Joost van der Weijde and Erik Vlasblom, whom I met in the first days after my arrival in the Netherlands, became my close friends and also paranymphs of the Ph.D. defense ceremony. I absolutely enjoyed our countless and funny talks about science, food, Japan, Korea, Netherlands, and attitude towards life. The four years of research would be much more problematic without Gijs van der Hoorn who took the responsibility of administrating computing servers and ensured that all tasks were running smoothly (and usually until the end). Special thanks go to Andrew Berry, Patricia Baines and Pavlo Bazilinskyy who helped me to proofread and polish dissertation texts. I have learned a lot from their feedback.

Every year we had several activities in our lab. I will miss our summer BBQs at Heike's house, Christmas Feuerzangenbowle, and occasional sports events. But most of all, I will miss our daily routine of lunches, coffee breaks, and borrels that we had together with Mukunda Bharatheesha, Carlos Hernandez Corbato, Tim Vercruyssen, Daniel Lemus Perez, Laurens Valk, Jeff van Egmond, Michiel Plooij, Wouter Wolfslag, Saher Jabeen, Martin Klomp, Ruben Burger, and other people mentioned above. This was an indispensable time to learn something new about the Netherlands or other countries, share food, and receive much advice about studies and research.

I am also thankful to Erik Schuitema for developing the robot Leo that kept me

busy for four years. Many thanks go to Jan van Frankenhuyzen and Guus Liqui Lung who helped me with repairing the robot and maintaining it in good shape.

I would like to thank the department of BioMechanical Engineering and Graduate School of TU Delft for providing management and accounting support and for solving occasional problems. In particular, many thanks go to Nancy Kouters, Mirjam Bierhuizen, Hanneke Hustinx, Sabrina Ramos Rodrigues, and Mascha Toppenberg.

Many thanks go to my friends at TU Delft – Gleb Polevoy, Nikita Lenchenkov, Pavlo Bazilinskyy, Alexey Ilyushkin, Natalia Vtyurina, and others – for being there with me when I have had ups and downs which are inevitable for almost any Ph.D. student.

Last but not least, I would like to thank my parents, grandfather, and brother for all their support and love throughout my life.

Ivan Koryakovskiy
Moscow, October 2018

A

Experimental setups

A.1 Bipedal walking robot Leo

A bipedal robot Leo was designed by Erik Schuitema in 2010 at the Delft Biorobotics Lab (Schuitema, 2012). This section gives a description of the robot.

Leo is the 2D bipedal robot. It is shown in Figure A.1. The robot hardware is designed with the purpose of conducting hours of autonomous learning. Leo is small (50 cm) in size and light-weight (1.7 kg). Foam bumpers are attached on both sides of the torso top and between the hip motors to secure the robot from damage due to falls in the wide range of configurations. Additionally, elastic couplings are added for protection of built-in motor gearboxes. Leo is connected to a boom at its right hip with power provided through a freely rotating joint. The on-board embedded computer (VIA Eden 1.2GHz CPU and 1GB RAM) communicates over RS-485 serial ports with servo motors. Although it is possible to run a standalone real-time software on the embedded computer, an external computer is used for resource-intensive algorithms. The communication between the embedded and external computers is established over Ethernet using ZeroMQ library. While this communication channel obstructs from using the real-time software, in practice it does not pose a serious problem for learning.

Leo has seven motors, two for each hip, knee, and ankle, and the last one for a shoulder. A single arm facilitates preprogrammed stand up motion thereby delivering a completely autonomous learning platform. All joints and the torso-to-boom connection are equipped with magnetic encoders that provide angle measurements. The resulting state vector consists of the angles ϕ and angular velocities $\dot{\phi}$ of all joints and the torso-to-boom connection, as well as the feet contact indicators measured by the pressure sensors located at toes and heels of each foot. The angular velocities are calculated using the finite difference and second order Butterworth filters applied to the angular positions ϕ. The on-board operating system implements a fixed 30 ms sampling period of the state vector. Temperature measurements are provided every minute by the dedicated sensors located in each motor.

All seven motors are actuated in a voltage control mode with the maximum absolute voltage of 10.8 V. The original robot design adopted Dynamixel RX-28 motors which were used in Chapter 2. In the later chapters, these motors were upgraded to newer Dynamixel XM-430-W210-R version, which supports voltage (PWM) and torque control modes, have built-in magnetic encoders and a better heat transfer.

In this thesis, two simulators are used for the robot modeling. The first simulator developed by Schuitema (2012) is based on the Open Dynamics Engine (ODE) (Smith, 2011). The second simulator, Rigid Body Dynamics Library (RBDL) (Felis, 2017b,a), uses Articulated Body Algorithm by Featherstone (2008). The latter simulator has an advantage over ODE in that is allows to initialize Leo in an arbitrary state and extract an additional information such as the position and velocity of the center of mass, or the angular momentum. Both simulators use similar models of Leo and deliver similar results. Table A.1 summarizes geometric properties of the schematic model of Leo shown in Figure A.2.

The ODE and RBDL simulators require torques as input signals. The following simplified model for a DC motor with the gearbox is used to calculate the joint

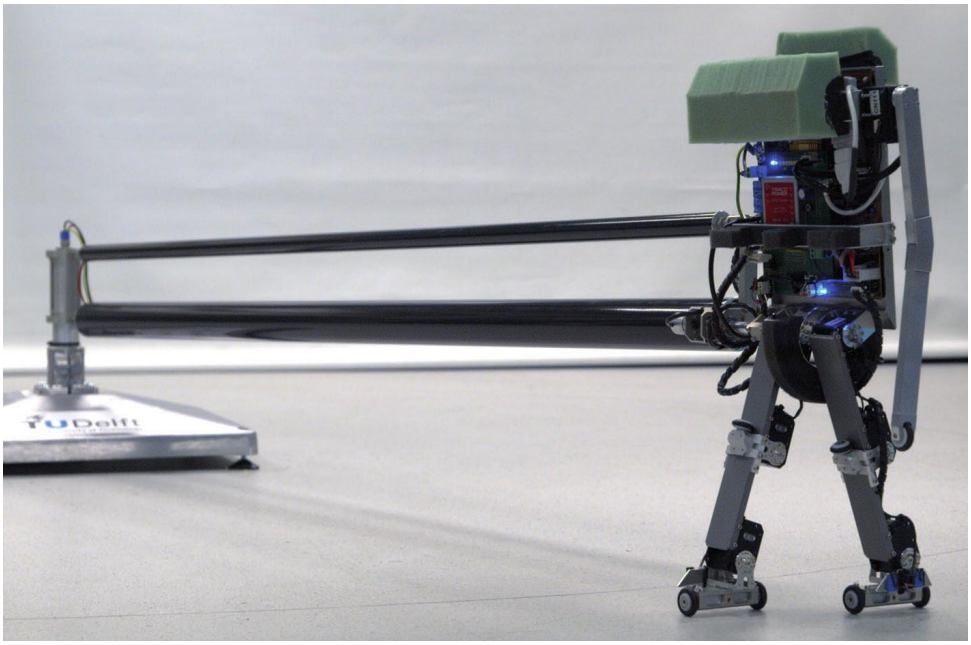


Figure A.1: A 2D bipedal walking robot Leo designed by Schuitema (2012) to conduct hours of learning experiments with a minimal human assistance. A boom enforces sideways stability, provides power to the robot and lets it walk in circles. A single arm facilitates preprogrammed stand up motion.

torque τ resulting from the motor voltage U:

$$\tau = K_\tau K_\mathrm{G} \frac{U - K_\tau K_\mathrm{G} \dot{\phi}}{R},$$

where $K_\tau = 6.33 \cdot 10^{-3}\,\mathrm{N\,m\,A^{-1}}$ is the motor's torque constant, $K_\mathrm{G} = 212.6$ is the gearbox ratio, and $R = 4.6\,\Omega$ is the winding resistance[1].

The difficulty of working with the real robot arises from its fragile hardware as well as model structure and parameter uncertainties. For instance, the experiments conducted by Schuitema (2012) revealed that the motor gearboxes are highly susceptible to the damage that arises due to the random exploration and the touchdown impacts experienced in certain robot states. In particular, the author showed that gearboxes broke every 30 min on average during the locomotion learning.

The model-plant mismatch arises due to the following model structure and parameter uncertainties. Note that the simulators do not account for these uncertainties.

- A large communication delay of 13.0 ± 1.7 ms, which comprises measurement, computation and actuation delays. This delay violates Markov property and significantly hampers the learning performance.

[1] These parameters are equivalent to the Dynamixel XM-430-W210-R motor parameters.

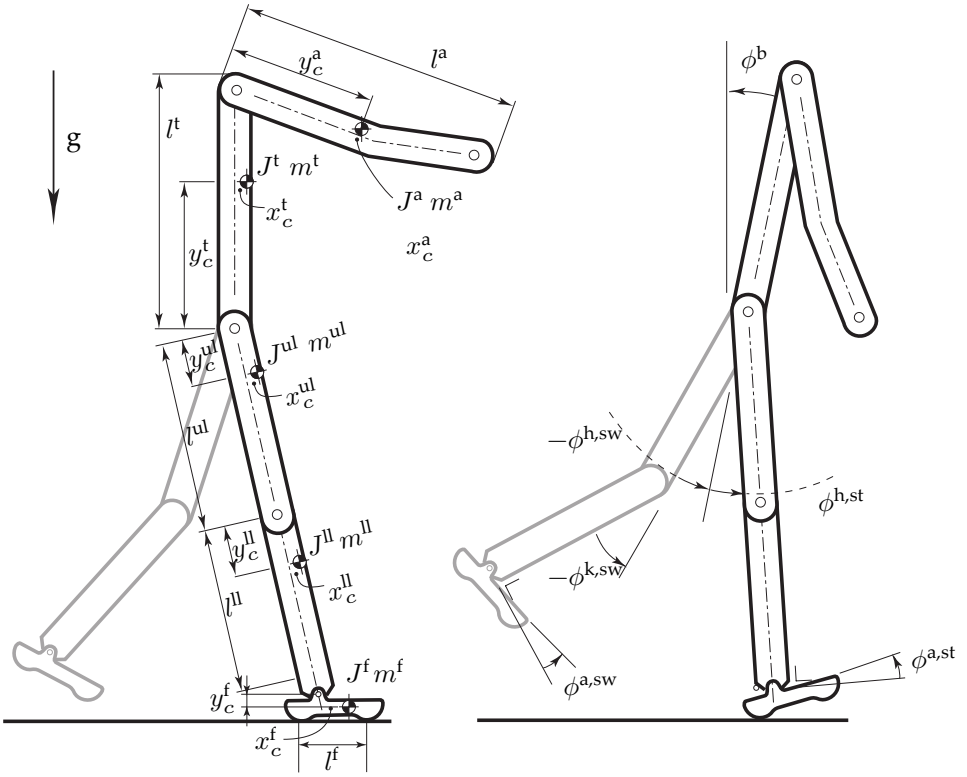


Figure A.2: Simulation model of Leo designed by Schuitema (2012). The left picture shows the parameter definitions with values given in Table A.1. The right picture shows the DoFs of the model.

- A Coulomb and viscous friction that depend on the motor temperature and the applied torque.
- The approximate value of the motor's torque constant.

A.2 The inverted pendulum on a movable cart

The cart-pendulum system is shown in Figure A.3. The system consists of a cart with mass m_M and a pendulum that is attached to the cart's center of mass c_M. The pendulum is a point mass m_m attached at the end of a massless rod of length l. The system has two DoFs, namely the linear motion of the cart along the x-axis, described here by the coordinate $s \in \mathbb{R}$, and the rotary motion of the pendulum with respect to the cart, described by the angle $\phi \in \mathbb{R}$. The only actuation is realized by a horizontal force $F_s \in \mathbb{R}$ acting on the cart body.

By summarizing the positions, velocities and accelerations in $\boldsymbol{q} = [s, \phi]^\top$, $\dot{\boldsymbol{q}} = [\dot{s}, \dot{\phi}]^\top$ and $\ddot{\boldsymbol{q}} = [\ddot{s}, \ddot{\phi}]^\top$, the forward dynamics are given by

$$\ddot{\boldsymbol{q}} = \left(\mathbf{H}(\boldsymbol{q})\right)^{-1} \left(\boldsymbol{F} - \boldsymbol{B}(\boldsymbol{q}, \dot{\boldsymbol{q}})\right),$$

Table A.1: Parameter values of the simulation model of Leo, where m is the mass, J is the moment of inertia, l is the joint length, y_c is the vertical center of mass offset, x_c is the horizontal center of mass offset of all body parts. The table is adapted from Schuitema (2012) to reflect changes in Leo after the motor upgrade.

	torso	torso + boom	upper leg	lower leg	foot	arm	boom
$m\,(\mathrm{kg})$	0.942	1.260	0.200	0.137	0.073	0.095	0.860
$J\,(\mathrm{g\,m^2})$	4.68	8.71	0.273	0.153	0.0488	0.873	319
$l\,(\mathrm{mm})$	212	212	116	109	81	−300	1700
$y_c\,(\mathrm{mm})$	131	97	63	61	31	148	0
$x_c\,(\mathrm{mm})$	−1	0	3	8	0	14	835

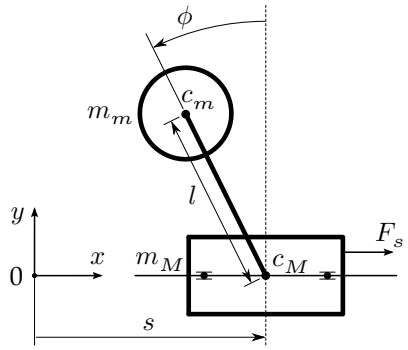


Figure A.3: The inverted pendulum on a movable cart.

where $\mathbf{H} \in \mathbb{R}^{2\times 2}$ is the system's mass matrix and $\boldsymbol{B}$ contains Coriolis, centrifugal, and gravitational terms and $\boldsymbol{F} = \left[F_s, \tau_\phi\right]^\top \in \mathbb{R}^2$ denotes the actual actuation consisting of the one for the cart and for the pendulum.

We use the RBDL (Felis, 2017b,a) for the efficient evaluation of the system's forward dynamics using the Articulated Body Algorithm by Featherstone (2008). The respective dynamic system in the form of an ordinary differential equation (3.1b) and initial values (3.3a) are retrieved from the forward dynamics.

For simulations, we use the following parameters:

$$m_M = 10.0\,\mathrm{kg}; \quad m_m = 1\,\mathrm{kg}; \quad l = 0.5\,\mathrm{m}; \quad 0\,\mathrm{s} \leq t \leq 5\,\mathrm{s}.$$

The cart and the pendulum are subject to simple constraints that enforce limits on the cart position and applicable force

$$-2.4\,\mathrm{m} \leq s(t) \leq 2.4\,\mathrm{m}, \quad -150\,\mathrm{N} \leq F_s(t) \leq 150\,\mathrm{N}.$$

In the experiment with unknown viscous friction coefficient, the internal torque in the rotary joint of the pendulum is $\tau_\phi = -\kappa\mu\dot{\phi}$, where κ is a coefficient that depends

on the configuration of the rotary joint, and μ is a viscous friction coefficient. We choose $\kappa = 1\,\mathrm{m}^2$, and vary μ in the range

$$0.0\,\mathrm{N\,s\,m^{-2}} \leq \mu \leq 0.2\,\mathrm{N\,s\,m^{-2}}.$$

B

Additional results

B.1 Influence of the reward shaping on the trajectory cost

The results of the effect of the shaping weight ψ on the cost (3.5) are presented in Figure B.1. According to the graph, for a shaping weight of up to 20, the total cost reduces, and then starts increasing again. Note that the total cost is calculated using (3.6), which does not include the shaping weight ψ. The error in position is more volatile, but one may notice that it gets smaller for higher shaping weights. We selected $\psi = 20$, because our primary goal was to reduce the total cost and accept a moderate steady-state error.

B.2 Influence of the discount rate on learning with MPML*

According to Corollary 1, MPML with the discount rate $\gamma > 0$ can minimize the mismatch better than with $\gamma = 0$. However, $\gamma > 0$ may hamper the learning performance because MPML with $\gamma > 0$ minimizes the expected future mismatch, which is more challenging computationally. To investigate the influence of γ, two additional experiments are performed for the multiple values of γ on two models of Leo.

The first model is the realistic model of Leo described in Section 4.5.1. The difference with the idealized model is mild, and the constraints in Section 4.5.2 are rather loose. Therefore, it is expected that for every state it is possible to find a control signal that eliminates the Coulomb friction. According to Corollary 1, MPML with $\gamma = 0$ can minimize the mismatch better than with $\gamma > 0$.

In the second model, the torsion spring of stiffness $K_{\text{hip}} = 3.50\,\text{N}\,\text{m}\,\text{rad}^{-1}$ is introduced in robot's hip joints. The spring results in the additional torque applied at the hips,

$$\Delta\tau_{\text{hip}} = -K_{\text{hip}}\left(\phi_{\text{hip}} - \bar{\phi}_{\text{hip}}\right),$$

where $\bar{\phi}_{\text{hip}} = 1.03\,\text{rad}$ corresponds to the hip angle when the robot reaches the upper setpoint. Although this modification does not apply to real Leo, it can imitate the stiffness of leg wires in other robots. The high stiffness forces RL to search for alternative means to minimize the mismatch because it is not possible to find such $\boldsymbol{u}^{\text{RL}}$ signal that eliminates the mismatch. According to Corollary 1, MPML with $\gamma > 0$ should minimize the mismatch better than with $\gamma = 0$.

Figure B.2a shows simulation results on the first model. As expected, $\gamma = 0$ achieves the smallest mismatch, which almost monotonically increases with γ. A different picture is seen in Figure B.2b for the hip springs model. It turns out that $\gamma \in [0.5, 0.6]$ finds solutions with lower mismatch and the number of squats closer to the idealized model. Surprisingly, $\gamma = 0$ also performs relatively well.

The smaller values of γ break the ties between the present and future mismatch, therefore, allowing a more efficient mismatch minimization. However, this

* This appendix is not a part of the published article.

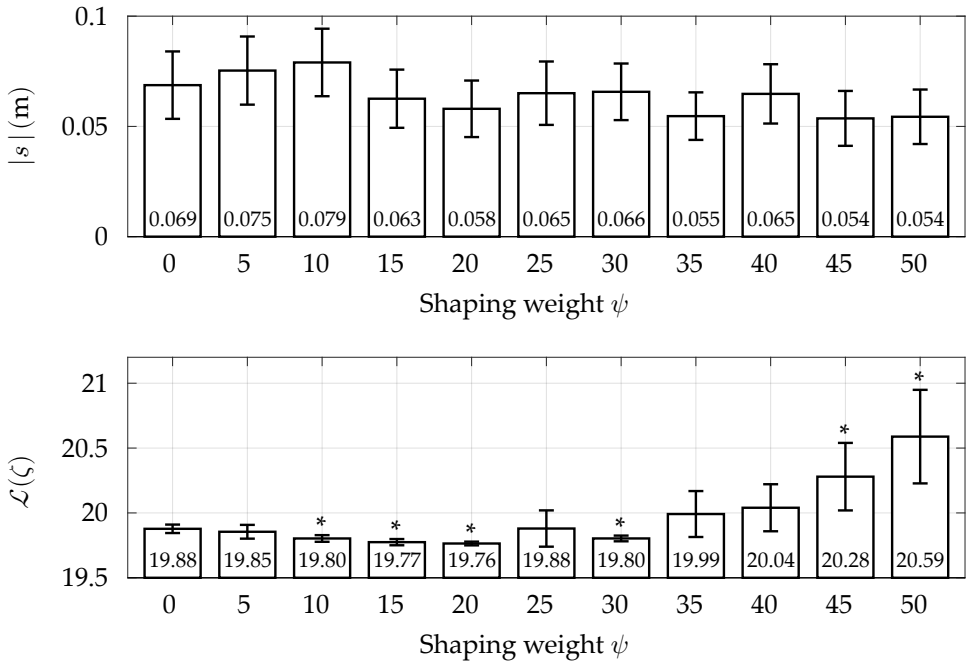


Figure B.1: Influence of the shaping function on the results of the "RL" method. The *top* plot shows the absolute error in the cart position at the end of an episode depending on the weight ψ of the shaping function. The *bottom* plot shows the total cost of the trajectory. Shaping is not used for $\psi = 0$. The numbers inside of the bars show the mean value of the error averaged over 50 independent runs, while the error bars show the upper and lower 95% confidence limits. The statistically significant result, for which the p-value is less then 0.05, is marked with $*$ above the bars.

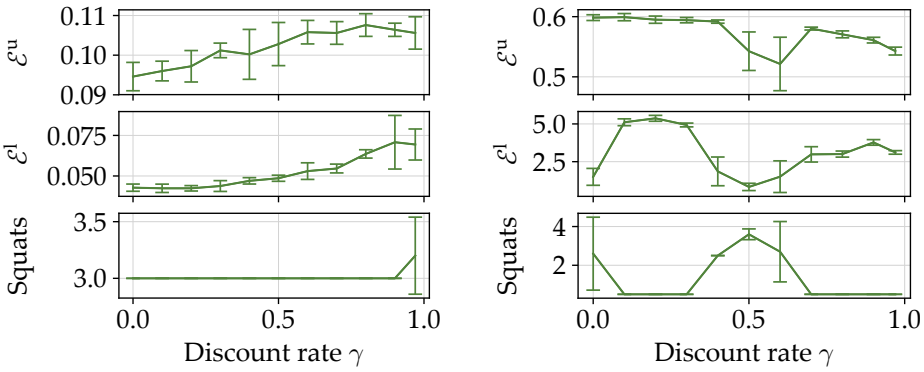
requires the possibility of reaching zero mismatch for every state of the idealized trajectory. Three situations of when this is not possible are listed below.

1. Intended cautiousness in the face of uncertainty forced by the tight constraints on $\boldsymbol{u}^{\mathrm{RL}}$.

2. The model-plant mismatch is such that idealized NMPC state-space trajectory is not realizable on a real system.

3. Large policy approximation errors, which may happen due to the regularization of policy parameters or a chosen function approximator.

In such situations, using $\gamma > 0$ is preferable.



(a) Coulomb friction (b) Hip springs

Figure B.2: These plots illustrate the results of MPML after learning on the Leo model subject to Coulomb friction (*a*) and on the model with added hip springs (*b*). Each subplot shows the model-plant mismatch accumulated during reaching the upper (*top*) and the lower (*middle*) setpoints, and the number of squats (*bottom*). Means with upper and lower 95% confidence limits are shown for five runs.

B.3 The mass distribution of Leo and Roboschool systems vs. human

Table B.1 compares the ratios of the upper-body mass to the total body mass for the studied systems vs. the human body. The presented result reveals that Leo's body is very similar to the human body. On the other hand, all Roboschool systems (Hopper, Halfcheetah, Walker2d) have a very different distribution of mass, with Walker2d having the least similarity to the human body.

B.4 Curriculum learning with samples obtained from different models

Figure B.3 compares the 3-task curriculum learning performance in two cases. In the first case, the upper-body balancing samples are removed from the replay buffer after the first task is completed (PV+PV·RB). In the second case, the samples are kept in the reply buffer until the end of learning (PV·RB+PV·RB). It can be seen that the RB option appears to have a negative influence on the total reward, the number of robot falls, and the walking distance of Leo. The changing system dynamics between the upper-body and whole-body balancing tasks can explain this outcome. Since the RB option improves the performance of the 2-task curriculum, we conclude that only the upper-body balancing samples are detrimental for learning on the whole-body model.

Table B.1: This table compares the mass distribution of the studied systems vs. human. The masses of the Roboschool systems are calculated directly from the Bullet Physics Engine.

System	Upper-body mass to the total body mass	Reference
Human	0.678	Winter (2009)
Leo	0.642	Schuitema (2012)
Hopper	0.232	
Halfcheetah	0.248	
Walker2d	0.155	

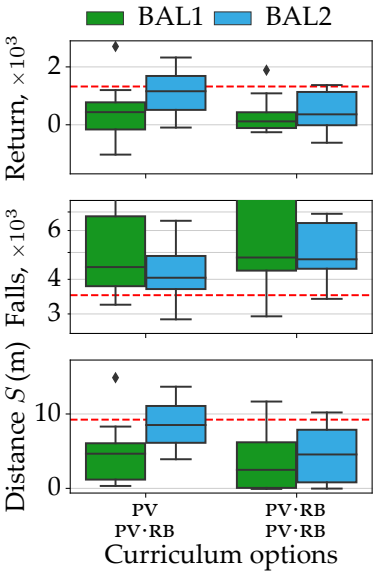


Figure B.3: Three-task curriculum learning results on Leo *(left group)* without the replay buffer transfer from the upper-body balancing task to the whole-body balancing task, and *(right group)* with the replay buffer transfer. The RB option appears to have a negative influence on the total reward and the walking distance of Leo. The optimal performance of the TIM strategy is shown by the *red dashed* line, cf. Figure 6.7.

Bibliography

Abbeel, P., Coates, A., and Ng, A. Y. (2010). Autonomous Helicopter Aerobatics Through Apprenticeship Learning. *International Journal of Robotics Research*, 29(13):1608–1639.

Abbeel, P. and Ng, A. Y. (2004). Apprenticeship Learning via Inverse Reinforcement Learning. In *International Conference on Machine Learning*, page 1.

Abbeel, P., Quigley, M., and Ng, A. Y. (2006). Using Inaccurate Models in Reinforcement Learning. In *International Conference on Machine Learning*, pages 1–8.

Adolph, K. and Robinson, S. (2013). The Road to Walking: What Learning to Walk Tells Us About Development. *Oxford handbook of developmental psychology*, 1:403–443.

Adolph, K. E., Berger, S. E., and Leo, A. J. (2011). Developmental Continuity? Crawling, Cruising, and Walking. *Developmental science*, 14(2):306–318.

Albu-Schaeffer, A., Eiberger, O., Grebenstein, M., Haddadin, S., Ott, C., Wimbock, T., Wolf, S., and Hirzinger, G. (2008). Soft Robotics: From Torque Feedback Controlled Lightweight Robots to Intrinsically Compliant Systems. *IEEE Robotics and Automation Magazine*, 15:20–30.

Albus, J. S. (1975). A New Approach to Manipulator Control: the Cerebellar Model Articulation Controller (CMAC). *Journal of Dynamic Systems, Measurement, and Control*, 97(3):220–227.

Alpaydın, E. (2014). *Introduction to Machine Learning*. MIT Press, 3 edition.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2017). Hindsight Experience Replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058.

Asada, M., Noda, S., Tawaratsumida, S., and Hosoda, K. (1996). Purposive Behavior Acquisition for a Real Robot by Vision-Based Reinforcement Learning. *Machine Learning*, 23(2):279–303.

Bachrach, A., He, R., and Roy, N. (2009). Autonomous Flight in Unknown Indoor Environments. *International Journal of Micro Air Vehicles*, 1(4):217–228.

Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., and Mordatch, I. (2017). Emergent Complexity via Multi-Agent Competition. arXiv:1710.03748.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):834–846.

Bayiz, Y. E. and Babuska, R. (2014). Nonlinear Disturbance Compensation and Reference Tracking via Reinforcement Learning With Fuzzy Approximators. *IFAC Proceedings Volumes*, 47(3):5393 – 5398.

Behzadan, V. and Munir, A. (2017). Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. In *Machine Learning and Data Mining in Pattern Recognition*, pages 262–275.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum Learning. In *International Conference on Machine Learning*, pages 41–48.

Berger, S. E., Theuring, C., and Adolph, K. E. (2007). How and When Infants Learn to Climb Stairs. *Infant Behavior and Development*, 30(1):36 – 49.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. (2009). Natural Actor-Critic Algorithms. TR09-10, University of Alberta, Canada.

Biegler, L. T. (2013). A Survey on Sensitivity-based Nonlinear Model Predictive Control. In *IFAC International Symposium on Dynamics and Control of Process Systems*, pages 499–510.

Bock, H. G., Diehl, M., Kostina, E. A., and Schlöder, J. P. (2007). Constrained Optimal Feedback Control of Systems Governed by Large Differential Algebraic Equations. In Biegler, L., Ghattas, O., Heinkenschloss, M., Keyes, D., and van Bloemen Waanders, B., editors, *Real-Time PDE-Constrained Optimization*, chapter 1, pages 3–24. SIAM.

Bock, H. G. and Plitt, K. J. (1984). A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems. In *IFAC World Congress*, pages 242–247.

Botev, Z. I., Kroese, D. P., Rubinstein, R. Y., and L'Ecuyer, P. (2013). The Cross-Entropy Method for Optimization. In *Handbook of statistics*, volume 31, pages 35–59.

de Bruin, T., Kober, J., Tuyls, K., and Babuska, R. (2016). Improved Deep Reinforcement Learning for Robotics Through Distribution-Based Experience Retention. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3947–3952.

Caarls, W. (2015). Generic Reinforcement Learning Library. `https://github.com/wcaarls/grl`.

Caarls, W. and Schuitema, E. (2016). Parallel Online Temporal Difference Learning for Motor Control. *IEEE Transactions on Neural Networks and Learning Systems*, 27(7):1457–1468.

Calandra, R., Seyfarth, A., Peters, J., and Deisenroth, M. P. (2016). Bayesian Optimization for Learning Gaits Under Uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23.

Cardona, G. A., Moreno, W., Weitzenfeld, A., and Calderon, J. M. (2016). Reduction of Impact Force in Falling Robots Using Variable Stiffness. In *IEEE SoutheastCon*, pages 1–6.

Chatzilygeroudis, K. I. and Mouret, J.-B. (2017). Using Parameterized Black-Box Priors to Scale Up Model-Based Policy Search for Robotics. arXiv:1709.06917.

Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., Abbeel, P., and Zaremba, W. (2016). Transfer From Simulation to Real World Through Learning Deep Inverse Dynamics Model. arXiv:1610.03518.

Clavera, I., Nagabandi, A., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. (2018). Learning to Adapt: Meta-Learning for Model-Based Control. arXiv:1803.11347.

Cutler, M., Walsh, T. J., and How, J. P. (2014). Reinforcement Learning With Multi-Fidelity Simulators. In *IEEE International Conference on Robotics and Automation*, pages 3888–3895.

Datta, A. and Xing, L. (1998). The Theory and Design of Adaptive Internal Model Control Schemes. In *American Control Conference*, volume 6, pages 3677–3684.

Deisenroth, M. P., Fox, D., and Rasmussen, C. E. (2015). Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423.

Deisenroth, M. P. and Rasmussen, C. E. (2011). PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *International Conference on Machine Learning*.

Diehl, M. (2001). *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Heidelberg University.

Diehl, M., Bock, H. G., and Schlöder, J. P. (2005). A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking Deep Reinforcement Learning for Continuous Control. In *International Conference on Machine Learning*, pages 1329–1338.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Endo, G., Morimoto, J., Matsubara, T., Nakanishi, J., and Cheng, G. (2008). Learning CPG-based Biped Locomotion with a Policy Gradient Method: Application to a Humanoid Robot. *International Journal of Robotics Research*, 27(2):213–228.

Engel, J.-M. and Babuska, R. (2014). On-line Reinforcement Learning for Nonlinear Motion Control: Quadratic and Non-Quadratic Reward Functions. In *IFAC World Congress*, volume 19, pages 7043–7048.

Erez, T., Lowrey, K., Tassa, Y., Kumar, V., Kolev, S., and Todorov, E. (2013). An Integrated System for Real-Time Model Predictive Control of Humanoid Robots. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 292–299.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why Does Unsupervised Pre-Training Help Deep Learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Ernst, D., Glavic, M., Capitanescu, F., and Wehenkel, L. (2009). Reinforcement Learning Versus Model Predictive Control: A Comparison on a Power System Problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(2):517–529.

Farchy, A., Barrett, S., MacAlpine, P., and Stone, P. (2013). Humanoid Robots Learning to Walk Faster: From the Real World to Simulation and Back. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pages 39–46.

Farshidian, F., Neunert, M., and Buchli, J. (2014). Learning of Closed-Loop Motion Control. In *International Conference on Intelligent Robots and Systems*, pages 1441–1446.

Featherstone, R. (2008). *Rigid Body Dynamics Algorithms*. Kluwer international series in engineering and computer science: Robotics. Springer.

Felis, M. (2012-2017a). Rigid Body Dynamics Library (RBDL). `https://bitbucket.org/rbdl/rbdl`.

Felis, M. L. (2017b). RBDL: an Efficient Rigid-Body Dynamics Library using Recursive Algorithms. *Autonomous Robots*, 41(2):495–511.

Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. (2015). Learning Visual Feature Spaces for Robotic Manipulation With Deep Spatial Autoencoders. *arXiv:1509.06113*.

Fisac, J. F., Akametalu, A. K., Zeilinger, M. N., Kaynama, S., Gillula, J. H., and Tomlin, C. J. (2017). A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems. arXiv:1705.01292.

Frasch, J. V., Wirsching, L., Sager, S., and Bock, H. G. (2012). Mixed–Level Iteration Schemes for Nonlinear Model Predictive Control. *IFAC Proceedings Volumes*, 45(17):138 – 144.

Fulton, N. and Platzer, A. (2018). Safe Reinforcement Learning via Formal Methods. In *AAAI Conference on Artificial Intelligence*.

Gal, Y., McAllister, R., and Rasmussen, C. E. (2016). Improving PILCO With Bayesian Neural Network Dynamics Models. In *Data-Efficient Machine Learning Workshop at International Conference on Machine Learning*.

Gamboa Higuera, J. C., Meger, D., and Dudek, G. (2017). Adapting Learned Robotics Behaviours Through Policy Adjustments. In *IEEE International Conference on Robotics and Automation*.

Garcia, J. and Fernandez, F. (2015). A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16:1437–1480.

Gehring, C. and Precup, D. (2013). Smart Exploration in Reinforcement Learning Using Absolute Temporal Difference Errors. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1037–1044.

Geibel, P. and Wysotzki, F. (2011). Risk-Sensitive Reinforcement Learning Applied to Control under Constraints. *arXiv:1109.2147*.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. (2017). Automated Curriculum Learning for Neural Networks. In *International Conference on Machine Learning*, volume 70, pages 1311–1320.

Grondman, I., Vaandrager, M., Busoniu, L., Babuska, R., and Schuitema, E. (2012). Efficient Model Learning Methods for Actor-Critic Control. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(3):591–602.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). Deep Reinforcement Learning for Robotic Manipulation With Asynchronous Off-Policy Updates. In *IEEE International Conference on Robotics and Automation*, pages 3389–3396.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous Deep Q-Learning With Model-Based Acceleration. In *International Conference on Machine Learning*, pages 2829–2838.

Ha, S. and Liu, C. K. (2015). Multiple Contact Planning for Minimizing Damage of Humanoid Falls. *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Ha, S. and Yamane, K. (2015). Reducing Hardware Experiments for Model Learning and Policy Optimization. In *IEEE International Conference on Robotics and Automation*.

Hanna, J. P. and Stone, P. (2017). Grounded Action Transformation for Robot Learning in Simulation. In *AAAI*, pages 3834–3840.

Hans, A., Schneegass, D., Schafer, A. M., and Udluft, S. (2008). Safe Exploration for Reinforcement Learning. In *European Symposium on Artificial Neural Networks*, pages 143–148.

Hansen, N. and Ostermeier, A. (2001). Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195.

Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S. M. A., Riedmiller, M. A., and Silver, D. (2017). Emergence of Locomotion Behaviours in Rich Environments. arXiv:1707.02286.

Heess, N., Wayne, G., Tassa, Y., Lillicrap, T. P., Riedmiller, M. A., and Silver, D. (2016). Learning and Transfer of Modulated Locomotor Controllers. arXiv:1610.05182.

Heger, M. (1994). Consideration of Risk in Reinforcement Learning. In *Machine Learning Proceedings*, pages 105 – 111.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2017). Deep Reinforcement Learning that Matters. arXiv:1709.06560.

Herdt, A., Diedam, H., Wieber, P.-B., Dimitrov, D., Mombaur, K., and Diehl, M. (2010). Online Walking Motion Generation with Automatic Foot Step Placement. *Special Issue: Section Focused on Cutting Edge of Robotics in Japan*, 24(5-6):719–737.

Hester, T., Quinlan, M., and Stone, P. (2012). RTMBA: A Real-Time Model-Based Reinforcement Learning Architecture for Robot Control. In *IEEE International Conference on Robotics and Automation*, pages 85–90.

Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. (2017). DARLA: Improving Zero-Shot Transfer in Reinforcement Learning. In *International Conference on Machine Learning*, volume 70, pages 1480–1490.

Higuera, J. C. G., Meger, D., and Dudek, G. (2018). Synthesizing Neural Network Controllers With Probabilistic Model-Based Reinforcement Learning. arXiv:1803.02291.

van Hoof, H., Neumann, G., and Peters, J. (2017a). Non-parametric Policy Search with Limited Information Loss. *Journal of Machine Learning Research*, 18(73):1–46.

van Hoof, H., Tanneberg, D., and Peters, J. (2017b). Generalized Exploration in Policy Search. *Machine Learning*, 106(9):1705–1724.

Hopkins, B. and Westra, T. (1990). Motor Development, Maternal Expectations, and the Role of Handling. *Infant Behavior and Development*, 13(1):117 – 122.

Howard, R. A. and Matheson, J. E. (1972). Risk-Sensitive Markov Decision Processes. *Management Science*, 18(7):356–369.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167.

Jakobi, N. (1998). Running Across the Reality Gap: Octopod Locomotion Evolved in a Minimal Simulation. In *European Workshop on Evolutionary Robotics (EvoRobot)*, pages 39–58.

Janka, D., Kirches, C., Sager, S., and Wächter, A. (2016). An SR1/BFGS SQP Algorithm for Nonconvex Nonlinear Programs with Block-Diagonal Hessian Matrix. *Mathematical Programming Computation*.

Jazwinski, A. H. (2007). *Stochastic Processes and Filtering Theory*. Dover Books on Electrical Engineering Series. Dover Publications.

Jiang, N., Kulesza, A., Singh, S., and Lewis, R. (2015). The Dependence of Effective Planning Horizon on Model Accuracy. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 1181–1189.

Johnson, T., Kirches, C., and Wächter, A. (2015). An Active-Set Quadratic Programming Method Based On Sequential Hot-Starts. *SIAM Journal on Optimization*, 25(2):967–994.

Jonschkowski, R. and Brock, O. (2015). Learning State Representations with Robotic Priors. *Autonomous Robots*, 39(3):407–428.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4(1):237–285.

Kakade, S. M. (2002). A Natural Policy Gradient. In *Advances in Neural Information Processing Systems*, pages 1531–1538.

Kamthe, S. and Deisenroth, M. P. (2017). Data-Efficient Reinforcement Learning With Probabilistic Model Predictive Control. arXiv:1706.06491.

Karpathy, A. and van de Panne, M. (2012). Curriculum Learning for Motor Skills. In *Advances in Artificial Intelligence*, pages 325–330.

Kawato, M. (1990). Feedback-Error-Learning Neural Network for Supervised Motor Learning. *Advanced Neural Computers*, 6(3):365–372.

Kimura, H. and Kobayashi, S. (1999). Stochastic Real-Valued Reinforcement Learning to Solve a Nonlinear Control Problem. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 510–515.

Kirches, C., Wirsching, L., Bock, H. G., and Schlöder, J. P. (2012). Efficient Direct Multiple Shooting for Nonlinear Model Predictive Control on Long Horizons. *Journal of Process Control*, 22(3):540–550.

Kober, J., Bagnell, J. A. D., and Peters, J. (2013). Reinforcement Learning in Robotics: A Survey. *International Journal of Robotics Research*.

Kober, J. and Peters, J. (2011). Policy Search for Motor Primitives in Robotics. *Machine Learning*, 84(1-2):171–203.

Kohl, N. and Stone, P. (2004). Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2619–2624 Vol.3.

Koryakovskiy, I., Vallery, H., Babuska, R., and Caarls, W. (2017). Evaluation of Physical Damage Associated with Action Selection Strategies in Reinforcement Learning. *IFAC-PapersOnLine*, 50(1):6928 – 6933.

Kudruss, M., Koryakovskiy, I., Vallery, H., Mombaur, K., and Kirches, C. (2018). Combining Multi-Level Real-Time Iterations of Nonlinear Model Predictive Control to Realize Squatting Motions on Leo. Technical Report 6425, Optimization Online.

Kühl, P., Diehl, M., Kraus, T., Schlöder, J. P., and Bock, H. G. (2011). A Real-Time Algorithm for Moving Horizon State and Parameter Estimation. *Computers & Chemical Engineering*, 35(1):71–83.

Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., and Tedrake, R. (2015). Optimization-based Locomotion Planning, Estimation, and Control Design for the Atlas Humanoid Robot. *Autonomous Robots*, pages 1–27.

Kumar, M. P., Packer, B., and Koller, D. (2010). Self-Paced Learning for Latent Variable Models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.

Lange, S. (2010). *Tiefes Reinforcement Lernen auf Basis Visueller Wahrnehmungen*. PhD thesis, Universitat Osnabruck.

Lange, S. and Riedmiller, M. (2010). Deep Auto-Encoder Neural Networks in Reinforcement Learning. In *International Joint Conference on Neural Networks*, pages 1–8.

Leineweber, D. B., Bauer, I., Bock, H. G., and Schlöder, J. P. (2003a). An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization. Part I: Theoretical Aspects. *Computers & Chemical Engineering*, 27(2):157–166.

Leineweber, D. B., Schäfer, A., Bock, H. G., and Schlöder, J. P. (2003b). An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization: Part II: Software Aspects and Applications. *Computers & Chemical Engineering*, 27(2):167–174.

Levine, S. and Koltun, V. (2013). Guided Policy Search. In *International Conference on Machine Learning*, pages 1–9.

Levine, S., Sampedro, P. P., Krizhevsky, A., Ibarz, J., and Quillen, D. (2017). Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous Control with Deep Reinforcement Learning. arXiv:1509.02971.

Lin, L.-J. (1993). Reinforcement Learning for Robots Using Neural Networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

Ljung, L., editor (1999). *System Identification: Theory for the User*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2 edition.

Mannucci, T., van Kampen, E. J., de Visser, C., and Chu, Q. (2018). Safe Exploration Algorithms for Reinforcement Learning Controllers. *IEEE Transactions on Neural Networks and Learning Systems*, 29(4):1069–1081.

Marco, A., Hennig, P., Bohg, J., Schaal, S., and Trimpe, S. (2016). Automatic LQR Tuning Based on Gaussian Process Global Optimization. In *IEEE International Conference on Robotics and Automation*.

Matignon, L., Laurent, G., and Le Fort-Piat, N. (2006). Reward Function and Initial Values: Better Choices for Accelerated Goal-Directed Reinforcement Learning. In *International Conference on Artificial Neural Networks*, volume 4131, pages 840–849.

Matsubara, T., Morimoto, J., Nakanishi, J., Sato, M.-a., and Doya, K. (2006). Learning CPG-Based Biped Locomotion With a Policy Gradient Method. *Robotics and Autonomous Systems*, 54(11):911 – 920.

Meijdam, H. J., Plooij, M., and Caarls, W. (2013). Learning While Preventing Mechanical Failure Due to Random Motions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 182–187.

Mihatsch, O. and Neuneier, R. (2002). Risk-Sensitive Reinforcement Learning. *Machine Learning*, 49(2-3):267–290.

Miyamoto, H., Gandolfo, F., Gomi, H., Schaal, S., Koike, Y., Osu, R., Nakano, E., Wada, Y., and Kawato, M. (1995). A Kendama Learning Robot Based on a Dynamic Optimization Theory. In *IEEE International Workshop on Robot and Human Communication*, pages 327–332.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., and others (2015). Human-Level Control Through Deep Reinforcement Learning. *Nature*, 518(7540):529.

Mochon, S. and McMahon, T. A. (1980). Ballistic Walking. *Journal of Biomechanics*, 13(1):49–57.

Moldovan, T. M. and Abbeel, P. (2012). Safe Exploration in Markov Decision Processes. In *International Conference on Machine Learning*, pages 1711–1718.

Mori, T., Nakamura, Y., Sato, M. A., and Ishii, S. (2004). Reinforcement Learning for a CPG-Driven Biped Robot. *American Association for Artificial Intelligence*, pages 623–630.

Morimioto, J., Zeglin, G., and Atkeson, C. G. (2003). Minimax Differential Dynamic Programming: Application to a Biped Walking Robot. In *SICE Annual Conference*, volume 3, pages 2310–2315.

Morimoto, J., Cheng, G., Atkeson, C. G., and Zeglin, G. (2004). A Simple Reinforcement Learning Algorithm For Biped Walking. In *International Conference on Robotics and Automation*, pages 3030–3035.

Muske, K. R., Rawlings, J. B., and Lee, J. H. (1993). Receding Horizon Recursive State Estimation. *American Control Conference*, 30:900 – 904.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. (2017). Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. arXiv:1708.02596.

Ng, A. Y., Harada, D., and Russell, S. J. (1999). Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *International Conference on Machine Learning*, pages 278–287.

Ostafew, C. J., Schoellig, A. P., and Barfoot, T. D. (2016). Robust Constrained Learning-Based NMPC Enabling Reliable Mobile Robot Path Tracking. *International Journal of Robotics Research*, 35(13):1547–1563.

Oßwald, S., Hornung, A., and Bennewitz, M. (2010). Learning Reliable and Efficient Navigation With a Humanoid. In *IEEE International Conference on Robotics and Automation*, pages 2375–2380.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical Black-Box Attacks Against Machine Learning. In *ACM on Asia Conference on Computer and Communications Security*, pages 506–519.

Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. (2009). Learning and Generalization of Motor Skills by Learning from Demonstration. In *IEEE International Conference on Robotics and Automation*, pages 1293–1298.

Peters, J., Mülling, K., and Altun, Y. (2010). Relative Entropy Policy Search. In *National Conference on Artificial Intelligence*, pages 1607–1612.

Peters, J. and Schaal, S. (2008). Reinforcement Learning of Motor Skills With Policy Gradients. *Neural Networks*, 21(4):682–697.

Popov, I., Heess, N., Lillicrap, T., Hafner, R., Barth-Maron, G., Vecerik, M., Lampe, T., Tassa, Y., Erez, T., and Riedmiller, M. (2017). Data-Efficient Deep Reinforcement Learning for Dexterous Manipulation. *arXiv:1704.03073*.

Qin, S. J. and Badgwell, T. A. (2003). A Survey of Industrial Model Predictive Control Technology. *Control Engineering Practice*, 11(7):733–764.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. (2017). EPOpt: Learning Robust Neural Network Policies Using Model Ensembles. In *International Conference on Learning Representations*.

Rubinstein, R. Y. (1997). Optimization of Computer Simulation Models with Rare Events. *European Journal of Operational Research*, 99(1):89 – 112.

Rusu, A. A., Večerík, M., Rothörl, T., Heess, N., Pascanu, R., and Hadsell, R. (2017). Sim-to-Real Robot Learning from Pixels with Progressive Nets. In *Annual Conference on Robot Learning*, volume 78, pages 262–270.

Sadeghi, F. and Levine, S. (2016). (CAD)2RL: Real Single-Image Flight Wihtout a Single Real Image. *arXiv:1611.04201*.

Saveriano, M., Yin, Y., Falco, P., and Lee, D. (2017). Data-Efficient Control Policy Search Using Residual Dynamics Learning. In *International Conference on Intelligent Robots and Systems*.

Schmid, K., Tomic, T., Ruess, F., Hirschmüller, H., and Suppa, M. (2013). Stereo Vision Based Indoor/Outdoor Navigation for Flying Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3955–3962.

Schork, L. (2015). A Parametric Active Set Method for General Quadratic Programming. Master thesis, Heidelberg University.

Schuitema, E. (2012). *Reinforcement Learning on Autonomous Humanoid Robots*. PhD thesis, TU Delft.

Schuitema, E., Hobbelen, D., Jonker, P., Wisse, M., and Karssen, J. (2005). Using a Controller Based on Reinforcement Learning for a Passive Dynamic Walking Robot. In *IEEE-RAS International Conference on Humanoid Robots*, pages 232–237.

Schuitema, E., Wisse, M., Ramakers, T., and Jonker, P. (2010). The Design of LEO: A 2D Bipedal Walking Robot for Online Autonomous Reinforcement Learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3238–3243.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust Region Policy Optimization. In *International Conference on Machine Learning*, pages 1889–1897.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2016). High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Shalev-Shwartz, S. (2012). Online Learning and Online Convex Optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge university press.

Shelhamer, E., Long, J., and Darrell, T. (2017). Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651.

Shen, Y., Tobia, M. J., Sommer, T., and Obermayer, K. (2014). Risk-Sensitive Reinforcement Learning. *Neural computation*, 26(7):1298–1328.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic Policy Gradient Algorithms. In *International Conference on Machine Learning*, pages 387–395.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., and others (2017). Mastering the Game of Go Without Human Knowledge. *Nature*, 550(7676):354.

Smart, W. D. and Kaelbling, L. P. (2000). Practical Reinforcement Learning in Continuous Spaces. In *International Conference on Machine Learning*, pages 903–910.

Smith, R. (2011). Open Dynamics Engine. `http://ode.org/`.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Stulp, F. and Schaal, S. (2011). Hierarchical Reinforcement Learning With Movement Primitives. In *IEEE-RAS International Conference on Humanoid Robots*, pages 231–238.

Sukhbaatar, S., Szlam, A., Synnaeve, G., Chintala, S., and Fergus, R. (2015). MazeBase: A Sandbox for Learning from Games. arXiv:1511.07401.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R., Barto, A., and Williams, R. J. (1992). Reinforcement Learning Is Direct Adaptive Optimal Control. *IEEE Control Systems*, 12(2):19–22.

Sutton, R., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). Horde: A Scalable Real-time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 761–768.

Svinin, M. M., Yamada, K., and Ueda, K. (2001). Emergent Synthesis of Motion Patterns for Locomotion Robots. *Artificial Intelligence in Engineering*, 15(4):353 – 363.

Sünderhauf, N., Brock, O., Scheirer, W., Hadsell, R., Fox, D., Leitner, J., Upcroft, B., Abbeel, P., Burgard, W., Milford, M., and Corke, P. (2018). The Limits and Potentials of Deep Learning for Robotics. *The International Journal of Robotics Research*, 37(4-5):405–420.

Tedrake, R., Zhang, T. W., and Seung, H. S. (2004). Stochastic Policy Gradient Reinforcement Learning on a Simple 3D Biped. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2849–2854 vol.3.

Tesauro, G. (1995). Temporal Difference Learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68.

Tessler, C., Givony, S., Zahavy, T., Mankowitz, D. J., and Mannor, S. (2017). A Deep Hierarchical Approach to Lifelong Learning in Minecraft. In *AAAI Conference on Artificial Intelligence*, volume 3, page 6.

Veerbeek, J., van Wegen, E., Harmeling-van der Wel, B., Kwakkel, G., and Investigators, f. t. E. (2011). Is Accurate Prediction of Gait in Nonambulatory Stroke Patients Possible Within 72 Hours Poststroke?: The EPOS Study. *Neurorehabilitation and Neural Repair*, 25(3):268–274.

Veerbeek, J., van Wegen, E., van Peppen, R., Hendriks, E., Rietberg, M., van der Wees, P., Heijblom, K., Goos, J., Hanssen, W., Harmeling-van der Wel, B., van Jong, L., Kamphuis, J., Noom, M., van der Schaft, R., Smeets, C., Vluggen, T., Vijsma, D., Vollmar, C., and Kwakkel, G. (2014). KNGF Clinical Practice Guideline for Physical Therapy in Patients with Stroke. `http://www.fysionet-evidencebased.nl/index.php/kngf-guidelines-in-english`.

Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. In *Machine Learning*, pages 229–256.

Winstein, C. J. and Kay, D. B. (2015). Translating the Science Into Practice: Shaping Rehabilitation Practice to Enhance Recovery After Brain Damage. In *Progress in Brain Research*, volume 218, pages 331–360.

Winter, D. (2009). *Biomechanics and Motor Control of Human Movement*. Wiley, 4 edition.

Wisse, M. (2004). *Essentials of Dynamic Walking: Analysis and Design of Two-legged Robots*. PhD thesis, Delft University of Technology, Netherlands.

Wu, H. G., Miyamoto, Y. R., Castro, L. N. G., Olveczky, B. P., and Smith, M. A. (2014). Temporal Structure of Motor Variability Is Dynamically Regulated and Predicts Motor Learning Ability. *Nature Neuroscience*, 17(2):312–321.

Wu, Y. and Tian, Y. (2017). Training Agent for First-Person Shooter Game with Actor-Critic Curriculum Learning. In *International Conference on Learning Representations*.

Yin, K., Coros, S., Beaudoin, P., and van de Panne, M. (2008). Continuation Methods for Adapting Simulated Skills. *ACM Transactions on Graphics*, 27(3):81.

Yu, W., Liu, C. K., and Turk, G. (2017). Preparing for the Unknown: Learning a Universal Policy with Online System Identification. arXiv:1702.02453.

Zaremba, W. and Sutskever, I. (2015). Reinforcement Learning Neural Turing Machines. arXiv:1505.00521.

Zhang, T., Kahn, G., Levine, S., and Abbeel, P. (2016). Learning Deep Control Policies for Autonomous Aerial Vehicles With MPC-Guided Policy Search. In *IEEE International Conference on Robotics and Automation*, pages 528–535.

List of publications

The following articles were published during studies at TU Delft. In this thesis, the articles were edited for the consistency of the narration.

Journal articles

2. Koryakovskiy, I., Kudruss, M., Vallery, H., Babuska, R., and Caarls, W. (2018). Model-Plant Mismatch Compensation Using Reinforcement Learning. *IEEE Robotics and Automation Letters*, 3(3): 2471 – 2477.

1. Koryakovskiy, I., Kudruss, M., Babuska, R., Caarls, W., Kirches, C., Mombaur, K., Schloder, J. P., and Vallery, H. (2017). Benchmarking Model-Free and Model-Based Optimal Control. *Robotics and Autonomous Systems*, 92:81 – 90.

Conference articles

3. Koryakovskiy, I., Kudruss, M., Vallery, H., Babuska, R., and Caarls, W. (2018). Model-Plant Mismatch Compensation Using Reinforcement Learning. *IEEE International Conference on Robotics and Automation*.[1]

2. Koryakovskiy, I., Vallery, H., Babuska, R., and Caarls, W. (2017). Evaluation of Physical Damage Associated with Action Selection Strategies in Reinforcement Learning. *IFAC-PapersOnLine*, 50(1): 6928 – 6933.

1. Feirstein, D. S., Koryakovskiy, I., Kober, J., and Vallery, H. (2016). Reinforcement Learning of Potential Fields to Achieve Limit-Cycle Walking. *IFAC-PapersOnLine*, 49(14): 113 – 118.

Workshop presentations

2. Rastogi, D., Koryakovskiy, I., Kober, J. (2018). Sample-efficient Reinforcement Learning via Difference Models, *The Third Machine Learning in Planning and Control of Robot Motion Workshop at ICRA 2018*, Brisbane, Australia.

1. Koryakovskiy, I., Kudruss, (2015). Combining Model Predictive Control Methods and Reinforcement Learning Approaches for Bipedal Walking. *Towards Truly Human-like Bipedal Locomotion: the Role of Optimization, Learning and Motor Primitives Workshop at IROS 2015*, Hamburg, Germany.

[1] The content of this article is identical the journal article of the same title.

Technical reports

1. Kudruss, M., Koryakovskiy, I., Vallery, H., Mombaur, K., and Kirches, C. (2018). Combining Multi-Level Real-Time Iterations of Nonlinear Model Predictive Control to Realize Squatting Motions on Leo. Technical Report 6425, *Optimization Online*.

Summary

In recent years, machine learning has approached the level of development suitable for solving real-world problems. In comparison to supervised learning, reinforcement learning (RL) applications are still rare. One of the reasons is that, in supervised learning applications, the actual actions are undertaken by a human or by an independent controller. In contrast, RL interacts autonomously with the world and learns from the outcome of its own actions. The advantage of this is that RL does not require human-provided data, which is often a problem for supervised learning because collecting and labeling the data is expensive and time-consuming. However, it is not surprising that, in the absence of any prior information about the real world, RL actions may lead to serious damage of the robot or its surroundings. Safety – an often neglected factor in the RL community – requires greater attention from researchers.

Prior knowledge can increase safety during learning. At the same time, it can severely limit a possible solution set and hamper learning performance. Therefore, the focus of this thesis is to identify different forms of prior knowledge required to achieve learning goals with minimum damage.

Bipedal walking robot Leo, which was developed in Delft Biorobotics Lab, is the benchmark example used throughout the thesis. The intrinsic vulnerability of Leo to damage and the challenging walking task itself motivate the usage of this robot.

This thesis starts with model-free RL, where commonly-used exploration strategies are studied with respect to the incurred damage and the cumulative reward. In the case of the bipedal walking robot Leo, the two sources of damage are identified: the fatigue of gearboxes due to backlash re-engagements, and the overall system damage due to robot falls. The results reveal a previously unknown trade-off between the two sources of damage. For example, the results show that a greedy exploration strategy leads to the highest gearbox fatigue, but at the same time it causes the least number of robot falls. On the other hand, the Previous-Action-Dependent-Action method drastically reduces the gearbox fatigue but increases the number of falls. These results suggest that a composite approach towards damage minimization is required. In principle, both sources of damage can be mitigated by using more durable materials and by improving the robot construction. However, repercussions of robot falls in an unstructured environment can be disastrous, because damage depends on the configuration of the environment, which itself is not prepared to withstand falling impacts. Thus, damage due to robot falls is primarily considered for the rest of the thesis.

A substantial part of this thesis is dedicated to learning with a provided approximate robot model. Model-plant mismatch arises from the difficulty of constructing the exact model of Leo due to factors such as hybrid dynamics of walking, torque- and temperature-dependent Coulomb friction, and the accuracy of the estimated

model parameters. In Chapter 3, similarities and differences of nonlinear model predictive control (NMPC) and RL are studied in the context of robustness to parametric and structural uncertainties. The results demonstrate that NMPC has advantages over RL if uncertainties can be eliminated by the identification of system parameters. Otherwise, there exists a break-even point after which model-free RL performs better than NMPC with an inaccurate model.

In Chapter 4, these findings lead to two approaches that combine model-free RL and a nominal NMPC controller. Both approaches (1) provide safety barriers to limit risky RL exploration in dangerous state-space regions, (2) can in principle compensate any type of uncertainty, and (3) allow easy integration with an existing model-based nominal controller. One of them – Model-Plant Mismatch Learning (MPML) – learns a compensatory signal from the difference of the state transition predicted by the internal model and the actual transition, and eliminates the model-plant mismatch by forcing the real system to behave as if it has no uncertainties. This approach was successfully implemented on the real robot Leo for a squatting task.

Unfortunately, MPML is currently not applicable to walking because it requires a different type of an online NMPC controller that would be able to handle hybrid dynamics properly. To overcome this problem, Chapter 5 proposes an iterative model-learning approach. In contrast to MPML, this approach collects some amount of the real-world experience to learn the difference between the real robot and its approximate model. Next, a control policy is learned for the original model enhanced by the difference model. The approach scales to high degree of freedom systems with nonlinear, contact-rich dynamics, and continuous states and actions. Simulated experiments performed with a perturbed model of Leo demonstrate a significantly fewer trials required to mitigate the mismatch compared to cold-started and warm-started RL. Compared to MPML, this approach does not provide safety barriers for constraining dangerous exploration.

The aforementioned approaches demonstrate the usefulness of an existing approximate model for damage reduction. An alternative nature-inspired approach towards reducing the damage while learning is proposed in the final part of the thesis. It is observed that often children learn to walk by progressing through the intermediate stages of sitting, crawling and balancing. The specific task arrangement is also practiced for the rehabilitation of individuals post-stroke. Applied to robots, this suggests that the particular arrangement of learning tasks can reduce the number of falls compared to the cases when the final task is learned directly. This hypothesis is verified in a set of experiments performed on four robot models and two dynamic simulators.

In summary, this thesis studies the influence of different forms of prior knowledge on learning performance and the risk to robot damage, where prior knowledge ranges from physics-based assumptions, such as the robot construction and material properties, to the knowledge of the task curriculum, or the approximate model possibly coupled with a nominal controller. The results reveal the importance of having the approximate model for increasing safety during learning. In the model-free setting, curriculum learning has the potential to gradually learn a

model, which can be combined with the aforementioned model-based RL methods to further reduce the risks of hardware damage.

Samenvatting

In de afgelopen jaren heeft machine learning het niveau bereikt waar het inzetbaar begint te worden voor problemen in de echte wereld. In vergelijking met supervised learning zijn toepassingen die gebruikmaken van reinforcement learning (RL) nog zeldzaam. Eén van de redenen hiervoor is dat bij supervised learning de werkelijke handelingen uitgevoerd worden door een mens of door een onafhankelijke regelaar. RL heeft daarentegen autonoom in de omgeving en leert zelf van de uitkomsten van deze interactie. Het voordeel hiervan is dat RL geen data nodig heeft die door mensen verzameld is. Dit is vaak een probleem voor supervised learning omdat het verzamelen en labelen van de data een duur en tijdrovend proces is. Desalniettemin kunnen RL-acties bij gebrek aan voorkennis over de echte wereld mogelijk leiden tot serieuze schade aan de robot of de omgeving. Veiligheid – een vaak verwaarloosde factor in de RL-gemeenschap – vraagt daarom meer aandacht van onderzoekers.

Voorkennis kan de veiligheid bevorderen tijdens het leerproces. Tegelijkertijd kan voorkennis het aantal mogelijke oplossingen ernstig inperken en de prestaties verminderen. De focus van dit proefschrift ligt daarom op het identificeren van verschillende vormen van voorkennis die nodig zijn om met minimale schade de leerdoelen te behalen.

De tweebenige looprobot Leo, ontwikkeld in het Delft Biorobotics Lab, is als referentiepunt gebruikt in dit proefschrift. De kwetsbaarheid van de robot Leo en de uitdagende taak om te leren lopen zijn de redenen geweest om deze robot te gebruiken.

Dit proefschrift begint met RL zonder een model. Veelgebruikte verkenningsstrategieën worden bestudeerd en vergeleken door te kijken naar de opgelopen schade en het rendement. Voor de tweebenige robot Leo worden twee vormen van opgelopen schade vastgesteld: de vermoeidheid in de overbrengingen door het herhaaldelijk contact ten gevolge van de speling, en de schade aan de robot door een val. De resultaten onthullen een voorheen onbekende afweging tussen deze twee vormen van schade. Uit de resultaten bleek bijvoorbeeld dat een zogenoemde greedy verkenningsstrategie leidt tot de hoogste vermoeidheid in de overbrengingen, maar dat dit tegelijkertijd resulteert in het minste aantal vallen. Daarentegen zorgt de Previous-Action-Dependent-Action methode voor een drastische vermindering van de vermoeidheid, maar neemt hierdoor het aantal vallen toe. Deze resultaten suggereren dat een samengestelde aanpak voor schadebeperking nodig is. In principe kunnen beide vormen van schade voorkomen worden door meer duurzame materialen te gebruiken en door de constructie van de robot te verbeteren. De gevolgen door een val van de robot in een ongestructureerde omgeving kunnen echter rampzalig zijn. De schade hangt namelijk af van de vorm van de omgeving die uit zichzelf niet bestand hoeft te zijn tegen een botsing. Om deze reden wordt in de

rest van dit proefschrift vooral uitgegaan van valschade.

Een substantieel deel van dit proefschrift wordt gewijd aan zelflerende algoritmes die gebruikmaken van een beschikbaar model dat de echte robot benadert. Het verschil tussen het model en de werkelijkheid komt doordat het moeilijk is om een exact model van Leo op te stellen. Dit is te wijten aan verschillende factoren zoals de hybride dynamica van het lopen, moment- en temperatuurafhankelijke Coulomb wrijving en de nauwkeurigheid van de geschatte parameters van het model. In hoofdstuk 3 worden de overeenkomsten en verschillen van nonlinear model predictive control (NMPC) en RL onderzocht met betrekking tot de robuustheid tegen dergelijke parametrische en structurele onzekerheden. De resultaten laten zien dat NMPC voordelen biedt ten opzichte van RL, mits de onzekerheden geëlimineerd kunnen worden door systeemidentificatie. Anders is er een break-even punt vanaf waar RL zonder wiskundig model beter presteert dan NMPC met een onnauwkeurig model.

Gebaseerd op deze bevindingen worden er in hoofdstuk 4 twee manieren van aanpak behandeld die RL zonder model combineren met een nominale NMPC regelaar. Beide methodes (1) zorgen voor veiligheidsbarrières die risicovolle verkenning begrenzen in gevaarlijke toestanden, (2) compenseren in principe voor elke vorm van onzekerheid en (3) kunnen makkelijk geïntegreerd worden met bestaande nominale regelaars die wel gebruik maken van een model. Eén van deze methodes – Model-Plant Mismatch Learning (MPML) – leert te compenseren op basis van het verschil tussen de toestandsverandering zoals voorspeld door het interne model en de werkelijke toestandsverandering. Het elimineert hiermee het verschil tussen het model en de realiteit door het echte systeem te dwingen zich te gedragen alsof er geen onzekerheden zijn. Deze aanpak is succesvol geïmplementeerd op de echte robot Leo tijdens het hurken.

Helaas is MPML momenteel niet toepasbaar op lopende robots, omdat het een ander type online NMPC regelaar nodig heeft die naar behoren om kan gaan met de hybride dynamica. Om dit probleem aan te pakken, wordt in hoofdstuk 5 een iteratieve methode geïntroduceerd om het model te leren. In tegenstelling tot MPML gebruikt de iteratieve methode deels ervaringen uit de echte wereld om het verschil te leren tussen de echte robot en het model dat de echte robot benadert. Vervolgens wordt er een regelstrategie geleerd die gebaseerd is op het originele model versterkt door de geleerde verschillen. Deze methode schaalt naar systemen met veel vrijheidsgraden, met niet-lineaire dynamica en veel contact met de omgeving, en naar systemen met continue toestanden en acties. Gesimuleerde experimenten die zijn uitgevoerd met een verstoord model van Leo laten zien dat er significant minder proeven nodig zijn om de discrepantie te verminderen, in vergelijking met koud gestarte en warm gestarte RL. Vergeleken met MPML zorgt deze aanpak niet voor veiligheidsbarrières die het verkennen van gevaarlijke toestanden limiteren.

De eerder genoemde methodes laten het nut zien van het gebruik van modellen die de echte robot benaderen, met als doel om de schade te beperken. Een alternatieve, op de natuur geïnspireerde methode om schade te beperken tijdens het leren wordt behandeld in het laatste deel van dit proefschrift. Het valt op dat kinderen vaak leren lopen door tussenliggende fases te doorlopen van zitten tot kruipen en

balanceren. Deze specifieke rangschikking van de taken wordt ook gebruikt voor de rehabilitatie van personen na een beroerte. Toegepast op robots suggereert dit dat de specifieke rangschikking van de leerdoelen het aantal vallen kan reduceren in vergelijking met situaties waarin de uiteindelijke taak direct geleerd wordt. Deze hypothese is geverifiëerd door het uitvoeren van een aantal experimenten met vier verschillende modellen van robots en twee dynamische simulatieomgevingen.

Samenvattend behandelt dit proefschrift de invloed van verschillende vormen van voorkennis op mogelijke schade en de prestatie van het zelflerende algoritme. De voorkennis varieert van fysische aannames, zoals de constructie van de robot en materiaaleigenschappen, tot de kennis over de volgorde van de leerdoelen of kennis over het model, mogelijk gecombineerd met een nominale regelaar. De resultaten laten het belang zien van een model dat de echte robot benadert om de veiligheid te verbeteren tijdens het leerproces. In de situatie zonder model heeft gefaseerd leren de potentie om geleidelijk een model te leren. Dit kan worden gecombineerd met de eerder genoemde RL methodes die wel gebruikmaken van een model om het risico op hardwareschade verder te beperken.

Curriculum Vitae

Ivan Koryakovskiy was born in Sosnoviy Bor, Russia on November 29, 1986. In 2004, he graduated from the Liceum №23 in Kaliningrad and the same year started his studies at National Research University of Electronic Technology, Moscow, Russia, specializing in the development of electronic circuits, embedded computers, and programming. In the meanwhile, he started to work part-time at "STK-VICOM Ltd.", Moscow, as a research engineer, successfully combining theoretical knowledge from the university with practical experience in the company.

He earned a Bachelor of Science Degree with distinction in 2008. Several months before graduation, he won Samsung Global Engineering Scholarship which allowed him to pursue a Master of Science Degree at Seoul National University, Seoul, South Korea. He joined the Computer Vision Lab supervised by prof. dr. Kyoung Mu Lee. In 2010, he graduated with a thesis titled "Image Deblurring with Depth-variant Kernel", for which he received a Best Master Thesis Award.

After graduation, he joined Samsung Electronics R&D center in Suwon, South Korea, where he conducted numerous projects related to computer vision and developed several patented inventions.

In 2014, he commenced his Ph.D. research on reinforcement learning applied to walking robots at the Delft Biorobotics Lab at Delft University of Technology in the Netherlands under the supervision of prof. dr. W. Caarls, prof. Dr.-Ing. H. Vallery, and prof. dr. R. Babuška. During this time he supervised several MSc students, assisted with and delivered lectures for reinforcement learning courses, published several journal articles and presented his work at international conferences.