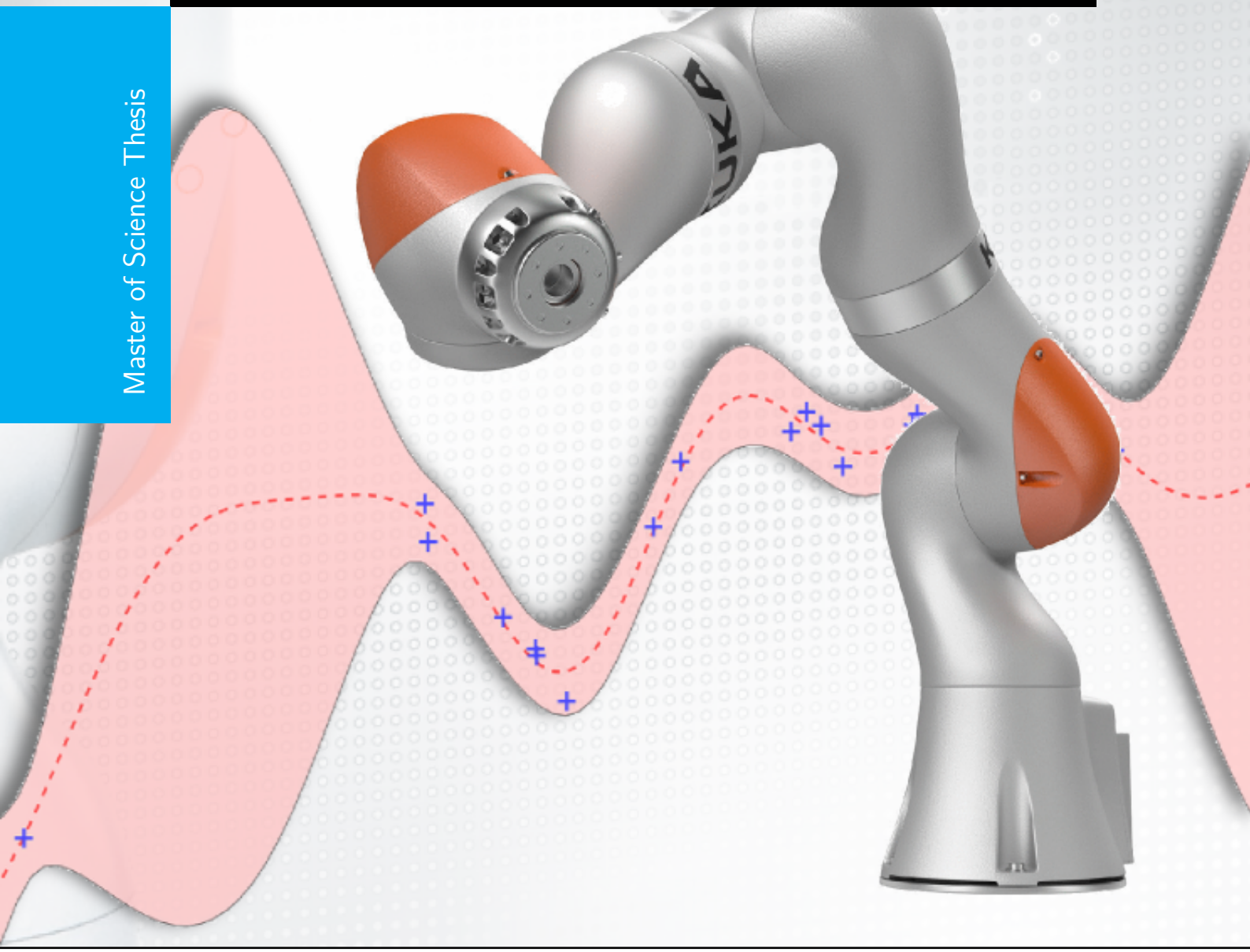
Learning Variable Impedance Control

A Model-Based Approach Using Gaussian Processes

Victor van Spaandonk

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Learning Variable Impedance Control

A Model-Based Approach Using Gaussian Processes

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering
at Delft University of Technology

Victor van Spaandonk

September 17, 2016

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

Abstract

Modern robotic systems are increasingly expected to interact with unstructured and unpredictable environments. This has reiterated the importance of sophisticated reasoning and adaptive motor skill learning. Although low-level methodologies for sensorimotor control have been relatively well studied, constrained motion for robotic manipulators in general environments still remains a challenge.

In addition to viable kinematic trajectories, successful interaction requires a system to adjust the magnitude and direction of the force applied on an object or human being. One force control architecture that is stable under a wide range of conditions is impedance control. However the defining inertial, damping and stiffness parameters are highly task dependent and often difficult to deduce a priori. To this end, one promising strategy is through reinforcement learning. Several frameworks have already emerged that are capable of learning compliant behaviour in this fashion. However the complex and sometimes discontinuous nature of physical interaction in robotics provides additional challenges in designing algorithms capable of learning complex behaviours with minimal interaction time.

This thesis details an extension to the PILCO algorithm for learning variable impedance control. The proposed method attempts to construct a Gaussian Process model of the robot-environment system through interaction. This approach permits long-term inference and planning in a fully Bayesian manner, reducing the required interaction time for convergence and allow for efficient analytical gradient-based policy updates. Two skill learning problems are investigated both in simulation and experiment on a 7-DOF 'KUKA LBR iiwa 7 R800' robot. The first scenario entails learning state-conditioned variable stiffness parameters along predefined motion plans. In the second framework, the agent learns both stiffness parameters and kinematic trajectories simultaneously to complete a constrained motion task.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For humans and animals, effectively engaging the world around us depends almost entirely on physical contact with it. This interaction is often so effortlessly accommodated by our brains and central nervous system, that it is easy to forget how difficult it has been to capture these behaviours in computer algorithms. The conclusion by Pinker [7] in 1994 was that the main lesson of thirty-five years of Artificial Intelligence (AI) research is that the hard problems are easy and the easy problems are hard. This is in line with the observation made in 1988 by roboticist Hans Moravec:

> "*The discovery by artificial intelligence and robotics researchers is that, contrary to traditional assumptions, high-level reasoning requires very little computation, but low-level sensorimotor skills require enormous computational resources.*" [8]

Whether we are writing, walking, opening a door or shaking someone's hand, our body's own dynamics are constantly coupling and interacting to different environments. Redundant degrees of freedom together with the co-activation of muscles allow for regulating a range of stiffness, damping and inertial properties throughout the body [9], [10], [11]. Tactile and visual feedback together with learned feed-forward signals based on internal models suggest how to vary these parameters for different scenarios. It is this time-varying regulation of *impedance* [12] that allow us to effortlessly manage the variety of contact tasks we do. Designing safe and effective interaction controllers requires defining these highly task-dependent parameters.

This thesis develops a framework for learning a variable impedance control task. The proposed method is an extension of the *Probabilistic Inference for Learning COntrol (PILCO)* algorithm [13]. Gaussian Processes [14] are used to learn a model of the coupled robot-environment system through interaction with the real world. These probabilistic models have been shown to increase data-efficiency and minimize the required interaction time on typical benchmark problems [6]. A characteristic that is highly desirable in the context of physical interaction, where contact with an unknown environment poses the risk of wear and possible damage to both the surroundings and the robot itself [15].
The specific task chosen for the robotic system in this thesis is a variant of the Peg-in-Hole (PiH) problem [16] [17]. It entails the (planar) insertion of a peg into an opening

of a secondary stationary part, with the added difficulty of an additional resistance during the final stage of insertion. Of all manufacturing procedures currently automated in an industrial setting, an estimated one third are some form of PiH assembly. For this thesis the task is merely a place-holder for benchmarking the underlying algorithm, however the actual procedure is very relevant to industrial automation.

From 2016 to 2018, industrial robot installations are estimated to increase by at least by 15% Compound Annual Growth Rate (CAGR) per year, reaching a total global sales of about 400.000 units in 2018. According to the International Federation of Robotics (IFR), it is estimated that between 2015 and 2018 about 1.3 million new industrial robots will be installed in factories around the world in addition to an estimated 152.400 service robots, not including personal care and entertainment robots [18]. Robotics is now the world's fastest growing industry, with global spending on robotics and related services to grow at a CAGR of 17% from more than $71 billion in 2015 to $135.4 billion in 2019, according to International Data Corporation (IDC).
Projections aside, let us only consider industrial robots currently in operation, roughly 1.6 million [18]. This translates to more than five hundred thousand robots executing some variant of PiH insertion at this very instance. Besides the earlier stated reasons for intelligent force control for physical Human-Robot-Interaction (pHRI), easy to implement compliance control strategies can yield great benefits for industrial automation in terms of cost reduction, increased flexibility and a safe human-robot workspace environment.

1-1 State-of-the-Art & Related Work

Hurdles in hardware development aside, the absence of predictability makes the safe and effective operation of the next generation robots one of the greatest challenges in shifting the paradigm from rigorously defined factory floors into our lives and dwellings. Hence the capacity for systems to adaptively modify their behaviour is critical if we wish to master the uncertain, non-linear and discontinuous nature of general environments. Learning from past experience is one of the cornerstones of intelligent behaviour that allows for this sophistication and it has further aligned the interests of the machine learning, robotics and adaptive control communities. Reinforcement Learning (RL) has been a popular learning strategy for multiple decades. Inspired by the way biological organisms acquire new skills, it allows an agent to discard fruitless strategies while pursuing those that yield progress through an iterative trial-and-error procedure of representation refinement and policy improvement. A brief and loosely chronological overview of related work in biological physiology research, robotic motor skill learning and the use of internal models is presented next.

1-1-1 Biological Sensorimotor Skill Learning

Considerable research has been done in physiology, biology and neuroscience on how humans learn new sensorimotor skills. Studies of limb movements have provided insights into the computations, mechanisms, and taxonomy of human sensorimotor learning. Motor tasks differ with respect to how they weight different learning processes. Which include adaptation, internal-model formation that reduces sensory-prediction errors and reinforced learning [19]. In skill learning tasks such as PiH, which for the most part do not involve a perturbation,

it seems improved performance is manifested as reduced motor variability and probably depends less on adaptation and more on success-based exploration [20]. Explicit awareness and declarative memory also contribute, with varying degrees, to motor learning. The modularity of motor learning processes maps, at least to some extent, onto distinct brain structures as well as signal generators in the Central Nervous System (CNS) [20].
Kording experimentally verified in [21] that the CNS employs probabilistic Bayesian models on multiple levels during sensorimotor learning. For example in sensor fusion, internal model formation and strategy building. A finding that is echoed by Wolpert in [19], noting the way humans use Kalman-like inference for making sense of sensory information.

In general, optimizing motor performance is achieved through three classes of control: (1) feed-forward control, which is critical given the feedback delays in the human sensorimotor system[1]; (2) reactive control, which involves the use of sensory inputs to update ongoing motor commands; and bio-mechanical control, which involves modulating the compliance of the limb [19]. This thesis focusses on developing computational strategies for determining reactive time-varying compliance.

Intelligent, context-specific compliance responses are consistent with the theoretical framework of optimal feedback control, which suggests that the CNS sets up feedback controllers[2] that continuously convert sensory inputs into motor outputs [19] and that these are optimally tuned to the goals of the task by trading off energy consumption with accuracy constraints (see Chapter 4-2).
In 1988 Kawato suggested in [22] that to learn a new voluntary movement skill, the nervous system has to overcome three problems; determining a desired trajectory in the visual coordinates (see Chapter 4-1); transformation of the trajectory from visual coordinates to body coordinates (see Chapter 2-1) and finally the generation of low-level motor commands (see Chapter 2-2). These distinct phases are very familiar to the robotics and control communities.

1-1-2 Learning Robotic Motor Control

In 1992 Gullapalli used an associative search network to learn reactive admittance control for a PiH task [23] for the first time. What was especially novel to this was the incorporation of uncertainty and noise. The neural network controllers determined velocity commands to the end-effector based on position and force inputs.

For robotic applications the focus has shifted towards a family of algorithms known as *direct policy search* [24] [25]. These methods directly search for a solution in (some subset of) the policy space. This requires the formulation of gradients[3] based on noisy observations. Throughout the last decades much research has been done on formulating viable techniques for gradient estimation. This culminated in the formulation of the REINFORCE method[4] [26] , *The Policy Gradient Theorem* [27] and finally the & Natural Gradient [28] [29]. This natural gradient formulation was employed by Byungchan in an actor-critic framework to learn impedance tasks such as opening a door, catching a ball and moving through an unknown force field [30].

[1] In the order of the time-delay to the international space station

[2] Sometimes referred to as motor schema or generalized motor programs

[3] gradient-free methods such as simulated annealing, evolutionary computation and reward-weighed regression aside.

[4] In optimization literature often referred to as the likelihood-ratio trick.

These methods were all action-perturbing algorithms, relying on random variations per time step for exploration of the search space. The disadvantage of this is that subsequent randomness tends to cancel itself out or may be too fast to fall within the bandwidth of the system. Also on robots it might not be safe to supply fast changing random actions as low-level motor commands and it can cause large variance in parameter updates, an effect which grows with the number of time steps [31]. The most important drawback however is in the number of trials needed for accurate gradient estimates, a minimum of $(n_\psi + 1)$, the number of parameters in the policy representation.

An alternative parameter-perturbing approach was summarized by Sehnke and Ruckstiess in [32] and [33]. Peters applied an approach of this family called *reward-weighted regression* to learning operational space control in 2007 [34]. The resulting algorithm was found to be efficient, smooth and scalable.
The *Policy learning by Weighting Exploration with the Returns (PoWER)* algorithm proposed by Kober in [31] computes an input by averaging the perturbation vector and weighing them with a measure for the reward-to-go. It turns out that this approach follows the natural gradient without explicitly computing it.

Efforts by, among others, Theodorou have successfully connected the framework of stochastic optimal control with path integrals to derive a novel approach to RL with parameterized policies in [35] called *Policy Improvements with Path Integrals (PI^2)*. Much like PoWER, it uses reward-weighted averaging for the policy updates. They showed that policy improvement can be transformed into an approximation problem of a path integral which has no open tunable algorithmic parameters other than the exploration noise. The key insight is the transformation of the stochastic optimal control problem into probabilistic estimation problem, which can then be solved using sampling techniques. The stochastic Hamilton-Jacobi-Bellman (HJB) equations allows formulating a 2^{nd}-order partial differential equations (pde) of the time derivative of the value function. This pde can be linearized under mild assumptions and solved in its its logarithmic form.
In contrast to policy gradient methods, in PI^2 there is no need to explicitly calculate a gradient, which can be sensitive to noise and large derivatives in the value function[5]. Which makes this computation robust to non-smooth dynamics and cost functions.
PI^2 has been used for learning variable (joint-space) impedance control in [1]. In this work, *Dynamic Movement Primitives (DMPs)* - proposed by Ijspeert [36] in 2006 - are used to represent movement and impedance trajectories in a single dynamical system, depicted schematically in Figure 1-1. DMPs are a compact way of describing dynamics that have guaranteed stable properties to a predefined attractor state.
The PI^2 algorithm has also been applied to learning reactive force control in stochastic environments by Stulp in [37]. From human arm studies it is observed that predictable perturbations are countered by feed-forward control whereas stochastic perturbations are reacted to by increased arm stiffness. Finally it was shown to be effective in learning more finely compliant manipulation such as opening a door and picking up a pen from a table in [38].

A recent use of DMPs in the context of Cartesian impedance learning is presented in [39]. the proposed algorithm takes a Cartesian reference trajectory and Force/Torque-profile as input and adapts the movement on-line so that the resulting forces and torques match some

[5]Instead the gradient of the exponentiated value function is implicitly calculated by a weighted average of the exploration parameter weighted by the exponentiated cost of every sampled trajectory.

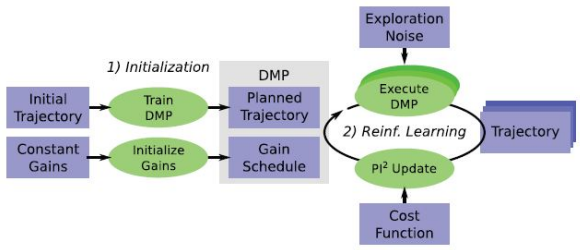


Figure 1-1: *High level overview of learning both motion plans and variable stiffness gains with the PI^2 learning algorithm [1].*

reference profiles supplied earlier by expert demonstrations. This strategy allows a robot to transfer these taught skills for a certain part-mating procedure to new configurations over the course of several iterations by iteratively adapting the learned trajectory to improve the task performance using force feedback

Figure 1-2 categorizes the different policy search architectures. The transition over the last decades has been from value function based methods to direct policy search based on various types of estimated gradients. Then going from Natural Actor-Critic (NAC) to PoWER represented a transition from action perturbation to parameter perturbation and from estimating the gradient to reward-weighted averaging. Stulp in [2] observes this trend continuing into more Black-Box Optimization (BBO).

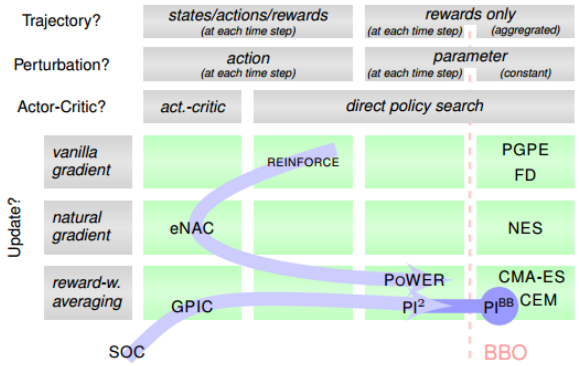


Figure 1-2: *Classification and trends within direct policy search RL (arrows indicate time) [2].*

1-1-3 Model-Based Learning

The progression of RL given above have all served to provide better policy improvements based directly on obtained experience. One strategy for further minimizing the required interaction time is through model-based policy updates. Model-learning RL utilizes recorded data to construct an internal model that can be exhausted for policy updates, thereby more efficiently exploiting data. This has been demonstrated to increase data-efficiency and robustness [40]. From Chapter 1-1-1, the concept of internal model-formation was found to be critical in human skill learning and minimizing interaction time. This work is an attempt to capture some of those benefits.

An early example of increasing data-efficiency and robustness through model-based planning for robotic learning can be found in the work by Aboaf in 1989 [41]. A comprehensive discussion on the modern use of models in RL is presented in [42]. A much cited design of model-based learning appears in the classic work by Sutton [4] as the Dyna-architecture (see Chapter 3-1-1).

However the resulting model-conditioned policies may not be a sufficient approximation to the real dynamics. This problem of *model-bias* is a major challenge and when not addressed can result in divergent - and in the case of pHRI - possibly dangerous control policies. To overcome this problems of inaccurate models, Abbeel in [43] adds a heuristic bias term along the recorded trajectory to update the model after gathering real experience. In this approach, real-life trials are used to evaluate, while gradient estimations of the policy with respect to its parameters are used for improvements. This only requires a single trial to improve the policy[6], whereas model-free methods typically require this to be in the order of the number of policy parameters [43].
Another way to diminish the effects of model-bias is through uncertainty quantification. Traditionally models have often been deterministic, an approach which rejects much of the noise and uncertainty in recorded data. The endeavour of effectively dealing with uncertainty has caught the shared interest of the machine learning, robotics and control communities. In order to minimize the required interaction time and mimic the way in which human use Bayesian inference, this work will follow such approach to model-based RL. A useful non-parametric framework for Bayesian modelling are Gaussian Processes.. An excellent introduction to the use of Gaussian Process (GP) models in machine learning was written by Rasmussen in 2006 [44]. Recently, Rasmussen & Deisenroth proposed a rigorous way of making long-term predictions for dynamical systems with GPs while propagating all the uncertainties [5]. This approach culminated in the PILCO algorithm. Its effectiveness was shown in several low-Degrees of Freedom (DOF) benchmark problems as well as 5-DOF unicycle balancing [6]. Its philosophy is similar to [43], iteratively making batched policy improvements grounded in real-life experience with the addition of rigorous uncertainty accounting and analytical policy gradients.

1-2 Current Research Objectives & Thesis Outline

However the speed of convergence and the limited dimensionality of the policies still remains an issue. This thesis attempts to fuse some of the discoveries made with respect to biological and robotic skill-learning mentioned in Chapter 1-1-1 and 1-1-2. For this we develop an extension to the analytical gradient-based learning strategy developed in [5] known as PILCO. The structure of the proposed model-based strategy allows formulating analytical expressions for gradient-descent policy updates in a fully Bayesian manner, instead of through estimation of deterministic gradients. Quantified uncertainty during learning can exploit parts of the state-space where the assumption of continuous dynamics is valid, while upholding a measure of parts of the state-action space where such a postulation breaks down. It is hypothesized that this decreases interaction time with respect to model-free methods while still allowing for high-level policies that can be generalized to different scenarios. Specifically this thesis details the development of two algorithms; *Variable Impedance Learning (VIL)* and *Variable*

[6] As long as the estimated direction is within 90 degrees of the true gradient.

Impedance Learning with Trajectory Adaptation (VILTA). Methods for learning stiffness parameters and kinematic trajectories for executing constrained motion tasks in Cartesian space. The architecture proposed in this thesis could also be employed to learn impedance control in joint-space, but this is not presented here. The general research objectives are given below, followed by an outline on the structure of this document.

(1) Formulate a sample-efficient learning framework for robotic impedance control. The general objective of this thesis is to design an algorithm capable of learning variable impedance control with minimal interaction time on the system.

(a) Learn to perform a constrained motion task in an optimal manner through variable stiffness gains. The primary practical goal of this thesis is to apply the aforementioned algorithm to learn a variable impedance control task on a simulated 3-DOF planar robot and on a 7-DOF *KUKA LBR 7 iiwa* manipulator.

(b) Learn to perform a constrained motion task by modulating stiffness gains and on-line references adaptation. The secondary practical goal of this thesis is to apply the aforementioned algorithm to learn variable impedance gains simultaneously with an optimal trajectory for a constrained motion task.

(c) Investigate the impact of local discontinuities on model-based learning. One important aspect of this research lies in investigating how discontinuities during contact transitions affect the validity of constructed models and how uncertainty quantification can be employed to mitigate these phenomena and minimize the effects of model bias.

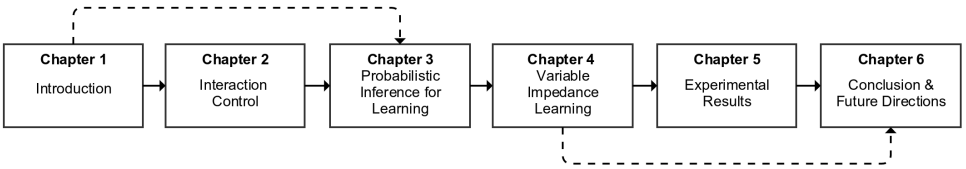


Figure 1-3: *Roadmap of this document. Solid arrows indicate recommended reading order. Dashed arrows indicate alternative order for quick impression.*

Figure 1-3 gives a schematic overview of this document along with a recommended reading order. First an introduction to the implementation of impedance control for robotic manipulators is given in Chapter 2. Some mathematical notation is introduced, followed by a brief introduction to the relevant practical implementation of low-level spatial impedance control. Chapter 3 summarizes in detail the steps that comprise the PILCO algorithm. This includes concepts for learning non-parametric GP models, making long-term predictions and gradient-based policy optimization. Novel concepts, additions and extensions of this thesis specific to model-based Variable Impedance Control (VIC) are presented in Chapter 4. Experimental results are reported in Chapter 5. Two scenarios are investigated in both simulation and real-world experiments; VIL along predefined motion plans; and VILTA for learning a PiH-like task. Concluding remarks and an articulation of future directions is given in Chapter 6.

A Note on Notation

This report follows standard metric and SI units and the comma is used as decimal sign. In general, scalar values are denoted by light, small case, symbols (x). Functions are indicated in the same manner. Vectors are indicated with bold, small case symbols ($\boldsymbol{x}$). Matrices are denoted with light, capital case letters (X). Scalar values indicating a constant such as the total number of iterations or time steps are also indicated with light, roman, capital case letters (T).

Chapter 2

Interaction Control

In general there exists two distinct modes of operation for any mechanical linkage; free motion and constrained motion. Many of the manufacturing processes that were the initial driving force of automation algorithms can be described as free motion tasks. Applications include pick-and-place tasks, spray painting, welding and laser cutting. Traditional control system design thus had a strong focus on these types of tasks. Positional feedback gains are applied to the actuators in order to control the robot into a desired configuration. A compensating dynamical model of the system can be embedded in the control law for higher speed and accuracy.

Contact tasks are motions in which one or more directions in the workspace are constrained due to external objects. When a manipulator makes contact, a transition discontinuity arises from the interaction forces between the robot and its environment. When using high-gain positional feedback schemes, these external forces can result in destructive and sometimes dangerous behaviour, especially in the case of high inertia systems. To adequately control a manipulator during constrained motion, safe and effective means for dealing with the resulting forces is required. These methodologies are often termed *compliance-*, *interaction-*, or *force control* methods. *Compliance* can be considered as a measure of the ability of a manipulator to react to interaction forces [45]. Broadly all these schemes can be categorized into two categories: direct and indirect force[1] control.

An extensive survey on the general topic of interaction control can be found in [46]. Earlier research includes experimental comparisons of a variety of strategies [47] and the relation to safe physical Human-Robot-Interaction (pHRI) [48]. The paper detailing the state-of-the-art in robotic force control after the first wave of research in the 1980's was by Whitney in [49], which has since been cited many times as a benchmark when referring to different methodologies. For a bird's eye view of force control strategies and an interesting analysis of their mutual relationships, the reader is referred to [50].

This research focuses on the implementation of an impedance control scheme with learned state-dependent gains and trajectories in Cartesian space. This chapter will outline the theory and implementation of simple and general Cartesian impedance control as applied to

[1] For readability, when referring to forces, the reader is expected to read forces/torques.

an n-Degrees of Freedom (DOF) robotic manipulator. The reasons for choosing this control methodology are outlined in Chapter 2-2. In order to establish clear and consistent notation, some underlying mathematics regarding the kinematic and dynamic formalism of robotic manipulators is presented in Chapter 2-1. As will be detailed, several types of implementation for Impedance Control (IC) exist, each with their respective strengths and weaknesses. The proposed learning algorithm is able to work with all architectures for learning optimal behaviour in each respective setting. To appreciate the consequences of this, the control theory for both types of IC are outlined in Chapter 2-2-1 and 2-2-2. Finally a description of force-guided Peg-in-Hole (PiH) tasks is given in Chapter 2-3.

2-1 Mathematical Background of Serial Manipulator Kinematics

We start by establishing some basic concepts related to kinematics and the spatial sets we use to express them. For a more in depth description regarding any of these topics the reader is referred to [3]. A discussion of orientation representation is presented in Appendix B.

We will consider robotic manipulators consisting of a serial (open) kinematic chain of $n+1$ links connected by n joints. We let $\boldsymbol{q}$ be the $(n \times 1)$ vector representing the state of the system in *joint space*[2]. We set $\boldsymbol{p}_e = \begin{bmatrix} x & y & z \end{bmatrix} \in \mathbb{R}^3$ as the position and $H_e \in \mathbb{R}^{3\times3}$ the rotation matrix of the Tool Center Point (TCP)[3] with respect to the base frame (see Figure 2-1).

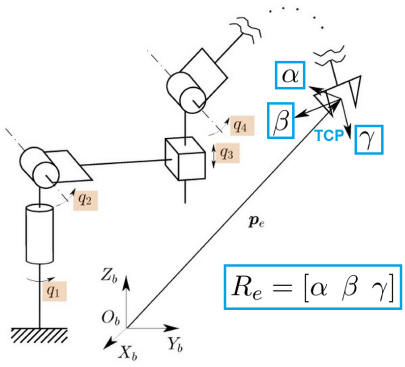


Figure 2-1: *Schematic drawing of a serial chain manipulator with notations [3].*

The kinematic model describes the mapping between the joint space and workspace variables of the manipulator. This mapping is completely defined by a *homogeneous transformation matrix*, belonging to the *special Euclidean group* $\mathbb{SE}(3)$ and where $R_e(\boldsymbol{q})$ is part of the *special orthonormal group* $\mathbb{SO}(3)$:

$$H_e = \begin{bmatrix} H_e(\boldsymbol{q}) & \boldsymbol{p}_e(\boldsymbol{q}) \\ \mathbf{0} & 1 \end{bmatrix} \tag{2-1}$$

The differential kinematics of the manipulator gives the relations between joint velocities $\dot{\boldsymbol{q}}$ and end-effector velocities

$$\boldsymbol{v}_e = \begin{bmatrix} \dot{\boldsymbol{p}}_e^T \\ \boldsymbol{\omega}_e^T \end{bmatrix}^T = \begin{bmatrix} J_p \\ J_o \end{bmatrix}^T = J_g(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2-2}$$

[2] Also called *configuration* space

[3] The point on the end-effector around which the orientation of the manipulator wrist is being defined.

This differential mapping is known as the *geometric Jacobian*. This description of the end-effector orientation by R_e is redundant due to six orthonormality constraints acting on the rotation matrix. The minimal orientation[4] of the end-effector can be described by a (3×1) vector $\boldsymbol{\phi}_e$, which can be extracted from the rotation matrix in terms of three *Euler Angles* [51]: $\boldsymbol{\phi} = [\alpha \ \beta \ \gamma]^T$:

$$R_e(\boldsymbol{\phi}) = R_i(\alpha)R_j(\beta)R_k(\gamma) \tag{2-3}$$

where $R_i(\alpha), R_i(\beta), R_i(\gamma)$ are the matrices of the elementary rotations about three independent coordinate axes of successive frames.

When expressing the state of the end-effector in this minimal form it is said to be in *Operational Space Formulation*, as introduced by Khatib in [52]:

$$\boldsymbol{x}_e = \begin{bmatrix} \boldsymbol{p}_e(\boldsymbol{q}) \\ \boldsymbol{\phi}_e(\boldsymbol{q}) \end{bmatrix} = f_k(\boldsymbol{q}) \qquad \dot{\boldsymbol{x}}_e = \begin{bmatrix} \frac{\partial \boldsymbol{p}_e}{\partial \boldsymbol{q}} \\ \frac{\partial \boldsymbol{\phi}_e}{\partial \boldsymbol{q}} \end{bmatrix} \dot{\boldsymbol{q}} = \begin{bmatrix} J_p \\ J_\phi \end{bmatrix} \dot{\boldsymbol{q}} = J_a(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2-4}$$

As a result, the Jacobian can be computed analytically by differentiation. This analytical Jacobian is in general different from the geometric one, since the end-effector angular velocity $\boldsymbol{\omega}_e$ with respect to the base frame is not in general given by $\dot{\boldsymbol{\phi}}_e$.
This relation between the end-effector angular velocity $\boldsymbol{\omega}_e$ and the derivative of Euler Angles $\dot{\boldsymbol{\phi}}_e$ for a given orientation angles is denoted by the transformation matrix $T(\boldsymbol{\phi}_e) \in \mathbb{R}^{3\times3}$. Which allows us to relate the two versions of a Jacobian:

$$J_g = T_J(\boldsymbol{\phi}_e)J_a = \begin{bmatrix} \boldsymbol{I}_3 & 0_3 \\ 0_3 & T(\boldsymbol{\phi}_e) \end{bmatrix} J_a \tag{2-5}$$

These transformation matrices already hint at the presence of singularities when using Euler angles, which arise when $T(\boldsymbol{\phi}_e)$ loses rank. In these singular configurations, it is not possible to describe an arbitrary angular velocity with a set of Euler angles time derivatives and the extraction of the Euler angles corresponding to a given rotation matrix is not unique. For this thesis, an alternative Euler angle representation (see Appendix B) is implemented that partially mitigates these representational singularity problems.

Besides these mathematical anomalies, the physical orientation of a manipulator can yield so called *kinematic singularities*. These occur either at the boundary of the workspace, when the arm is fully stretched, or inside the workspace by the alignment two or more axes[5]. As opposed to the former type, the latter case requires careful consideration, since they can appear anywhere in the workspace for certain operational space trajectories. Kinematic singularities can yield an infinite solution to the *inverse kinematic problem*, impose restrictions on the mobility of te structure and yield extreme joint velocities for small velocities in the operational space as the Jacobian approaches a numerical singularity[6]. Inverse kinematics has been described by Paul as one of the hardest problems in control [53].

[4]The term '*orientation*' is used when referring to rotational states, while the term '*pose*' is reserved for the complete position and rotational state of the end-effector.

[5]The former type are known as boundary singularities, while the latter are referred to as internal singularities.

[6]Manipulators with a *spherical wrist* allow decoupling the internal singularities into arm- and wrist singularities, simplifying singularity formalism.

2-2 Impedance Control

As the name suggests, indirect force control is achieved by establishing an implicit relation between motion and force. The framework that elegantly describes this relation was introduced by Neville Hogan in [12], who termed it *IC*. Instead of trying to model the interaction as a disturbance - which violates the typical state-independence assumption - or trying to model it as an uncertainty - one that will likely alter the system's order and be in the same frequency range as control - it aims to achieve a dynamical relationship between end-effector motion and contact forces.

> "*The dynamic interactions are not a source of disturbances to be rejected, but an integral part of the task. This is the idea behind impedance control. It is assumed that the robot's task must fundamentally be described, not in terms of motions, nor in terms of forces, but in terms of the relations between them.*" - Hogan [12]

To this end the system is modeled as an equivalent physical network with ports, capable of exchanging energy through force and motion.
Generalized from concepts of electrical circuit theory; *Impedance* takes an input velocity to impose a force: $\mathrm{Z}(s) = \boldsymbol{f}(s)/s\boldsymbol{x}$ and its causal dual *Admittance*: $\boldsymbol{Y}(s) = s\boldsymbol{x}/\boldsymbol{f}(s)$, are port operators that establish a relation between force (effort) and velocity (flow)[7]. IC then tries to control the system behaviour at its *interaction ports*, characterized by the power exchange $\boldsymbol{P} = \boldsymbol{e}^T\boldsymbol{f}$. Any part of a system can be characterized as an interaction port, as long as its exchange variables have an inner product equalling power. Figure 2-2 shows several instances of a TCP acting as an interaction port for stiffness regulation in impedance control.

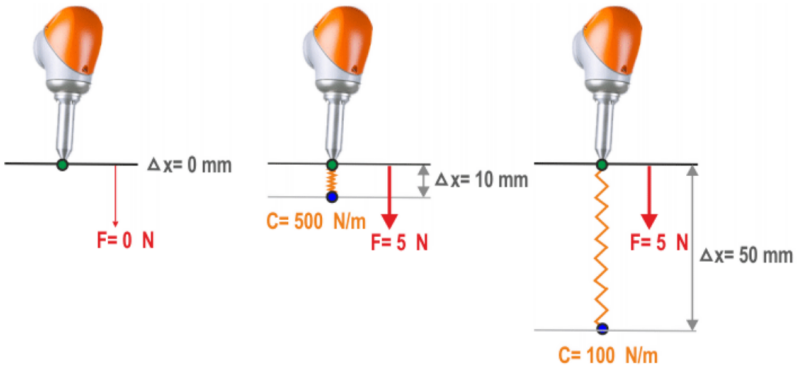


Figure 2-2: *Depiction of closed-loop stiffness behaviours. $\Delta\boldsymbol{x}$ indicates the deviation from a virtual trajectory. The force exerted is the response proportional to the desired stiffness here called C.*

Physical objects have two main characteristics associated to them: Inertial and kinematic surface constraints. If the object is stiff and/or heavy this means it is best modeled as an admittance, since we can not impose a motion on a kinematic constraint or large mass. Causal duality would then have the manipulator act as a pure impedance. If the object is compliant, the causality is inverted. However most robots consist of links of relatively large mass driven

[7] In mechanics often position is used as output variable instead of velocity. This is due to some inherent peculiarities of mechanical systems being heavily dependent on configuration.

by highly geared actuators to amplify motor torque. The total inertia apparent at the end effector is increased by the reflected inertia of the motor, which for high gear ratios can dwarf the mass of the links. These inertial properties are difficult to overcome and tend to dominate the system's response. Thus even though stiff environments are naturally modeled as an admittance, it is difficult to make a robot behave as a pure impedance, and it is usually more feasible to make it behave as primarily a mass (or admittance).
In other words, impedance behaviour is usually ideal, but admittance behaviour is often more easily implemented in real hardware, which itself prefers admittance causality because of its inertial nature. The choice of particular controller structure for an application must be based on the anticipated structure of the surroundings and robot, as well as the way that they are coupled. A switching strategy was developed in [54] that is capable of switching between impedance and admittance control in the face of different environments.

2-2-1 Simple Cartesian Impedance Control

In order to approach the ideal of impedance control, being able to specify the apparent spring and damping behaviour at the end-effector, one has to overcome the inherent impedance characteristics of the manipulator. One way of course is to compensate for the robot's dynamics with active control, detailed in Chapter 2-2-2. Another more straightforward approach is through highly back-drivable mechanics; minimal gearing, low friction and small inertias controlled with high bandwidth DC motors. Allowing for the specification of desired end-point behaviour as a spring and damper:

$$\boldsymbol{W}_{ext} = K_P(\boldsymbol{x}_v - \boldsymbol{x}_e) + K_d(\dot{\boldsymbol{x}}_v - \dot{\boldsymbol{x}}_e) \tag{2-6}$$

First let us review the control theory of robotic manipulator dynamics. The general class of systems we consider can be represented by a set of n nonlinear differential equations in *joint space*:

$$\dot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{u}) - J_g^T \boldsymbol{W}_{ext}, \qquad \text{where } \boldsymbol{q} \in \mathbb{R}^n,\ \boldsymbol{u} \in \mathbb{R}^m$$

Using the Lagrangian formulation, the form known as the *Mechanical Equation* can be adopted:

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{u} - J_g^T(\boldsymbol{q})\boldsymbol{W}_{ext} \tag{2-7}$$

where $\boldsymbol{M}$ is the inertial matrix, C is the centrifugal and Coriolis term, $\boldsymbol{g}$ accounts for gravity forces and $\boldsymbol{W}_{ext} = [\boldsymbol{f}^T, \boldsymbol{\tau}^T]^T$ is the wrench of the external contact forces and torques applied on the end-effector and $\boldsymbol{u}$ is the vector of input torques applied to the joints, accounting for possible gearing.

The closed-loop behaviour of Equation (2-6) is approached by transposed Jacobian control, utilizing the direct and differential kinematic map of (2-4) and possible gravity compensation:

$$\boldsymbol{\tau}_m = J(\boldsymbol{q})^T \left[K_P \left(\boldsymbol{x}_v - \boldsymbol{f}(\boldsymbol{q}) \right) + K_D(\dot{\boldsymbol{x}}_v - J(\boldsymbol{q})\dot{\boldsymbol{q}}) \right] + \boldsymbol{g}(\boldsymbol{q}) \tag{2-8}$$

Resulting in the closed-loop system:

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = J(\boldsymbol{q})^T \left[K_d \left(\boldsymbol{x}_v - \boldsymbol{f}(\boldsymbol{q}) \right) + \boldsymbol{B}_d(\dot{\boldsymbol{x}}_v - J(\boldsymbol{q})\dot{\boldsymbol{q}}) \right] - J^T(\boldsymbol{q})\boldsymbol{W}_{ext} \tag{2-9}$$

Simple impedance control thus consists of driving an intrinsically light, low-friction mechanism with force- or torque-controlled actuators, and using motion feedback to increase output

impedance. This approach makes no attempt to compensate for any physical impedance (mass, friction) in the mechanism, so the actual output impedance consists of that due to the controller plus that due to the mechanism.
Due to the non-linear kinematics of a typical robot, constant stiffness and damping matrices in the control law of Equation (2-6) result in stiffness and damping apparent at the robot's end-effector (the usual interaction port) that vary with its configuration. Note that the simple impedance controller does not rely on force feedback and does not require a force sensor.

Some interesting possibilities for Variable Impedance Learning (VIL) present itself with respect to Simple Impedance Control (SIC). Accurate dynamics compensation can be a major challenge. Precise models might not always be available or practical implementation might be outside the scope of moderate embedded hardware specs. For constant gain IC a trade-off has to be made between performance on the one hand and energy, safety and stability constraints on the other hand. Variable Impedance Control (VIC) allows time-varying gains to be conditioned on a reward function tailored to the higher-level goal specification, which can resolve the trade-off in an optimal manner, compensating for model mismatch only at places where it is beneficial for the agent, while still exploiting the mentioned benefits and simplicity of SIC.

2-2-2 General Cartesian Impedance Control

General Impedance Control (GIC) is an extension in which we can fully specify the inertial, damping and stiffness parameters experienced by the end-effector. It was proposed by Hogan in [12] and it aims to achieve the closed-loop system:

$$M_d\ddot{\tilde{\boldsymbol{x}}}_e + K_D\dot{\tilde{\boldsymbol{x}}}_e + K_P\tilde{\boldsymbol{x}}_e = \boldsymbol{W}_{ext} \tag{2-10}$$

The block diagram in Figure 2-3 shows the control system design that achieves this behaviour up to a kinematic coupling term. *Dynamic model-based compensation*[8] is achieved by the following control law

$$\boldsymbol{u} = M(\boldsymbol{q})\boldsymbol{\alpha} + C(\dot{\boldsymbol{q}}, \boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) \tag{2-11}$$

which when substituted in (2-7) results in *resolved acceleration* in joint space:

$$\ddot{\boldsymbol{q}} = \boldsymbol{\alpha} - \boldsymbol{\delta} - M(\boldsymbol{q})^{-1}(\boldsymbol{q})J_g^T\boldsymbol{W}_{ext} \tag{2-12}$$

Where $\boldsymbol{\delta}$ accounts for any unmodeled dynamics resulting from inaccurate parameter estimation or unmodeled friction.

The dynamics of the system can be directly expressed in the operational space formulation [52]. This will require careful consideration of kinematic and representational singularities, as well ass redundant DOFs. Taking the time derivative of (2-4) yields

$$\ddot{\boldsymbol{x}}_e = \dot{J}_a(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + J_a(\boldsymbol{q})\ddot{\boldsymbol{q}} \tag{2-13}$$

Which allows to express the Mechanical Equation from (2-7) in operational space coordinates:

$$\Theta(\boldsymbol{x}_e)\ddot{\boldsymbol{x}}_e + \Gamma(\boldsymbol{x}_e, \dot{\boldsymbol{x}}_e)\dot{\boldsymbol{x}}_e + \boldsymbol{\eta}(\boldsymbol{x}_e) = J_a^{-T}(\boldsymbol{q})T_J^T(\boldsymbol{x}_e)\boldsymbol{u} + T_J^T(\boldsymbol{x}_e)\boldsymbol{W}_{ext} \tag{2-14}$$

[8] Sometimes called *inverse dynamics* or *computed torque control*

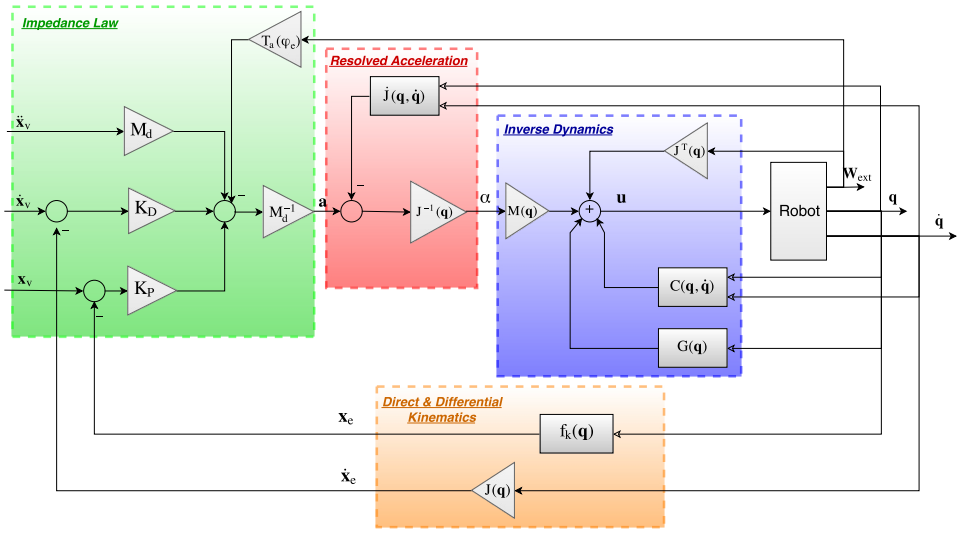


Figure 2-3: *Block diagram of general Cartesian impedance control. Four distinct blocks can be identified; inverse dynamics in joint space, transformation of joint variables to Cartesian coordinates, resolved acceleration in Cartesian space and defining the desired closed-loop behaviour. Open arrowheads indicate the variable is used as an input to the block, solid heads indicate a multiplication.*

Which for the case of a non-redundant manipulator ($n = m$) in a non-singular configuration (J_a^{-1} exists) and dropping the dependencies for readability, has

$$\begin{aligned} \Theta &= J_a^{-T} M J_a^{-1} \\ \Gamma &= J_a^{-T} \left(C - J_a^{-1} \dot{J}_a \right) J_a^{-1} \\ &= \left(J_a^{-T} C - \Theta \dot{J}_a \right) J_a^{-1} \\ \boldsymbol{\eta} &= J_a^{-T} \boldsymbol{g} \end{aligned} \tag{2-15}$$

To arrive at a closed-loop double integrator system in Cartesian space, $\boldsymbol{\alpha}$ in (2-12) is set to

$$\boldsymbol{\alpha} = J_a^{-1}(\boldsymbol{q}) \left(\boldsymbol{a} - \dot{J}_a(\boldsymbol{q}, \dot{\boldsymbol{q}}) \dot{\boldsymbol{q}} \right) \tag{2-16}$$

Looking at (2-15) and (2-16) the inherent difficulties with respect to control in the operational space become apparent. Besides having to compute the kinematic mappings on-line, it is not guaranteed to be well-behaved throughout the workspace. Even when avoiding exact singularities, in computer-controlled systems, inversion of a near-singular matrix will cause numerical problems long before the physical system reaches a physical singularity[9].

To arrive at a system that is uncoupled from the system's inertial matrix and external forces, computed torque control is combined with force-based inner loop impedance control. For this the control law in (2-11) is expanded to:

$$\boldsymbol{u} = M(\boldsymbol{q})\boldsymbol{\alpha} + C(\dot{\boldsymbol{q}}, \boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + J_g^T(\boldsymbol{q})\boldsymbol{W}_{ext} \tag{2-17}$$

[9]For a redundant manipulator, a joint space control scheme is naturally transparent to this situation, since redundancy has already been solved by inverse kinematics, whereas an operational space control scheme should incorporate a redundancy handling technique inside the feedback loop.

The inclusion of the last term exactly compensates the contact forces and makes the manipulator infinitely stiff with respect to the environment. For $\boldsymbol{\alpha}$ we set

$$\begin{aligned}\boldsymbol{\alpha} &= J_a^{-1}(\boldsymbol{q})\left(\boldsymbol{a} - \dot{J}_a(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - T_J(\boldsymbol{\phi})\boldsymbol{W}_{ext}\right) \\ &= J_a^{-1}(\boldsymbol{q})\left(\ddot{\boldsymbol{x}}_d + M_d^{-1}\left(K_D\dot{\tilde{\boldsymbol{x}}} + K_P\tilde{\boldsymbol{x}}\right) - \dot{J}_a(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - M_d^{-1}T_J(\boldsymbol{\phi})\boldsymbol{W}_{ext}\right)\end{aligned} \tag{2-18}$$

In this case the (transformed) wrench in the last term was introduced to render the end-effector linearly compliant with respect to the equivalent forces. Combining these control laws defines the following closed-loop dynamics:

$$M_d\ddot{\tilde{\boldsymbol{x}}}_e + K_D\dot{\tilde{\boldsymbol{x}}}_e + K_P\tilde{\boldsymbol{x}}_e = T_J(\boldsymbol{\phi}_e)\boldsymbol{W}_{ext} \tag{2-19}$$

Notice that rotational impedance still depends on the end-effector's current orientation through $T_J(\boldsymbol{\phi}_e)$. Here lies another possible application of VIL. Decoupled Cartesian stiffness parameters is critical when having to specify impedance parameters by hand. In VIC however, the time-varying gains are tailored to the higher level goal. For sufficiently expressive policies, the coupling can be implicitly incorporated into the controller.

Finally a note on practical implementation on commercial systems. For good performance it is critical that the torque commanded to the motors is close to the actual torque delivered to the joints. Due to effects such as friction, this is not always the case. Hence IC works best on torque controlled systems, such as the KUKA LWR iiwa. For the more common position-based manipulators, an admittance is defined that maps the desired Cartesian torques to spatial commands presented to the inner control loop. Hence the performance of these systems is highly dependent on the bandwidth of the inner position loop.
A promising research effort has been made at the '*DLR Institute of Robotics and Mechatronics*', where they have developed a unified framework based on passivity theory for force, torque and impedance control of flexible joint robots, detailed in [55]. They showed that an inner torque feedback loop can be incorporated into a passivity based analysis by interpreting torque feedback in terms of shaping of the motor inertia. They implemented these architectures on a Deutsches Zentrum für Luft- und Raumfahrt - German Aerospace Center (DLR) flexible-joint Manipulator and highlighted its performance in completing a piston-insertion task, wiping a table and opening a door. Due to the cooperation between the DLR institute and KUKA, some of these advances in hardware and software have made its way into the KUKA iiwa series. This light weight torque-controlled 7-DOF arm is therefore highly suited for impedance controlled tasks and is utilized in this thesis to perform experimental studies, documented in Chapter 5.

2-3 Force Guided Peg-in-Hole Insertion

Estimates from industry suggest that more than 30% of all industrial procedures can be classified as part mating [18]. A typical example of such tasks is PiH insertion. PiH is a category of assembly tasks in which a generally shaped peg is inserted in a suitably sized hole. Humans are incredibly adept at performing such tasks at great speed and accuracy. For modern robots however, these procedures are immensely difficult as eluded to in Chapter 1. This is because when two parts mate, there will always exist small errors in the positioning of

both parts. These arise from inaccurate sensing, uncertainty in the mechanical parameters of the manipulator and parts or from the dynamics of the control algorithm (see Chapter 5-2).

For a colourful - admittedly speculative - example, we present an application in the field of interplanetary robotics. Specifically that of an interplanetary rover equipped with a multi-DOF force-sensing arm and optical imaging sensor. These robots will soon be expected to perform construction and repairs on both themselves and their surroundings on distant planets. The robot might autonomously decide that a certain interaction is necessary and ask for confirmation from an earth-based control center. Due to inherent time delays the operators can not perform the task remotely through haptic tele-operation. This is but one outline of a scenario in which time for the agent to 'think' is abundant, but the risk for irreversible damage in the face of unknown contact dynamics to the distant and expensive rover is extremely high. Increasing data-efficiency and minimizing interaction time then becomes a critical requirement, while computational complexity constraints can be greatly relaxed. For a rover stationed on mars, the round-way trip for communication can quickly be up to an hour. As we will see in Chapter 5, in this time the robot can have reached an almost converged policy for the task with just a few roll-outs using the proposed algorithm. Even for relatively simple insertion tasks, an exact estimate for the goal state might be unavailable. Increasing the physical intelligence of the robot (see Chapter 4) alleviates the requirement for expensive, weighty and fault-vulnerable camera systems. In this setting, the robot can infer a rough initial estimate of the hole location through on board image recognition modules. In such a setting, compliance in the closed-loop system, whether passive [56] or active [57], is of paramount importance for successful insertion.

Traditionally, several phases in PiH can be identified; (1) an initial *approach* followed by (2) a *search* for the hole's location together with (3) an *alignment*, the actual *insert* (4) and finally (5) a *retreat*.
The PiH can be solved by straightforward completion of these individual phases, such as in [57]. The *approach* phase includes possibly picking up the relevant part and moving towards the vicinity of the hole. The initial contact with the surrounding environment happens during this approach (see Chapter 5-2-1). Next the robot will start its *search* for the hole. Often this is done by following a predefined motion plan [58] and monitoring for tell-tale signs that hint at the presence of the hole (see Chapter 4-1 and 5-1). When the orientation of the peg is relevant for the insertion[10] an additional path is defined for the orientation of the TCP. In theory, when placing the desired position of the TCP behind the hole and repeating these exploratory motions, successful insertion can be achieved, even for non-symmetric pegs. Such an approach was experimentally demonstrated in which a square peg is inserted in a hole with a clearance of 0.05 [mm] on either side [59]. An advantage of this simple approach is that there is no need for of force sensing from the robot's wrist (see Chapter 5-4-2). This essentially combines the search, align and insert phase into a single procedure, although it resulted in jamming more than 20% of trials, tripling insertion time, with the risk of damaging the mated part or manipulator. For more agile and rapid insertion, an additional *alignment* can be performed. A similar motion plan as before is executed where now a force sensor monitors for any signs of the hole's location. These moment measurements contain clues to the location and orientation of the hole. Theoretically, a single measurement made in the appropriate *capture region* is enough to deduce the size and location of the hole. Ideally, this mapping $(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{y}) \rightarrow \boldsymbol{W}_{ext}$ is well behaved for this capture region. To achieve this mapping

[10] This is the case for non-symmetric pegs

Newman [60] performed multiple roll-outs around the hole, measuring position and force signals along the trajectory (see Chapter 5-5). To obtain the inverse mapping from force to position command, the data was presented to a neural network trained by back-propagation. However, the moments observed along the trajectory are not unique and hence inversion of the observed function is problematic. Instead of directly using single force measurements, a higher level feature map can be build to infer the hole's location. Other more advanced strategies have since been developed for estimating hole location within the capture region based on force measurements. When the peg is properly aligned with the hole, *insertion* can start. During this time, the robot sets its desired TCP location somewhat behind the hole until the parts are successfully mated. In certain scenarios such as the one considered in this thesis, an additional stiffness is encountered during insertion (see Chapter 5-2-1), for example when inserting an electric plug into a socket or click-mating two mechanical parts. When insertion is completed, the gripper may detach from the peg and *retreat* back to its starting pose. Most of these five phases will show up in the problem description of Chapter 5. In essence what the proposed strategy attempts to do is combine all phases into a single learning problem by defining a high level goal (see Chapter 3-4-1) that steers the agent towards viable policies.

Chapter 3

Probabilistic Inference for Learning

Traditional control system design has allowed highly accurate control, often at high speeds and repeatability. Much of the success of strategies such as PID and optimal control design can be attributed to their solid mathematical foundation. Often the environments that allow for these rigorous derivations are assumed to be linear, stationary and exhibiting low uncertainty. Several issues arise when attempting to apply such methods for complex manipulation tasks or systems containing highly non-linear or uncertain dynamics. An alternative approach for these types of tasks is Reinforcement Learning (RL). In RL an agent learns a control strategy by interacting with its environment and, akin to the way humans learn, the system tries to improve its behaviour by minimizing a (long-term) cost in a trial-and-error fashion [4].

There are several challenges with respect to RL in the context of robotics that need to be addressed if one attempts to formulate a tractable problem. This chapter will begin by introducing some basic concepts of RL and highlight some of the challenges specific to robotic interaction tasks in Chapter 3-1, including the advantages and pitfalls of models in Chapter 3-1-1. Chapter 3-2 will introduce a Bayesian approach to model-based learning called Probabilistic Inference for Learning COntrol (PILCO) [13] that is able to mitigate some of the difficulties and decrease the required interaction time for the agent. The mathematics underlying planning with Gaussian Processes is presented in Chapter 3-3. Chapter 3-4 briefly details how policy updates are performed, although the exact structures suitable for learning impedance control is saved for Chapter 4.
Note that some of the nomenclature has changed slightly with respect to the first chapter. To stay congruent with most of the RL literature, $\boldsymbol{s}$ and $\boldsymbol{a}$ are used for states and actions respectively. J_π represents the total return over a trajectory $\boldsymbol{\tau}$. Notice that the objective measure in this work is in the form of a cost function, as opposed to its conventional inverse: reward.

3-1 Reinforcement Learning - a Brief Introduction

RL is a trial-and-error learning strategy, popularized by the work of Sutton & Barto [4]. The goal of RL is making an *agent* achieve a certain *goal* through executing *actions* by maximizing its reward[1] over time by interacting with its *environment*, as schematically framed in Figure 3-1.

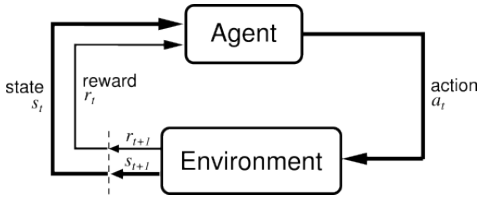


Figure 3-1: *Simple representation of the reinforcement learning framework [4].*

At each time step t the agent interacts with the environment by executing a specific action $\boldsymbol{a} \in \mathbb{R}^{D_u}$ which changes the state $\boldsymbol{s} \in \mathbb{R}^{D_s}$ of the system as well as presenting a cost $c \in \mathbb{R}$. At each time step, the agent implements a mapping from state representations to probabilities of selecting each possible action. This (possibly stochastic) conditional mapping is called a *policy* π, often represented by a parametrized function through $\boldsymbol{\psi}$:

$$\pi_{\boldsymbol{\psi}}(\boldsymbol{s}, \boldsymbol{a}, t) = p\left(\boldsymbol{a}_t | \boldsymbol{s}_t\right)$$

The system under consideration is a (Finite) Markov Decision Process (MDP)[2] [61]. Which is defined as a (non-linear) one-step ahead, state-space transition model with independent and identically distributed (i.d.d.) Gaussian noise σ_o:

$$\boldsymbol{s}_{t+1} = \boldsymbol{f}(\boldsymbol{s}_t, \boldsymbol{a}_t) + \sigma_o \tag{3-1}$$

Following a certain policy will result in a subsequent state, according to some (stochastic) transition function $p(\boldsymbol{s}_{t+1} = \boldsymbol{s}' | \boldsymbol{s}_t, \boldsymbol{a}_t)$, leading the agent to an expected numeric penalty:

$$C = \mathbb{E}\left[c_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}\right]$$

One trial lasting N_T steps of t_s seconds, is defined by the trajectory $\boldsymbol{\tau} = \{\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1 \dots \boldsymbol{s}_{N_T}\}$. The objective measure judging the quality of a roll-out we call the *expected return* J:

$$J_\pi(\boldsymbol{\psi}) = \mathbb{E}[C(\boldsymbol{\tau}|\boldsymbol{\psi})] = \int C(\boldsymbol{\tau}) p_{\boldsymbol{\psi}}(\boldsymbol{\tau}) d\boldsymbol{\tau} \tag{3-2}$$

where $C(\boldsymbol{\tau}) = \sum_{t=0}^{T} \gamma^t \cdot c(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}, t)$ is the (possibly discounted through γ) accumulated cost during one episodic trajectory. Over the course of multiple roll-outs[3] this measure of success allows the agent to update its policy in a way that increases its expected return.

[1] For this thesis we actually work with the inverse reward, i.e. *cost*.

[2] This essentially means that all relevant information is retained in the state of the system. If an environment has the Markov property, then its one-step dynamics enable us to (stochastically) predict the next state and expected next cost given the current state and action.

[3] These interactive phases are also called *trials*, here and in other RL literature.

This manner of updating the policy without an explicit representation of a value function is known as *direct policy search*. These algorithms are pervasive in robotic applications, since the more traditional *value-based* methods do not perform well in those types of domains [24], also see Chapter 1-1-2. There are in fact several characteristics inherent to robotics that make successful implementation of RL more challenging. The survey by Kober [15] gives a comprehensive overview of concepts, successes and challenges in robotic RL. For this work, learning robotic impedance control, we can identify the following key challenges:

1. High dimensionality
2. Continuous state & action-spaces with local discontinuities
3. Partially observable Markov Processes
4. Noise and uncertainty in sensor data
5. For physical contact, interaction could be damaging or dangerous.

Within direct policy search strategies we can distinguish between *model-free* and *model-based* algorithms. The former relies on trials to obtain the relevant trajectory data in Equation (3-2). One popular methodology is to then use that data to estimate a gradient in which to policy updates.
Model-free policy search imposes only general assumptions on the entire learning process, but due to the sampling nature, they can be applied to policies with a moderate number of parameters. The objective of model-based methods essentially comes down to increasing the data-efficiency compared to model-free methods. Model-based methods do not update based directly on experience from the real world, but instead use the experience to formulate an internal model, upon which policy updates can then be performed, i.e. *planning*.

3-1-1 The Use of Models

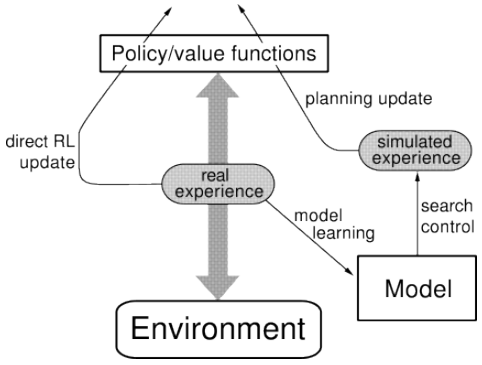


Figure 3-2: *Schematic diagram of the Dyna architecture introduced by Sutton in [4].*

Figure 3-2 gives a schematic overview of a strategy known as the Dyna-architecture, proposed by Sutton in [4]. The idea is that in addition to direct (model-free) RL, the same data can be used to construct a model for further policy improvements. Which in the case of the Dyna architecture can even be done on-line during the spare time of every control loop.

In model-based RL we can thus distinguish two phases: *model learning* and *planning*. There are different methodologies for all these phases and varying ways of combining them depending on whether the algorithm learns on-line or off-line in batch-mode or in real-time [42].

On top of the standard topics in RL such as parameter tuning, there are a few major components that need to be specifically addressed in model-based RL [24]; (1) what *model type* to learn: (non-)parametric, stochastic or deterministic. (2) How to use the model for *long-term prediction*: stochastic inference (e.g. PEGASUS [62]) or deterministic approximate inference (e.g. PILCO [13]) and lastly (3) the *policy update* method based on these predictions: gradient-based (e.g. REINFORCE [26]) or gradient-free (e.g. PoWER [31]).

One of the main performance criteria for model-based learning is *sample complexity*, or data-efficiency. Naturally the efficacy of model-based algorithms are highly dependent on the accuracy and expressiveness of the learned model. Errors in learned forward dynamics can not only cause degraded policies, but often drastically bias the learning process [15]. It is thus of crucial importance to account for model bias by accounting for the uncertainty about the underlying dynamics itself.
Typically in deterministic model-based learning, a certainty-equivalence assumption is made and planning is done based on the maximum likelihood model [42]. Moreover, many approaches obtain derivatives of the expected return by back-propagating derivatives through learned forward models of the system. This step is particularly prone to model errors since gradient-based optimizers improve the policy parameters along their gradients.

3-2 The PILCO Algorithm

The main weakness with any model-based algorithm is that of a false certainty-equivalence assumption, i.e. model bias. For on-line model-based control algorithms such as a resolved rate controller on a robotic arm, an erroneous model will often lead to decreased performance in terms of speed, accuracy or energy efficiency. For off-line learning algorithms however, the consequences for control based on erroneous models can be much greater. Due to the iterative nature of policy search algorithms, an inaccurate model can cause policy iterations to diverge into unstable regions, unbeknownst to the agent. This discrepancy will only manifest itself when the policy is applied to the real system for data acquisition. By then a possibly reckless policy could cause harm to the system or its environment. It would thus be desirable for learned models to contain a measure of certainty about their beliefs regarding the model in specific parts of the state space.

PILCO is such an algorithm which applies the concept of Gaussian Processs (GPs) to construct non-parametric models in a Bayesian fashion. These learned models are then used to perform long-term predictions and exhaustive planning. Pseudo-code for PILCO is given in Algorithm 1. The lines of the code supplemented with forward references are presented in more detail in the respective sections of this chapter.

The algorithm can be decomposed into two phases: `Interaction` and `Simulation`, as color-coded[4] in Algorithm 1. During an interaction trial, relevant trajectory data is recorded. This will include states, actions and possibly the immediate cost at each time step, although not

[4]Note that all results reported in Chapter 5 follow the same color scheme where red always refers to simulated distributions and blue is recorded data.

Algorithm 1: Pseudo-code for the PILCO algorithm.

Data: Initial roll-out data: $\boldsymbol{\tau}^{init}$
Result: Learned transition model and final policy

```
Initialize ψ & θ                                                          ▷ Sec. 5-1
repeat
    Learn GP dynamics model using all available data                      ▷ Sec. 3-3-3
    (Learn sparse GP model)                                               ▷ Sec. 4-4
    repeat
        Perform long-term predictions: s_t ~ N(μ_t, Σ_t) for t = 1,2,...T ▷ Sec. 3-3-4
        Evaluate policy: J_π(ψ)                                           ▷ Sec. 3-4-1
        Compute value gradient for policy improvement: dJ/dψ;             ▷ Sec. 4-2
        Update policy parameters w/ gradient-descent optimization         ▷ Sec. 3-4
    until convergence or max. iterations;
    Set π ← π(ψ^new)
    Record new roll-out data: τ* = {(s_0, a_0), (s_1, a_1), ..., s_T}
until task learned or max. iterations;
```

directly used for updates. This data is used to train a GP model, and to verify the current policy's performance in the real world.
When the hyper-parameters are conditioned on the recorded data, planning starts. During this phase, the GP model is used for making long-term predictions according to the methodology outlined in Chapter 3-3-4. These planned trajectories serve as input to an immediate cost and ultimately an expected return distribution. When taking care of certain smoothness constraints, the gradients to these fitness measures can be derived analytically. This is one of the main strengths of the PILCO algorithm. The efficiency of analytical optimization as opposed to gradient estimation schemes allows the usage of much richer and expressive policies with more parameters. Please refer to the references in Algorithm 1 for where the different concepts are detailed and implemented. Figure 3-3 shows the result of a comparison on a pendulum swing-up task for seven algorithms with respect to PILCO. A tremendous decrease in interaction time is seen. The price to pay for this improvement lies in the computational complexity of the algorithm (See the discussions in Chapter 4-4 and 5-6).

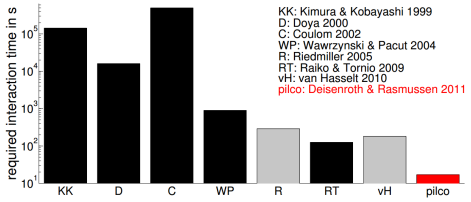


Figure 3-3: *Comparison of the required interaction time for policy convergence on a pendulum swing-up task. KK= Q-learning w/ local linear control, D= Continuous Actor-Critic, C= TD-λ with NNs, WP= Intensive Randomized Policy Optimizer (direct adaptive control), R= Neural-fitted Q-iteration, RT=Optimistic Inference Control, vH= Continuous Actor-Critic Learning Automata (CACLA).*

Chapter 3-3 will establish the mathematical definitions related to GPs. After this, a methodology for fitting the posterior GP to observed data is outlined. Finally a strategy for making predictions using the computed posterior at new and uncertain inputs is introduced. These

one-step-ahead predictions can then used for probabilistic simulations of dynamical systems, as explained in Chapter 3-3-4.

3-3 Gaussian Process Models

Supervised learning can be divided into two categories: regression and classification. Where classification deals with assigning data to discrete categories, regression relates observed data to continuous quantities. In the context of robotics, one example of non-linear regression is that of learning forward or inverse dynamics from recorded data.
In the following chapter, $\boldsymbol{x} \in \mathbb{R}^{N \times D}$ and $\boldsymbol{y} \in \mathbb{R}^{N \times 1}$ are used as the general independent (*regressor*) and dependent (*regressand*) variables, where N is the number of data points. Still $\boldsymbol{s}$ and $\boldsymbol{a}$ are specifically used to denote states and actions. For multiple target dimensions we will use E independent GPs.
A GP is a general class of probability distributions on functions. They can be seen as an infinite-dimensional generalization of multivariate normal distributions and can conveniently be used to specify very flexible non-linear regression. With proper care, GPs can identify the uncertainty coming from noisy data as well as the inherent uncertainty of the system.
The following definition is taken from [44]:

Definition 3.1. *Gaussian Process* A GP is a collection of random variables, any finite number of which have consistent joint Gaussian distributions

Given is the following process, governed by the latent (non-linear) function f:

$$y = f(\boldsymbol{x}) + \epsilon_o \tag{3-3}$$

Here, $\boldsymbol{x} \in \mathbb{R}^D$ and $f \in \mathbb{R}^D \to \mathbb{R}$ and $\epsilon_o \sim \mathcal{N}(0, \sigma_o^2)$ is identically distributed Gaussian white noise. This system can then be modeled by a GP:

$$\begin{aligned} f &\sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}_q, \boldsymbol{x}_p) \\ m(\boldsymbol{x}) &= \mathbb{E}[f(\boldsymbol{x})] \\ k(\boldsymbol{x}_q, \boldsymbol{x}_p) &= \mathbb{E}[(f(\boldsymbol{x}_q) - m(\boldsymbol{x}_q))\,(f(\boldsymbol{x}_p) - m(\boldsymbol{x}_p))] \end{aligned} \tag{3-4}$$

A GP is fully specified by its mean function $m(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}_p, \boldsymbol{x}_q)$. Hence whereas the Gaussian distribution is over vectors, the GP is over functions. The goal of GP regression is to find a posterior distribution of possible models, conditioned on observed training data using *Bayes' theorem* [63].

3-3-1 GP priors

The *prior mean function* $m(\boldsymbol{x})$ reflects the expectation of how the average is likely to behave. When using GPs for dynamical system identification it is common to assume a prior mean function $m(\boldsymbol{x}) = 0$. This work proposes an extension to non-zero priors, which we will return to in Chapter 4-1-1, but for now we assume $m(\boldsymbol{x}) = 0$.

There are many possibilities for the choice of covariance function. A general name for a function k of two arguments mapping a pair of inputs $\boldsymbol{x}_p, \boldsymbol{x}_q \in \mathcal{X}$ into $\mathbb{R}$ is a *kernel*. The kernel

fulfilling the role of covariance function must be symmetric positive-definite. These functions can be either stationary or non-stationary and degenerate or non-degenerate. Some stationary examples include a constant, (squared) exponential, Matern or rational quadratic. Non-stationary, degenerate examples are linear or (piecewise) polynomial covariance functions. A more complex non-stationary, degenerate covariance function is a neural network.

Given a set of input points $\{\boldsymbol{x}|i = 1 \dots N\}$, we can compute a *Gram matrix* K, with $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. When using kernels as covariance function this matrix is usually called the covariance matrix and is of crucial importance during predictions. The *consistency requirement* says that if a GP specifies some joint (y_1, y_2), it must also specify a distribution for just y_1. Handling a bigger dataset does not alter distributions of smaller subsets of that data. This is known as the *marginalization property* [44]. This requirement can be automatically fulfilled if the covariance function specifies the entries of the covariance matrix. However if one were to choose a covariance function specifying the entries of the inverse covariance matrix, the marginalization property would be violated and the resulting model would not be a GP. As we will see later, this means K^{-1} will have to be computed numerically.

For this thesis we will utilize a *squared exponential* covariance function[5] with Automatic Relevance Determination (ARD) [64] plus a noise covariance term:

$$k(\boldsymbol{x}_q, \boldsymbol{x}_p) = k_{SE} + \delta^{pq}\sigma_o = \sigma_f \cdot \exp\left(-1/2\left(\boldsymbol{x}_p - \boldsymbol{x}_q\right)^T \Lambda^{-1}\left(\boldsymbol{x}_p - \boldsymbol{x}_q\right)\right) + \delta^{pq}\sigma_o \tag{3-5}$$

Where σ_f controls the the typical amplitude of the function. The length-scale matrix $\Lambda = \text{diag}[\lambda_1, \dots, \lambda_n]$ can informally be thought of as the distance you have to move in input space before the function value can change significantly. The fact each input dimension has its own length-scale is known as ARD. If a particular input dimension d has no relevance for the regression, then the appropriate length-scale will increase to essentially filter that feature out. This is because the evidence will suggest that the underlying function is very slowly varying in the direction of that feature. This means that alternative ad-hoc feature selection methods are not necessary. σ_o is included to model the output noise ϵ_o in (3-3). The Kronecker symbol δ_{pq} is unity for $p = q$ and zero otherwise (independent noise contributions). All these characteristic parameters are known as *hyper-parameters* and are collected in the vector $\boldsymbol{\theta}$.

Additionally, the input locations and targets can be considered parameters to be optimized too. These are then often referred to pseudo-inputs and pseudo-targets. This gives the final hyper-parameter vector:

$$\boldsymbol{\theta} = \begin{bmatrix}\sigma_f & \sigma_o & \Lambda & \boldsymbol{x}_{in} & \boldsymbol{y}\end{bmatrix}^T \in \mathbb{R}^{n_\theta} \tag{3-6}$$

Given a certain $\boldsymbol{\theta}$, a GP posterior can be used for making predictions at unseen inputs.

3-3-2 GP Posteriors

Assume for one target dimension a training set of N observations: $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\,|i = 1, \dots, N\}$. Where $\boldsymbol{x} \in \mathbb{R}^D$ denotes an input vector and y_i denotes a scalar output or target. The vector of inputs for all N observations are aggregated in the *design matrix* $X \in \mathbb{R}^{N \times D}$. The targets for a given output dimension are collected in the vector $\boldsymbol{y} \in \mathbb{R}^N$, so we can write $\mathcal{D} = (X, \boldsymbol{y})$.

[5] This corresponds to a Bayesian linear regression model with an infinite number of basis functions

Bayesian theory calls for the use of the posterior predictive distribution to do predictive inference, i.e., to predict the distribution of a new, unobserved data point. That is, instead of a fixed point as a prediction, a distribution over possible points is returned. In order to compute the posterior distribution over possible models, we need to reject all realizations that do not agree with the data by employing *Bayes' Rule* [63]:

$$p(f|X, \boldsymbol{y}, \boldsymbol{\theta}) = \frac{Likelihood \cdot Prior}{Evidence} = \frac{p(\boldsymbol{y}|X, f, \boldsymbol{\theta})p(f|\boldsymbol{\theta})}{p(\boldsymbol{y}|X, \boldsymbol{\theta})} \tag{3-7}$$

The finite probability distribution denoted as the *likelihood* encodes an assumed noise model and is represented by a finite distribution:

$$p(\boldsymbol{y}|f, X, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(y_i|f(\boldsymbol{x}_i), \boldsymbol{\theta}) = \prod_{i=1}^{N} \mathcal{N}(y_i|f(\boldsymbol{x}_i), \sigma_o^2) = \mathcal{N}(\boldsymbol{y}|, f(X), \sigma_o^2 I_N) \tag{3-8}$$

Where for tractability we have assumed that the observations y_i are conditionally independent given X, allowing for the multiplication. Despite this likelihood being finite dimensional, the infinite dimensional GP prior makes that the posterior behaves as a GP as well.
The normalizing *evidence* $p(\boldsymbol{y}|X, \boldsymbol{\theta})$ is often termed a *marginal likelihood*, since f has been "marginalized" out by integration:

$$p(\boldsymbol{y}|X, \boldsymbol{\theta}) = \int p(\boldsymbol{y}|X, f, \boldsymbol{\theta})p(f|\boldsymbol{\theta}) \, \mathrm{d}f \tag{3-9}$$

When fitting the hyper-parameters to the observed data, this term is used as a measure for the fit in a two-level inference scheme, explained in Chapter 3-3-3.

3-3-3 GP Model Learning

In the previous section we defined a posterior distribution over f given a known set of hyper-parameters $\boldsymbol{\theta}$. For training the posterior distribution to reflect the observed data as closely as possible, we consider these hyper-parameters to be latent unknown variables. Following Bayes theorem, we could find the optimal set of parameters by placing a hyper-prior $p(\boldsymbol{\theta})$ on them and integrating them out. However this gives an intractable integral due to multiple dependencies on $\boldsymbol{\theta}$. So instead, we employ an optimization technique known as Evidence Maximization (EM) to find a good point estimate $\hat{\boldsymbol{\theta}}$.

Figure 3-4 shows the two-level inference scheme employed for learning $\boldsymbol{\theta}$. The EM procedure goes through this diagram bottom to top, starting with the data.

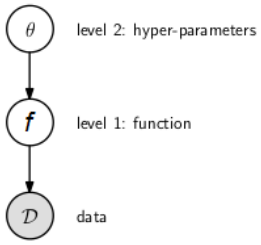


Figure 3-4: *Hierarchical model for 2-level Bayesian inference for GP model learning [5].*

The first level of inference gives a GP posterior on the latent function f, as defined in Equation (3-7). At the second level we analogously express a posterior distribution over the hyper-parameters given the data:

$$p(\boldsymbol{\theta}|X,\boldsymbol{y}) = \frac{p(\boldsymbol{y}|X,\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y}|X)} \tag{3-10}$$

Where we see that the evidence of level-1 now plays the role of likelihood at level-2. The prior in (3-10) is known as the *hyper-prior*, assumed to be 'flat' so as to not exclude possible models. An implicit advantage of this flatness is that the posterior in (3-10) is proportional to the likelihood, which was the evidence from level-2: $p(\boldsymbol{\theta}|X,y) \propto p(\boldsymbol{y}|X,\boldsymbol{\theta})$. As eluded to in the beginning, ideally we would maximize the evidence in this level-2 inference. However the integral in $p(\boldsymbol{y}|X) = \int p(\boldsymbol{y}|X,\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$ is analytically intractable. Due to the flat prior however, the *maximum a posteriori* estimate $\hat{\boldsymbol{\theta}}$ can be found by maximizing the *Log Marginal Likelihood (LML)* from Equation (3-7)

$$\begin{aligned}\log p(\boldsymbol{y}|X,\boldsymbol{\theta}) &= \log \int p(\boldsymbol{y}|f,X,\boldsymbol{\theta})p(f|\boldsymbol{\theta})\,\mathrm{d}f \\ &= -\frac{1}{2}\boldsymbol{y}^T\left(K+\sigma_o^2 I\right)^{-1}\boldsymbol{y} - \frac{1}{2}\log\left|K+\sigma_o^2 I\right| - \frac{D}{2}\log 2\pi\end{aligned} \tag{3-11}$$

The resulting optimized vector is known as a type-II maximum likelihood estimate:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|X,\boldsymbol{\theta})$$

Looking at the expression in Equation (3-11) we can see several characteristics that are penalized. The first term is a measure for data-fit. The second term is a complexity term that avoids over-fitting. It is this built-in trade-off between model-fit and model-complexity that makes the marginal likelihood measure such a useful one for model selection. The new parameter vector $\hat{\boldsymbol{\theta}}$ can now be used in the posterior prediction estimates of Equation (3-12). Of course, one should be careful with such an optimization step, since it opens up the possibility of over-fitting, especially if there are many hyper-parameters [44]. In the following section we will look at making short and long-term predictions using GPs in the case of time-series data.

3-3-4 Dynamic Simulation using Gaussian Processes

So far we have established the prior and posterior characteristics of a GP. In the context of RL, the true strength of GPs lies in long-term simulations of dynamical systems. This is done by cascading one-step-ahead predictions [5]. The following sections will detail how to use the posteriors found in Chapter 3-3-2 for making predictions at unseen and uncertain test points.

Predictions at Deterministic Inputs

From the definition of a GP the function values for training and test inputs are consistently jointly Gaussian. This allows writing the joint of training $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \sigma_o$ and test outputs $\boldsymbol{y}^*$

$$p(\boldsymbol{y},\boldsymbol{y}_*|X,\boldsymbol{x}_*) = \begin{bmatrix}\boldsymbol{y}\\ \boldsymbol{y}_*\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}K(X,X)+\sigma_o I & K(\boldsymbol{x}_*,X)\\ K(X,\boldsymbol{x}_*) & K(\boldsymbol{x}_*,\boldsymbol{x}_*)\end{bmatrix}\right) \tag{3-12}$$

where $\boldsymbol{y} = \begin{bmatrix} f(\boldsymbol{x}_1) + \epsilon_o \dots f(\boldsymbol{x}_N) + \epsilon_o \end{bmatrix}^T$ and $\boldsymbol{k}_* = K(X, \boldsymbol{x}_*) = K(\boldsymbol{x}_*, X)^T$ denotes matrix of the covariances evaluated at all mutual pairs of training and test points.

For a given $\boldsymbol{\theta}$ and using the prior, likelihood and evidence defined in Equation (3-5), (3-8) and (3-9) respectively, the conditional rule from Equation (A-12) results in the posterior:

$$p(\boldsymbol{f}_*|X_*, \mathcal{D}) \sim \mathcal{N}\left(\boldsymbol{\mu}_{pred}, \boldsymbol{\Sigma}_{pred}\right) \tag{3-13}$$

where

$$\begin{aligned} \boldsymbol{\mu}_{pred} &= \mathbb{E}_f[f(\boldsymbol{x}_*)|X, \boldsymbol{y}, \boldsymbol{\theta}] = k(\boldsymbol{x}_*, X)\left(K(X,X) + \sigma_o^2 I\right)^{-1} \boldsymbol{y} \\ &= \boldsymbol{k}_*^T \left[K(X,X) + \sigma_o^2 I\right]^{-1} \boldsymbol{y} \\ &= \boldsymbol{k}_*^T \boldsymbol{\beta}^T \end{aligned} \tag{3-14}$$

$$\begin{aligned} \boldsymbol{\Sigma}_{pred} &= \mathrm{Cov}[f(\boldsymbol{x}), f(\boldsymbol{x}_*)|X, \boldsymbol{y}, \boldsymbol{\theta}] \\ &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{x}_*, X)\left(K(X,X) + \sigma_o^2 I\right)^{-1} k(X, \boldsymbol{x}_*) \\ &= \boldsymbol{k}_{**} - \boldsymbol{k}_*^T \left[K(X,X) + \sigma_o^2 I\right]^{-1} \boldsymbol{k}_* + \sigma_o^2 I \end{aligned} \tag{3-15}$$

The mean in Equation (3-14) is the result of a linear combinations of observations $\boldsymbol{y}$. Also note that the predicted covariance is solely a function of the inputs, and not the targets. The first term is the prior covariance of the GP. The second term represents the gained insight on the variance through the data. Since this latter term is positive definite, the posterior variance will always be lower than that of the prior.
Bayesian inference can thus be considered a three-step procedure: First, a prior on the unknown quantity is specified. In our case, this is the function $\boldsymbol{f}$ itself. Then data is observed and lastly a posterior distribution over $\boldsymbol{f}$ is computed that refines the prior by incorporating evidence from the observations.

Predictions at Uncertain Inputs

For our case, the input at time t will be the prediction from $t-1$. This section will focus on making predictions for $\boldsymbol{x}_* \in \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$. In the case of multiple target dimensions ($\boldsymbol{y} \in \mathbb{R}^{N \times E}$) we can train E independent GPs. Generally, mapping a Gaussian distributed input through a non-linear GP is done by averaging over the input. As can be seen by the green shaded area in Figure 3-5, simply averaging over the distribution of $\boldsymbol{x}_*$ yields a result that is not Gaussian. Therefore instead, we approximate the result by a Gaussian and compute the moments of the predictive distribution exactly. This is known as *moment matching* and is done by averaging over both the input and the distribution of the function represented by the GP. The *predictive mean* at uncertain inputs can then be computed using the law of iterated expectations (See Appendix A):

$$\begin{aligned} \mu_{pred} &= \int\int p(f, \boldsymbol{x}_*) f(\boldsymbol{x}_*)\, \mathrm{d}(f, \boldsymbol{x}_*) \\ &= \mathbb{E}_{\boldsymbol{x}_*}\left[\mathbb{E}_f\left[f(\boldsymbol{x}_*)|\boldsymbol{x}_*\right]\right] \\ &= \mathbb{E}_{\boldsymbol{x}_*}\left[m_f(\boldsymbol{x}_*)\right] \\ &= \boldsymbol{\beta}^T \boldsymbol{q} \end{aligned} \tag{3-16}$$

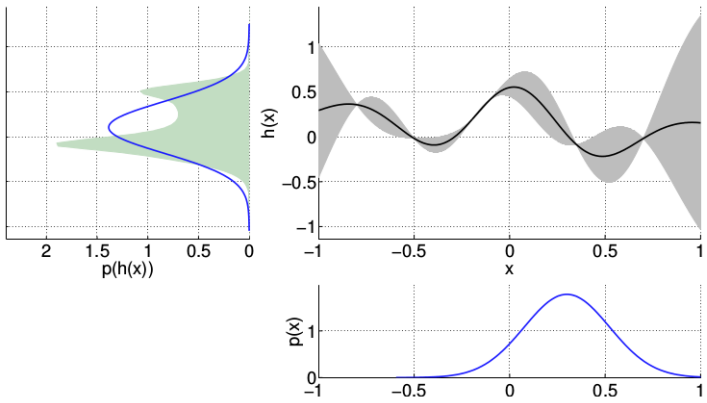


Figure 3-5: *GP predictions at an uncertain input (blue line in lower plot). The exact predictive distribution (left panel, shaded area) is approximated by a Gaussian (left panel, blue line) after mapping through the non-linear dynamics (top right) [5].*

Where q_i is the expected covariance between the function values[6] $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_*)$:

$$\begin{aligned} q_i &= \int k(\boldsymbol{x}_i, \boldsymbol{x}_*) p(\boldsymbol{x}_*) \, \mathrm{d}\boldsymbol{x}_* \\ &= \sigma_f^2 \left| \Sigma_* \Lambda^{-1} + I \right|^{-1/2} \exp\left(-1/2 (\boldsymbol{x}_i - \boldsymbol{\mu}_*)^T (\Sigma_* + \Lambda^{-1})^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_*) \right) \end{aligned}$$

For multivariate targets, each mean is simply computed separately. For the *predictive variance* we can construct the following matrix:

$$\Sigma_{pred} = \begin{bmatrix} \sigma_{11}^2 & \cdots & \sigma_{1m}^2 \\ \vdots & \ddots & \vdots \\ \sigma_{m1}^2 & \cdots & \sigma_{mm}^2 \end{bmatrix}$$

Now clearly it matters whether there are multivariate targets, since these will start to covary on the off-diagonals. First we derive the predictive variance of a single target dimension σ_{ii}, again by integrating over the input and function distributions. This time with help of the law of total covariance (See Appendix A):

$$\begin{aligned} \sigma_{ii}^2 &= \mathrm{Var}_{\boldsymbol{x}_*, f_i} \left[f_i(\boldsymbol{x}_*) | \boldsymbol{\mu}_*, \boldsymbol{\Sigma}_* \right] \\ &= \mathbb{E}\left[\mathrm{Var}_f \left[f_i(\boldsymbol{x}_*) | \boldsymbol{x}_* \right] \right] + \mathrm{Var}_{\boldsymbol{x}_*} \left[\mathbb{E}\left[f(\boldsymbol{x}_*) | \boldsymbol{x}_* \right] \right] \\ &= \mathbb{E}\left[\sigma_{f_i}^2 \right] + \mathbb{E}\left[m_{f_i}(\boldsymbol{x}_*)^2 | \boldsymbol{x}_* \right] - \mathbb{E}\left[m_{f_i}(\boldsymbol{x}_*) | \boldsymbol{x}_* \right]^2 \\ &= \mathbb{E}\left[\sigma_{f_i}^2 \right] + \mathbb{E}\left[m_{f_i}(\boldsymbol{x}_*)^2 | \boldsymbol{x}_* \right] - (\mu_{pred}^{(i)})^2 \end{aligned} \tag{3-17}$$

[6] Conflated by the input's uncertainty $\boldsymbol{\Sigma}_*$.

The variance of the off-diagonal terms will have one term less due to the assumption of independence for a given $\boldsymbol{x}_*$. But as a whole are still non-zero due to input uncertainty.

$$\begin{aligned}\sigma_{ij}^2 &= \text{Cov}\left[f_i(\boldsymbol{x}_*), f_j(\boldsymbol{x}_*)\right] \\ &= \mathbb{E}_{f,\boldsymbol{x}_*}\left[f_i(\boldsymbol{x}_*)\cdot f_j(\boldsymbol{x}_*)\right] - \mu_{pred}^{(i)}\mu_{pred}^{(j)} \\ &\quad\vdots \\ &= \beta_i^T\left(\int k_f^{(i)}(\boldsymbol{x}_*, X)\cdot k_f^{(j)}(\boldsymbol{x}_*, X)\cdot p(\boldsymbol{x}_*)d\boldsymbol{x}_*\right)\beta_j - \mu_{pred}^{(i)}\mu_{pred}^{(j)} \\ &= \beta_i^T Q\beta_j - \mu_{pred}^{(i)}\mu_{pred}^{(j)}\end{aligned} \tag{3-18}$$

For more in depth derivations the interested reader is referred to [5].

Probabilistic Long-Term Predictions

For policy evaluation it is required to compute the full state trajectory $\boldsymbol{\tau} = \{\boldsymbol{s}_0, \boldsymbol{a}_0, \dots \boldsymbol{s}_T\}$ through deterministic approximate inference. This amounts to solving the integral

$$p(\boldsymbol{s}_t) = \int\int\int p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})p(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})\,\mathrm{d}\boldsymbol{s}_{t-1}\,\mathrm{d}\boldsymbol{a}_{t-1}\,\mathrm{d}\sigma_o \tag{3-19}$$

In the previous chapter it was detailed how to make predictions at deterministic and uncertain inputs. PILCO needs to propagate uncertainties since even for a deterministically given input the GP dynamics model returns a Gaussian predictive distribution to account for the model uncertainty. These trajectories are found by cascading one-step-ahead predictions. Predicting the subsequent state distribution involves several steps, outlined in (3-20).

$$p(\boldsymbol{s}_{t-1}) \xrightarrow[1]{exact} p(\boldsymbol{a}_{t-1}) \xrightarrow[2]{approx.} p(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}) \xrightarrow[3]{MM} p(\boldsymbol{s}_t) \tag{3-20}$$

Step 1 involves computing the *distribution over controls*: $p(\boldsymbol{a}_t) \in \mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)$, where the policy input $\boldsymbol{s}_\pi(t)$ is a subset of the GP outputs. During roll-outs these states are deterministically sensed and fed to the controller. During a planning iteration however, they are distributions. Hence to properly account for this uncertainty, a distribution over controls is computed. This imposes a requirement on the policy structure that this distribution should be analytically available.

In step 2 we compute the joint distribution $p(\boldsymbol{s}_t, \boldsymbol{a}_t)$, which serves as the input to the GP. When using a concatenated saturating controller (see Chapter 4-3) this involves two actions. First the joint distribution between the state and unsaturated control is computed: $p(\boldsymbol{s}_t, \hat{\pi})$. Next the joint $p(\boldsymbol{s}_t, \hat{\pi}, \pi)$ is computed, after which the unsaturated control signal is marginalized out to arrive at the required joint state-action distribution.

Step 3 involves taking this joint probability as the input to the learned GP. As was shown in Chapter 3-3-4, the result is no longer Gaussian and moment matching[7] is employed.

When learning relative state differences and possibly non-zero priors are used, an additional step 4 has to be included where we account for all the cross-covariances between the state-action input pair and the predictions. More on this in Chapter 4-1-1.

[7] Alternatively one could linearize the GP for an analytical moments, which is computationally advantageous but the result is not in general the exact predictive moments of the real distribution.

We have now arrived at the distribution of the state at time t, which can now be fed back to the beginning and used as the initial condition for the subsequent time step. As in any dynamical simulation, due to the sequential nature of this procedure, it is not possible to parallelize the computations in any significant manner.

3-4 Policy Updates

The previous sections outlined the way in which GPs can be used to formulate analytical long-term predictions of the state evolution during a trial. What rests is to formulate a policy structure that also allows for analytical expressions of its (saturated) outputs, given a distributed input. The simplest possible policy is a straightforward affine linear combination of the inputs.

$$\tilde{\pi}_{Lin}(\boldsymbol{s}_\pi) = A\boldsymbol{s}_\pi + \boldsymbol{b} \tag{3-21}$$

Resulting in a *policy parameter vector* $\boldsymbol{\psi}_{Lin} = [A\ \boldsymbol{b}]^T$. Quasi-Newton gradient-descent optimization as detailed in Appendix A-2 is then used to update $\boldsymbol{\psi}$. The measure used for guiding this search is the expected return, a summation of immediate costs over a trajectory, as detailed below. Non-linear policies are possible and the policy structures and saturation functions used in this work are dealt with in Chapter 4-3.

3-4-1 Cost Function and Expected Returns

The total cost over an entire episode is referred to as the *expected return* of that episode. It is the sum of all immediate costs received at each time step. When taken care of smoothness constraints throughout the algorithm, a learned GP model allows the derivation of analytical gradients with respect to the value function. The immediate target cost $c_s(\boldsymbol{s}_t)$ can be any (piecewise) differentiable function; linear, quadratic or saturating. The inputs are any or all output dimensions of the GP model. Being careful to properly account for all the uncertainties and gradients, multiple cost functions can be stacked on top of each other, possibly acting on different parts of the state space.

The expected return in PILCO is defined in (3-22). It is composed of two terms, summed over the entire trajectory.

$$J_{\pi_{\boldsymbol{\psi}}}(\boldsymbol{s}_0|w_1, w_2)7 = \sum_{t=0}^{T} \Big[w_1 \cdot \mathbb{E}\left[c_s(\boldsymbol{s}_t)\right] + w_2 \cdot \sigma\left[c_s(\boldsymbol{s}_t)\right] \Big] \tag{3-22}$$

$$= \sum_{t=1}^{T} \left[w_1 \mathcal{E} + w_2 \mathcal{S} \right] \tag{3-23}$$

The first term $\mathcal{E}$ is the target cost, penalizing deviations from the desired goal state.
The second term $\mathcal{S}$ is an exploration term. It is defined as the standard deviation of the cost function. During long-term predictions, the state uncertainty will be reflected in the cost $c \sim \mathcal{N}(\mu_c, \sigma_c^2)$. When $w_2 < 0$, uncertainty in the predictive states will drive the overall cost down and hence incentivize exploration in that space. The weight w_2 can thus be used to tune the exploration-exploitation trade-off. It also smooths out the optimization

numerically, increasing speed. When w_2 is positive, the system is inclined to exploit more certain trajectories.

For this thesis the expected return was modified in order to accommodate Variable Impedance Learning (VIL). Chapter 4-2 gives the exact definition along with the expanded derivation of that return function. For now it suffices to say that the policy-dependent parameter vector $\boldsymbol{\psi}$ is updated through analytical gradient-descent in the direction that minimizes an expected return similar to that in Equation (3-22). The specific analytical minimization techniques used in this thesis are Limited-memory Broyden-Fletcher-Gretchen-Shanno ((L)BFGS) and Conjugate Gradient (CG), which are described in Appendix A-2. Both these techniques are *line-search methods*, which are iterative algorithms to compute (local) minima. In general, each iteration k of a line-search algorithm computes a search direction $\boldsymbol{p}_k = \nabla J_\pi(\boldsymbol{\psi}_k)$ and a positive step length α_k along the search direction that satisfies a sufficient decrease in the function as measured by the inequality

$$J(\boldsymbol{\psi}_k + \alpha_k \boldsymbol{p}_k) \leq J(\boldsymbol{\psi}_k) + c_1 \alpha_k \boldsymbol{p}_k$$

Chapter 4

Learning Variable Impedance Control

We shall now combine the theoretical groundwork laid out in the previous chapters to formulate a methodology for learning Variable Impedance Control (VIC). The ideas articulated here are original to this work with a few well indicated exceptions.
For this thesis, two skill-learning problems are investigated. The first scenario is termed Variable Impedance Learning (VIL). For this problem, appropriate state-conditioned stiffness parameters are learned along predefined kinematic trajectories so as to complete the task with an optimal trade-off between accuracy and minimization of contact forces.
The second scenario is called Variable Impedance Learning with Trajectory Adaptation (VILTA), for a task where neither the trajectory nor the desired end-pose is known and the agent is additionally expected to learn an optimal motion plan in Cartesian space.

The constrained motion task chosen for this work is a variant of the Peg-in-Hole (PiH) problem described in Chapter 2-3. To return to our imagined example, imagine again an industrial robot or interplanetary rover set to perform an unknown mating procedure of two parts. Several degrees of abstraction can be identified with increasing problem complexities. The first scenario is where both the initial condition $\boldsymbol{x}_0$ and the target $\boldsymbol{x}_{goal}$ are constant. In this case, VIL is used to compute an optimal stiffness trajectory that achieves its goal while minimizing contact forces and energy expenditure through time-varying stiffness gains. When properly trained, the agent can accommodate for small changes in the goal's location by allowing deviations from the reference that yield higher reward.
A more difficult scenario is when the location of the goal is known only to a lower degree of certainty: $\boldsymbol{x}_{goal} \in \mathcal{N}(\boldsymbol{\mu}_{goal}, \Sigma_{goal})$, likely due to imperfect measurement. For this case, VILTA attempts to formulate a motion plan that achieves insertion, effectively learning the exact location of the hole, while additionally optimizing the closed-loop stiffness gains. One additional abstraction is investigated in this work, namely that of variable start states $\boldsymbol{x}_0 \in \mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$. For which the agent is expected to learn VILTA, regardless of its initial condition. Finally one can imagine the fully general scenario, where both the start *and* goal state vary between each episode. Even though the architecture of the proposed algorithm allows for such problems, it is not experimentally verified for reasons discussed in Chapter 5-6.

Figure 4-1 contains a block diagram showing the complete structure of the VIL problem. Solid lines indicate direct signal flow, while dashed lines indicate learned adjustments to the parameters inside the respective blocks. The subsystem labelled Impedance Controller contains the Simple Impedance Control (SIC) or General Impedance Control (GIC) laws from Chapter 2-2. For VILTA, the references are part of the policy blocks, instead of being externally supplied. Descriptions and justifications of the different concepts relevant to this diagram are presented in this chapter.

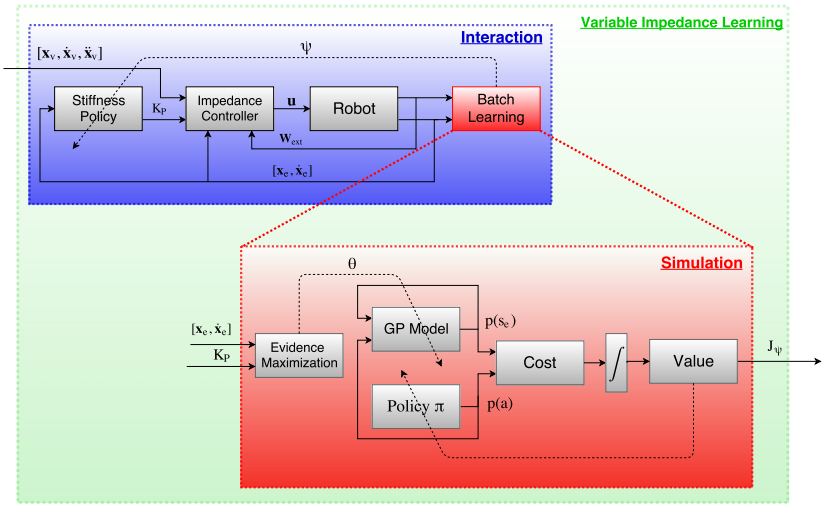


Figure 4-1: *Block diagram of the VIL algorithm. Data is recorded during the interaction phase and passed to the Learning block. Policy updates based on probabilistic long-term predictions occurs inside the simulation sub-system. The block diagram shown here is specific to the VIL problem. For VILTA, the virtual trajectory signal* $[\boldsymbol{x}_v \;\; \dot{\boldsymbol{x}}_v \;\; \ddot{\boldsymbol{x}}_v]$ *is suggested by the policy.*

4-1 GP Model and Actions

Wolpert in [19] notes three ways in which humans make Bayesian sense of the stream of sensory inputs; First, multiple channels (for example, visual and tactile inputs) are optimally combined to achieve estimates that reduce the effects of noise. Interestingly, this integration process can take into account the properties of external objects, so that the visuo–haptic-integration is optimal even through a tool. Second, by learning the statistical distribution of possible states of the world, in our case through the Gaussian Process (GP) prior (see Chapter 4-1-1). Lastly, by combining these processes with internal models of the body that map the motor commands into the expected sensory inputs, Bayesian inference can be used to estimate the evolving state of our body and the world. These observations suggest GP models are a good choice for mimicking a human approach to sensorimotor control.

Chapter 3-3 gave the definition of the system dynamics to be modeled in terms of regressor and regressand variables:

$$\boldsymbol{y}^{(i)} = f^{(i)}(\boldsymbol{x}) + \epsilon_o \tag{4-1}$$

In general we can say that $\boldsymbol{x} = \left(\boldsymbol{s}_t', \boldsymbol{a}_t\right) \in \mathbb{R}^{N\times(D_s+D_u)}$ and $\boldsymbol{y} = \boldsymbol{s}_{t+1} \in \mathbb{R}^{N\times E}$, where

the apostrophe indicates a subset of the outputs[1]. Typical benchmarks for Reinforcement Learning (RL) algorithms are problems such as the cart-pole balancing or swing-up of the single and double pendulum. For these problems the state- and input vector are chosen in joint space. This make sense in that they are the lowest level of state representations that define the dynamics of the system.
For VIL, several considerations have to be made. Namely what states and actions to use for the GP regressor variables, which subset of the state to use as targets and finally the type of output for the policy. Which does not necessarily have to lie in the same space. Presented next is the rationale for choosing the contents of $\boldsymbol{x}$ and $\boldsymbol{y}$.

For $\boldsymbol{s}$ the options are either the joint- or Cartesian state vector. The former has the advantage that angles are mapped to finite intervals, thereby reducing the range that states can take while still encapsulating all dynamics. Another consideration is that of redundancy and non-uniqueness. Alternative null-space configurations might result in different behaviour at the Tool Center Point (TCP)[2]. This dependence would be encoded in the joint-space formulation, but possibly be lost in the Cartesian one. Though the Cartesian state vector has a length that is shorter by twice the dimensionality of the redundancy.
However the most important consideration lies in the relation between the output of the policy and the control inputs to the GPs. The control law in Chapter 2-2-2 can be reduced to $\boldsymbol{u}_t = \boldsymbol{u}_{imp} + \boldsymbol{u}_{dyn}$. If any part of $\boldsymbol{u}$ is chosen as control input to a (joint-space) GP model, this would require the output of the policy to preferably be in this same space: $\pi(\boldsymbol{s}_\pi) = \boldsymbol{u}_{t/imp}$. The advantage is that the distribution over controls is easily computed. An important weakness of this formulation is that the resulting policy will be black-box controllers. We can not deduce the impedance characteristics of these controllers by looking at their outputs. A solution could be an intermediate layer between a policy suggesting high-level stiffness gains and control inputs to the GPs. However this would greatly increase the complexity of computing the distribution over controls, which is then a combination of multiple interdependent Gaussians.

Additionally we have to honour some real world implementation constraints. In computer simulations, all states and inputs are theoretically available at any time step. For commercial manipulators this is often not the case. Chapter 2-2-2 already hinted at some pitfalls when designing impedance controllers for real robots. It was found that the torque-controlled robots have been a major improvement over position-controlled systems. However, for safety and usability, the commands send to commercial robots are not torques, but high-level instructions for the proprietary controllers to interpret and translate into signals applied to the motors. This means that the joint-space implementation is infeasible. The joint torques τ_i are available, but these will be inflated with both $\boldsymbol{\tau}_{dyn}$ and $\boldsymbol{\tau}_{ext}$ and there is no way of extracting the signal we are interested in. This would result in erroneous inputs during long-term inference and planning. For these reasons we thus define a high-level stiffness policy as:

$$\pi(\boldsymbol{s}_\pi)_{VIL} = \boldsymbol{k}_P \tag{4-2}$$

Where $\boldsymbol{k}_P = \text{diag}\,[K_P] \in \mathbb{R}^D_u$ depends on the problem and dimensionality and $\boldsymbol{s}_\pi$ is the subset of $\boldsymbol{y}$ that serves as input to the policy and can be freely chosen. The complete model becomes:

$$\boldsymbol{x}_{VIL} = \begin{bmatrix} \boldsymbol{s} & \boldsymbol{a} \end{bmatrix}^T = \begin{bmatrix} \begin{pmatrix} \boldsymbol{s}'_e, & t \end{pmatrix} & \boldsymbol{k}_P \end{bmatrix}^T \qquad \boldsymbol{y}_{VIL} = \begin{bmatrix} \boldsymbol{s}_e & \boldsymbol{W}_{ext} \end{bmatrix}^T \tag{4-3}$$

[1] In order to be available as input at time t, it needs to be part of the outputs at $t-1$.
[2] This is especially true for non-quaternion orientation control, see Appendix B.

Where $\boldsymbol{s}_e^{'}$ denotes a desired subset of $\boldsymbol{s}_e$, which itself depends on the dimensionality of the problem. The external forces are not a part of $\boldsymbol{x}$, since in closed-loop these will be a direct consequence of the states, gains and reference. They can thus be inferred without serving as inputs. Notice the inclusion of time for VIL in $\boldsymbol{x}$. Since we are doing closed-loop identification, omitting the driving references would result in conflicting state-transitions when identical states are encountered at different times during a roll-out[3]. However concatenating $\boldsymbol{x}_{VIL}$ with all references would increase the dimensionality too quickly. Instead what we do is have time t serve as an input. Since reference trajectories are constant between roll-outs time serves as an order-reduction, encoding for virtual trajectories in all dimensions.

For the case of VILTA, the policy is divided in two main parts; one for the stiffness gains and one for the reference values[4]. The additional dimensions of the policy will suggest changes in positional reference in Cartesian space. The advantage of suggesting absolute positions would be that they are easily saturated in a smooth manner to keep the robot from leaving its workspace or running into singularities. The obvious problem is that the initial policy would likely suggest absolute values that are far away from each other, resulting in dangerous accelerations. This problem does not occur when computing velocities, which we can saturate at desired values. This gives the final policy:

$$\pi(\boldsymbol{s}_\pi)_{VILTA} = \begin{bmatrix} \boldsymbol{k}_P & \dot{\boldsymbol{s}}_e^{ref} \end{bmatrix}^T \tag{4-4}$$

For VILTA, the reference becomes part of the action space and thus input t is dropped:

$$\boldsymbol{x}_{VILTA} = \begin{bmatrix} \boldsymbol{s}_e^{'} & \left(\boldsymbol{k}_P,\ \dot{\boldsymbol{s}}_e^{ref}\right) \end{bmatrix}^T \qquad \boldsymbol{y}_{VILTA} = \begin{bmatrix} \boldsymbol{s}_e & \boldsymbol{W}_{ext} \end{bmatrix}^T \tag{4-5}$$

Again the specific dimensionality depends on the problem at hand. Chapter 5-4 and 5-5 detail several scenarios with specific choices for the above expressions applied to Cartesian Impedance Control.

The advantage of defining the problem in such a manner is that it permits VIL and VILTA in joint-space without any further modifications. To make the problem more tractable, several additions are presented next.

4-1-1 Relative Targets

We have seen the mathematical framework for learning hyper-parameters and making long-term predictions in Chapter 3-3-4. The work in [6] presents the concept of relative targets where instead of $\boldsymbol{y}_{abs} := \boldsymbol{s}_{t+1}$, we use $\boldsymbol{y}_{rel} := \boldsymbol{\Delta s} = \boldsymbol{s}_{t+1} - \boldsymbol{s}_t$. In Chapter 3-3 we assumed a prior mean of zero. When using the absolute targets this will cause the predictions to fall back to zero when little data is available. Using instead the state transitions $\boldsymbol{\Delta s}$ as the GP targets implies the assumption that the state will remain unchanged in scarce data spaces. This is a more reasonable assumption of Newtonian mechanics, including dexterous manipulation and interaction. Also, learning state transitions as opposed to absolute values has a positive numerical effect on learning [5].

[3] This observation was confirmed experimentally but is not presented here.

[4] For algorithmic simplicity, we treat the policy as one entity, taking the same inputs.

For this work these benefits will be exploited by defining relative targets for all spatial dimensions in (4-3) and (4-5). Note that this has the following effect when making long-term predictions:

$$\begin{aligned} \boldsymbol{\mu}_t &= \boldsymbol{\mu}_{t-1} + \boldsymbol{\mu}_{\boldsymbol{\Delta s}} \\ \boldsymbol{\Sigma}_t &= \boldsymbol{\Sigma}_{t-1} + \boldsymbol{\Sigma}_{\boldsymbol{\Delta s}} + \mathrm{Cov}[\boldsymbol{s}_{t-1}, \boldsymbol{\Delta s}] + \mathrm{Cov}[\boldsymbol{\Delta s}, \boldsymbol{s}_{t-1}] \end{aligned} \tag{4-6}$$

The most difficult dimension to model is force. When using relative targets for these dimensions, the most significant non-zero value would occur at the contact transition. Thus it is found that a better Log Marginal Likelihood (LML) is found when using absolute targets for the force dimensions.

4-1-2 Informative Prior Means

In addition to relative targets, this work proposes an extension in the form of *informative prior means*. Especially for VIL, we would be foolish to suppose that there is no prior belief to be formulated about the expected motion of a system following a predefined trajectory. Indeed this observation suggests the option to encode the computed reference trajectory as a non-zero prior on the posterior model.
Since the reference is merely a time dependent deterministic constant, we can encode it as a prior and maintain the properties of a zero-mean GP. This is done by subtracting the relevant reference value at each time step from the corresponding target. For absolute targets this gives $\boldsymbol{y}^r_{abs} = \boldsymbol{s} - \boldsymbol{s}^{ref}_e$ and for relative targets: $\boldsymbol{y}^r_{rel} = \Delta\boldsymbol{s} - \Delta\boldsymbol{s}^{ref}_e$. Which again gives a zero-mean GP that is translated by a time-dependent reference. During long-term predictions, the respective reference value is added back onto the translated GP. Now when the model encounters parts of the input space with little or no data, the prediction will fall back to zero as usually. Which is now a prediction of the error between the state and virtual trajectory and hence the prediction will either fall back to the reference or remain at constant distance in the case of absolute and relative targets respectively.
From a system identification point of view, this approach requires the GP to model the deviation of the realized state trajectory and the imposed virtual one. This deviation originates from environmental interaction, unmodelled friction effects and errors in model compensation. For the case of SIC of Equation (2-6) this deviation will thus be much more explicit due to there being no inverse dynamics. This is where one potential application of the algorithm lies, where now the GPs can model the inverse dynamics and adept the stiffness values and virtual trajectories to achieve its goal. There are a few reasons for opting for SIC that were given in Chapter 2-2-1, such as naturally resolving singularities. Chapter 5-4-2 present the results of VIL for SIC.

4-2 Cost Function and Expected Returns with Energy Penalty

This work proposes an expansion to the expected return presented in Chapter 3-4-1 in the form of an energy cost function $\mathcal{A} = \mathbb{E}[c_a(\boldsymbol{a}_t)]$, giving:

$$\begin{aligned} J_{\psi}(\boldsymbol{\tau}) &= \sum_{t=1}^{T} \left[w_1 \cdot \mathbb{E}[c_s(\boldsymbol{s}_t)] + w_2 \cdot \sigma[c_s(\boldsymbol{s}_t)] + w_3 \cdot \mathbb{E}[c_a(\boldsymbol{a}_t)] \right] \\ &= \sum_{t=1}^{T} \left[w_1 \mathcal{E} + w_2 \mathcal{S} + w_3 \mathcal{A} \right] \end{aligned} \tag{4-7}$$

A justification for the choice of cost functions is given next, followed by a derivation of the gradient.

4-2-1 Accuracy Cost Function for Sensorimotor Control

With respect to sensorimotor control, it was experimentally validated by Kording in [65] that a saturating cost function resembles the one used by humans when learning a new skill. They showed the cost to be approximately quadratic for small errors, but significantly less than quadratic for large errors, thereby encoding a robustness towards outliers. Incidentally such a saturating cost function is also proposed by Deisenroth in the original Probabilistic Inference for Learning COntrol (PILCO) algorithm of [5]. There it is noted such a cost has benefits for learning, stating that it has a smoothing effect on policy optimization and carries an intrinsic exploration incentive for the agent when far away from the goal, while preferring exploitation of low-variance trajectories when close to the target. For these reasons we will implement a saturating cost function in the form of inverted squared exponential:

$$c_s(\boldsymbol{s}_t) = 1 - \exp\left(-(\boldsymbol{s}_t - \boldsymbol{s}_{target})^T S^{-1} (\boldsymbol{s}_t - \boldsymbol{s}_{target}) \right) \tag{4-8}$$

Where S^{-1} acts as both a selector and scaling matrix, defining the width of the inverted Gaussian.
Figure 4-2 shows the distribution of an instance far away from the goal (left) and close to it (right). The saturation makes that in the former case, wide state distributions will have a lower total expected cost than a narrow distribution, resulting in exploration of uncertain trajectories. For the latter case however, narrow state distributions yield a lower expected cost when integrating over the whole state distribution and hence will steer the agent into exploiting that certainty when close to the finish [6].

4-2-2 Energy Cost Function for Sensorimotor Control

Optimal control, RL and adaptive control often employ an energy penalty term in their cost functions. A strategy that seems to have a resemblance with biological learning. Diedrichsen shows in [66] that motor schema in the Central Nervous System (CNS) are optimally tuned to the goals of the task by trading off energy consumption with accuracy constraints. Generally, the goal of effective impedance control that is confirmed from human arm studies can be summarized as

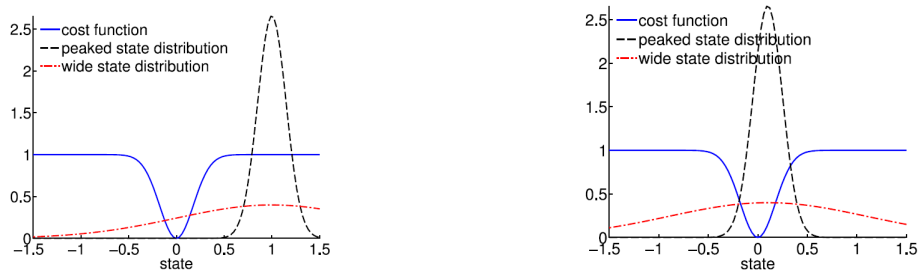


Figure 4-2: *Plots showing the cost and state distribution far away from the goal (left) and close to the goal state (right) [6].*

"Be compliant when possible, stiffen up only when the task desires it." [67]

Such behaviour is highly desirable for robotic systems since it promotes efficient and safe operation when the possibility for spatial reward is small. Two ways present themselves for serving as energy penalty. When active force sensing is performed, a zero-centred state penalty on the external forces can be employed, as is done in this work. This ensures that during constrained motion, when spatial rewards are inhibited by the surroundings, the stiffness gains can be lowered in order to reduce the forces at the end-effector. However, for combined free and constrained motion, such an approach is insufficient, since the forces experienced by the end-effector are by definition zero during this phase. This results in "flat" gradients and thus no incentive to lower the gains. Instead we propose an additional energy term to be added to the Expected return in Equation (3-22) of Chapter 3-4-1. An quadratic cost is assigned proportional to the normalized control actions at each time step, indicated by $\mathcal{A}$ in Equation (4-7).

$$c_a(\boldsymbol{a}_t) = p(\boldsymbol{a}_t)^T S^{-1} p(\boldsymbol{a}_t) \tag{4-9}$$

where $S = \text{diag}\begin{bmatrix} a^2_{1_{max}} & a^2_{2_{max}} & \dots & a^2_{m_{max}} \end{bmatrix} \in \mathbb{R}^{D_u \times D_u}$ is a diagonal matrix used to normalize the actions with respect to their maximum value for that respective dimension. For excluding dimensions from energy penalties the respective index can be set to zero. This is relevant for parts of the policy that suggest reference changes in VILTA (see Chapter 4-1). We can derive the following moment definitions using the rules in Appendix A.

$$\begin{aligned} \mathbb{E}[c_{\boldsymbol{a}}(\boldsymbol{a}_t))] &= \text{Tr}[\boldsymbol{\Sigma}_a S^{-1}] + \boldsymbol{\mu}_a^T S^{-1} \boldsymbol{\mu}_a \\ \text{Var}[c_{\boldsymbol{a}}(\boldsymbol{a}_t))] &= \text{Tr}[2 S^{-1} \boldsymbol{\Sigma}_a S^{-1} \boldsymbol{\Sigma}_a] + 4 \boldsymbol{\mu}_a S^{-1} \boldsymbol{\Sigma}_a S^{-1} \boldsymbol{\mu}_a \end{aligned} \tag{4-10}$$

4-2-3 Gradient of the Expected Return

We now present a concise analytical derivation of the expected return's gradient. The full derivatives related to the accuracy cost are given in [6]. The derivative with respect to the energy penalty is original to this work.

Accuracy Gradient

For the mean of the target cost $\mathcal{E}$ the gradient is derived as follows.

$$\begin{aligned}\frac{\mathrm{d}\mathcal{E}}{\mathrm{d}\boldsymbol{\psi}} &= \frac{\partial \mathcal{E}}{\partial p(\boldsymbol{s}_t)} \frac{\mathrm{d}p(\boldsymbol{s}_t)}{\mathrm{d}\boldsymbol{\psi}} \\ &= \underbrace{\frac{\partial \mathcal{E}}{\partial p(\boldsymbol{s}_t)}}_{(1)} \left[\underbrace{\frac{\partial p(\boldsymbol{s}_t)}{\partial p(\boldsymbol{s}_{t-1})}}_{(2)} \cdot \frac{\mathrm{d}p(\boldsymbol{s}_{t-1})}{\mathrm{d}\boldsymbol{\psi}} + \underbrace{\frac{\partial p(\boldsymbol{s}_t)}{\partial \boldsymbol{\psi}}}_{(3)} \right]\end{aligned} \tag{4-11}$$

From Equation (4-11) it is deduced that the state penalty is affected by the policy parameters through three partial derivatives[5].

$$(1) : \frac{\partial \mathcal{E}}{\partial p(\boldsymbol{s}_t)} = \left\{ \frac{\partial \mathcal{E}}{\partial \boldsymbol{\mu}_t}, \quad \frac{\partial \mathcal{E}}{\partial \boldsymbol{\Sigma}_t} \right\} \qquad (2) : \frac{\partial p(\boldsymbol{s}_t)}{\partial p(\boldsymbol{s}_{t-1})} \qquad (3) : \frac{\partial p(\boldsymbol{s}_t)}{\partial \boldsymbol{\psi}}$$

The first derivative (1) follows from the structure of the cost function, which here is a saturated squared exponential. The partial derivative in (2) is that of the state distribution with respect to the previous state. Its form depends on the manner in which predictions are made and is found by applying the chain rule to all steps of Equation (3-20) in Chapter 3-3-4, with a small addition due to relative targets[6].

$$\frac{\partial p(\boldsymbol{s}_t)}{\partial p(\boldsymbol{s}_{t-1})} = \frac{\partial p(\boldsymbol{s}_t)}{\partial p(\boldsymbol{s}_t')} \cdot \frac{\partial p(\boldsymbol{s}_t')}{\partial p(\boldsymbol{\Delta s})} \cdot \frac{\partial p(\boldsymbol{\Delta s})}{\partial p(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})} \cdot \frac{\partial p(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})}{\partial p(\pi(\boldsymbol{s}_{t-1}))} \cdot \frac{\partial p(\pi(\boldsymbol{s}_{t-1}))}{\partial p(\boldsymbol{s}_{t-1})}$$

Where again each distribution derivative is taken to both mean and variance:

$$\begin{aligned}\frac{\partial p(\boldsymbol{s}_t)}{\partial p(\boldsymbol{s}_{t-1})} &= \left\{ \frac{\partial \boldsymbol{\mu}_t}{\partial p(\boldsymbol{s}_{t-1})}, \quad \frac{\partial \boldsymbol{\Sigma}_t}{\partial p(\boldsymbol{s}_{t-1})} \right\} \\ &= \left\{ \frac{\partial \boldsymbol{\mu}_t}{\partial \boldsymbol{\mu}_{t-1}} + \frac{\partial \boldsymbol{\mu}_t}{\partial \boldsymbol{\Sigma}_{t-1}}, \quad \frac{\partial \boldsymbol{\Sigma}_t}{\partial \boldsymbol{\mu}_{t-1}} + \frac{\partial \boldsymbol{\Sigma}_t}{\partial \boldsymbol{\Sigma}_{t-1}} \right\}\end{aligned} \tag{4-12}$$

Finally, (3) denotes the change in state distribution with respect to a change in parameters. These parameters are mapped through the policy and hence

$$\frac{\partial p(\boldsymbol{s}_t)}{\partial \boldsymbol{\psi}} = \frac{\partial p(\boldsymbol{s}_t)}{\partial p(\tilde{\pi}_{\boldsymbol{\psi}}(\boldsymbol{s}_{t-1}))} \frac{\partial p(\tilde{\pi}_{\boldsymbol{\psi}}(\boldsymbol{s}_{t-1}))}{\partial p(\pi_{\boldsymbol{\psi}}(\boldsymbol{s}_{t-1}))} \frac{\partial p(\pi_{\boldsymbol{\psi}}(\boldsymbol{s}_{t-1}))}{\partial \boldsymbol{\psi}} \tag{4-13}$$

[5] The remaining derivative of the previous state distribution with respect to the policy parameters is retrieved from memory at the previous time step.

[6] For learning relative targets we additionally need to account for the input/output covariance and thus also their derivatives. The translated prior has a derivative of zero.

Exploration Gradient

The gradient of the target cost standard deviation is relatively straightforward. Where the squared root appears due to a transformation from standard deviation to variance:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{S}}{\mathrm{d}\boldsymbol{\psi}} &= \frac{\partial \mathcal{S}}{\partial p(\boldsymbol{s}_t)} \frac{\partial p(\boldsymbol{s}_t)}{\partial \boldsymbol{\psi}} \\ &= \frac{\partial \sqrt{\boldsymbol{\Sigma}_c}}{\partial \boldsymbol{\Sigma}_c} \frac{\partial \boldsymbol{\Sigma}_c}{\partial p(\boldsymbol{s}_t)} \frac{\partial p(\boldsymbol{s}_t)}{\partial \boldsymbol{\psi}} \\ &= \frac{1}{2\sqrt{\boldsymbol{\Sigma}_c}} \frac{\partial \boldsymbol{\Sigma}_c}{\partial p(\boldsymbol{s}_t)} \frac{\partial p(\boldsymbol{s}_t)}{\partial \boldsymbol{\psi}}\end{aligned} \tag{4-14}$$

Energy Gradient

Finally we examine the derivative of the energy penalty with respect to the parameter vector based on the expressions in Equation (4-10)

$$\begin{aligned}\frac{\mathrm{d}\mathcal{A}}{\mathrm{d}\boldsymbol{\psi}} &= \frac{\partial \mathbb{E}[c_a(\pi(\boldsymbol{s}_t))]}{\partial p(\boldsymbol{a}_t)} \frac{\mathrm{d}p(\boldsymbol{a}_t)}{\mathrm{d}\boldsymbol{\psi}} \\ &= \frac{\partial \mathbb{E}[\boldsymbol{a}_t^T S^{-1} \boldsymbol{a}_t]}{\partial \boldsymbol{\mu}_a} \frac{\mathrm{d}\boldsymbol{\mu}_a}{\mathrm{d}\boldsymbol{\psi}} + \frac{\mathrm{d}\mathbb{E}[\boldsymbol{a}_t^T S^{-1} \boldsymbol{a}_t]}{\partial \boldsymbol{\Sigma}_a} \frac{\mathrm{d}\boldsymbol{\Sigma}_a}{\mathrm{d}\boldsymbol{\psi}} \\ &= \quad 2\boldsymbol{\mu}_a^T S^{-1} \frac{\mathrm{d}\boldsymbol{\mu}_a}{\mathrm{d}\boldsymbol{\psi}} \quad + \quad S^{-1} \frac{\mathrm{d}\boldsymbol{\Sigma}_a}{\mathrm{d}\boldsymbol{\psi}} \\ &= 2\boldsymbol{\mu}_a^T S^{-1} \left[\frac{\partial \boldsymbol{\mu}_a}{\partial p(\boldsymbol{s}_t)} \frac{\mathrm{d}p(\boldsymbol{s}_t)}{\mathrm{d}\boldsymbol{\psi}} + \frac{\partial \boldsymbol{\mu}_a}{\partial \boldsymbol{\psi}} \right] + S^{-1} \left[\frac{\partial \boldsymbol{\Sigma}_a}{\partial p(\boldsymbol{s}_t)} \frac{\mathrm{d}p(\boldsymbol{s}_t)}{\mathrm{d}\boldsymbol{\psi}} + \frac{\partial \boldsymbol{\Sigma}_a}{\partial \boldsymbol{\psi}} \right]\end{aligned} \tag{4-15}$$

Where for $\frac{\mathrm{d}p(\boldsymbol{s}_t)}{\mathrm{d}\boldsymbol{\psi}}$ we substitute the bracketed expression from equation (4-11). The gradient $\frac{\partial \boldsymbol{\mu}_a}{\partial \boldsymbol{\psi}}$ denotes a direct change in mean control effort relative to a change in $\boldsymbol{\psi}$, but a change in parameters will also result in a different state distribution, which in turn affects the policy outputs, captured by $\frac{\mathrm{d}\boldsymbol{\mu}_a}{\mathrm{d}p(\boldsymbol{s}_t)}$ multiplied with the partial derivative of the state with respect to $\boldsymbol{\psi}$, which was computed earlier.
Not all output states of GP model are necessarily inputs to the policy and hence these gradients will have non-zero values only at relevant indices. They both depend on the particular controller structure under consideration. For example the non-saturated linear controller in Equation (3-21) results in the gradients

$$\frac{\partial \boldsymbol{\mu}_a}{\partial p(\boldsymbol{s}_t)} = \begin{cases} \frac{\partial \boldsymbol{\mu}_a}{\partial \boldsymbol{\mu}_t} = A \in \mathbb{R}^{D_u \times n} \\ \frac{\partial \boldsymbol{\mu}_a}{\partial \boldsymbol{\Sigma}_t} = \mathbf{0} \in \mathbb{R}^{D_u \times n^2} \end{cases} \qquad \frac{\partial \boldsymbol{\mu}_a}{\partial \boldsymbol{\psi}} = \begin{bmatrix} \boldsymbol{s}_t & I_{m \times 1} \end{bmatrix} \in \mathbb{R}^{D_u \times n_{\boldsymbol{\psi}}}$$

For more complex control policies, including the saturation described in Chapter 4-3-1, the mean of the control input depends on the state variance supplied to the preliminary control policy. Rendering the partial derivative in the expression above to non-zero values.

Gradient of the Expected Return

For determining the gradient of the expected return with respect to the policy parameters, the summation rule is used for taking the derivative of each term separately. The expressions typeset in red indicate the contribution original to this work.

$$\begin{aligned}
\frac{\mathrm{d}J_\pi(\boldsymbol{\psi})}{\mathrm{d}\boldsymbol{\psi}} &= \sum_{t=1}^{T}\left[w_1\frac{\mathrm{d}\mathcal{E}}{\mathrm{d}\boldsymbol{\psi}} + w_2\frac{\mathrm{d}\mathcal{S}}{\mathrm{d}\boldsymbol{\psi}} + w_3\frac{\mathrm{d}\mathcal{A}}{\mathrm{d}\boldsymbol{\psi}}\right] \\
&= \sum_{t=1}^{T}\left[\frac{\partial(w_1\mathcal{E} + w_2\mathcal{S} + w_3\mathcal{A})}{\partial p(\boldsymbol{s}_t)}\cdot\frac{\mathrm{d}p(\boldsymbol{s}_t)}{\mathrm{d}\boldsymbol{\psi}} + w_3\left(2\boldsymbol{\mu}_a^T S^{-1}\frac{\partial\boldsymbol{\mu}_a}{\partial\boldsymbol{\psi}} + S^{-1}\frac{\partial\Sigma_a}{\partial\psi}\right)\right] \\
&= \sum_{t=1}^{T}\left[\left(w_1\frac{\partial\mathcal{E}}{\partial p(\boldsymbol{s}_t)} + w_2\frac{1}{2\sqrt{\boldsymbol{\Sigma}_c}}\frac{\partial\boldsymbol{\Sigma}_c}{p(\boldsymbol{s}_t)} + w_3\left(2\boldsymbol{\mu}_a^T S^{-1}\frac{\partial\boldsymbol{\mu}_a}{\partial p(\boldsymbol{s}_t)} + S^{-1}\frac{\partial\Sigma_a}{\partial p(\boldsymbol{s}_t)}\right)\right)\right. \\
&\qquad \left.\cdot\frac{\mathrm{d}p(\boldsymbol{s}_t)}{\mathrm{d}\boldsymbol{\psi}} + w_3\left(2\boldsymbol{\mu}_a^T S^{-1}\frac{\partial\boldsymbol{\mu}_a}{\partial\boldsymbol{\psi}} + S^{-1}\frac{\partial\Sigma_a}{\partial\psi}\right)\right] \\
&= \sum_{t=1}^{T}\left[\left(w_1\frac{\partial\mathcal{E}}{\partial p(\boldsymbol{s}_t)} + w_2\frac{1}{2\sqrt{\boldsymbol{\Sigma}_c}}\frac{\partial\boldsymbol{\Sigma}_c}{p(\boldsymbol{s}_t)} + w_3\left(2\boldsymbol{\mu}_a^T S^{-1}\frac{\partial\boldsymbol{\mu}_a}{\partial p(\boldsymbol{s}_t)} + S^{-1}\frac{\partial\Sigma_a}{\partial p(\boldsymbol{s}_t)}\right)\right)\right. \\
&\qquad \left.\cdot\left(\frac{\partial p(\boldsymbol{s}_t)}{\partial p(\boldsymbol{s}_{t-1})}\frac{\mathrm{d}p(\boldsymbol{s}_{t-1})}{\mathrm{d}\boldsymbol{\psi}} + \frac{\partial p(\boldsymbol{s}_t)}{\partial\boldsymbol{\psi}}\right) + w_3\left(2\boldsymbol{\mu}_a^T S^{-1}\frac{\partial\boldsymbol{\mu}_a}{\partial\boldsymbol{\psi}} + S^{-1}\frac{\partial\Sigma_a}{\partial\psi}\right)\right]
\end{aligned} \tag{4-16}$$

Where for distributions the derivative is always taken to both the mean and variance as in Equation (4-12). We have thus arrived at an analytical expression for the expected return's gradient which accounts for all the uncertainties in correct Bayesian manner, providing gradients for line-searches as detailed in Appendix A-2.

4-3 Policy Structure

The linear policy was already mentioned in Chapter 3-4. For some learning problems, possibly including VIL, such a simple policy might be enough to successfully learn the task at hand. For more complex scenarios such as VILTA, more expressive representations are likely necessary. A more expressive policy that meets the smoothness and analytical requirements is the deterministic GP, which according to [6] is equivalent to a Radial Basis Function (RBF) network. The following explanation is inspired by that work.

$$\tilde{\pi}_{GP}(\boldsymbol{s}_\pi) = \sum_{i=1}^{N_c} k_\pi(\boldsymbol{s}_i, \boldsymbol{s}_\pi)\boldsymbol{\beta}_i = k_\pi(X_\pi, \boldsymbol{s}_\pi)^T\boldsymbol{\beta}_\pi \in \mathbb{R}^{D_u} \tag{4-17}$$

This expression is very similar to that used for the GP model. Where N_c is the number of basis functions used to express the policy. $X_\pi = [\boldsymbol{s}_1, \dots \boldsymbol{s}_{N_c}]$ are the centers of the Gaussian basis functions, also known as *support points* and $k_\pi(X_\pi, \boldsymbol{s}_\pi)$ is a covariance function similar to Equation (3-5). Just as in Equation (3-14), $\boldsymbol{\beta}_\pi$ is a weight vector containing the entries of

the design matrix $(K_\pi)_{ij}$, the noise variance σ_π^2 and the target vector $\boldsymbol{y}_\pi = \tilde{\pi}(X_\pi) + \epsilon_\pi$, ϵ_π. Contrary to the GP model of the dynamics, here it is assumed that there is no uncertainty regarding the controller itself ($\mathrm{Var}_\pi\left[\pi(\boldsymbol{s})\right] = 0$) making it *degenerate*. This means that the uncertainty in the output of this controller when presented with an uncertain input arises from the cross-covariance between the inputs. Its predictive covariance matrix will thus have zeros on its diagonal. As discussed in the next section, this preliminary policy will be squashed through a saturation function. If this function has a finite input domain before repeating, we can further set the signal variance to 1. This allows to define a Signal-to-Noise Ratio (SNR) for the policy by fixing σ_π^2. For example a value of 0.01^2 would give a SNR of 100. This is essentially a measure for how smooth the policy is.

The training of this RBF is similar to the methodology in Chapter 3-3-3, where in this case the locations of both the support points and the targets are treated as learnable hyper-parameters. Such a data set is sometimes referred to as a pseudo-training set. This leaves the following hyper-parameter vector:

$$\boldsymbol{\psi}_{GP} = \begin{bmatrix} \Lambda & X_\pi & \boldsymbol{y}_\pi \end{bmatrix}^T \in \mathbb{R}^{n_\psi} \tag{4-18}$$

Instead of conditioning on observed data, the gradient of the value function serves as a constraint to guide updates of the hyper-parameters. Generally, for an M-dimensional signal and n_π policy inputs, this leaves $n_\psi = N_c(n_\pi + m) + n_\pi m$ learnable parameters, since the support points can be shared across target dimensions.

4-3-1 Saturation

The second requirement for suitable policies is the possibility for the magnitude of the preliminary input to be saturated between certain extrema without violating any of the smoothness properties needed for the gradient derivation. The requirement for bounded control inputs is in some ways obvious in that all actuators will saturate at certain limits and not respecting those might result in unexpected or dangerous behaviour.
For this thesis an additional requirement is imposed on the stiffness part of the policy, namely *non-negativity*.

It is convenient to define the controller as a concatenation of two parts; the computation of the unbounded control signal, and a subsequent saturation and scaling.

$$\pi(\boldsymbol{s}_t) = \mathbf{sat}(\tilde{\pi}(\boldsymbol{s}_t)) = \left\{ \boldsymbol{a}_t \in \mathbb{R}^{D_u} | \boldsymbol{a}_{min} \geq \boldsymbol{a}_t \geq \boldsymbol{a}_{max} \right\} \tag{4-19}$$

Several options for saturating an arbitrary function are given in Equation (4-20): sigmoid functions, cumulative distribution functions through integration of the probability density function (pdf), or sinusoids.

$$\mathbf{h}_{sat}(\tilde{\pi}) = \frac{1}{1 + e^{-\tilde{\pi}}}, \qquad \mathbf{g}_{sat}(\tilde{\pi}) = \int_{-\infty}^{+\infty} p(\tilde{\pi})\, \mathrm{d}\tilde{\pi}, \qquad \mathbf{f}_{sat}(\tilde{\pi}) = \sin(\tilde{\pi}) \tag{4-20}$$

The sigmoid $\mathbf{h}_{sat}$ is a simple way of saturating any function between $[-1, 1]$ and still be differentiable. However it attains its maximum values at $\pm\infty$, which can lead to numerical instability when extreme values are computed. The cumulative distribution function $\mathbf{g}_{sat}$ will map any continuous random variable between $[0, 1]$. However it suffers the same problem at

its extremes as the sigmoid.
The sinusoid $\mathbf{f}_{sat}$ does not have this problem and is bounded by the interval $[-1, 1]$. Which it attains for the points $\tilde{\pi}(\boldsymbol{s}_t) = \frac{\pi}{2} + k\pi$ for $k \in \mathbb{Z}$. This allows the preliminary policy to merely map its output to an interval of $\pm\pi$, reducing the number of learnable parameters by $2m$ as the signal variance σ_π and noise can be set to constant values. It also allows an analytical derivation of the distribution over controls, given a Gaussian distributed input, although in this case the mathematics is less involved than for $\mathbf{g}_{sat}$ [44].
Presented with an uncertain input the rules in Appendix A give:

$$\mathbf{f}_{sat}(\tilde{\pi}) = p(\pi) \begin{cases} \boldsymbol{\mu}_a = \mathbb{E}[\sin(\tilde{\pi})] = \int \sin(\tilde{\pi}) p(\tilde{\pi}) \, \mathrm{d}\tilde{\pi} = \mathrm{e}^{-\frac{1}{2}\Sigma_{\tilde{a}}} \sin\left(\boldsymbol{\mu}_{\tilde{a}}\right) \\ \Sigma_a = \mathbb{E}[\sin(\tilde{\pi})^2] - \mathbb{E}[\sin(\tilde{\pi})]^2 = \frac{1}{2}\left(1 - \mathrm{e}^{-2\Sigma_{\tilde{a}}} \cos 2\boldsymbol{\mu}_{\tilde{a}}\right) - \boldsymbol{\mu}_{\tilde{a}} \end{cases} \tag{4-21}$$

For distributed inputs the output mean depends on the negative exponential of the input variance. Practically this results in mid-range outputs for parts of high uncertainty.

For this thesis, in the case of a linear policy, the preliminary output is scaled by $\tilde{\pi}_{lin}(\boldsymbol{s}_t) = (\tilde{\pi}_{lin}(\boldsymbol{s}_t)' \cdot (\boldsymbol{a}_{max} - \boldsymbol{a}_{min})) \cdot \frac{3}{2}$, ensuring effective saturation and unique outputs. The saturation function suggested in [6] and employed here is a combination of sinusoids: a 3$^{\text{rd}}$-order Fourier Transform scaled by the upper limit:

$$\pi(\boldsymbol{s}_t) = \boldsymbol{a}_{max} \frac{\left[9 \sin \tilde{\pi}(\boldsymbol{s}_t) + \sin \tilde{\pi}(3\boldsymbol{s}_t)\right]}{8} \tag{4-22}$$

Partial Non-negative Saturation

For the case of learning high-level policies for VIC, additional care has to be taken for saturating the policy. First and foremost, the suggested control actions for all stiffness, damping or inertia gains have to be strictly positive[7] since implementing negative gains will inject energy into the system and cause instability. Thus after saturating and scaling the actions on the interval $[-\boldsymbol{a}_{max},\ \boldsymbol{a}_{max}]$, an additional translation of $\boldsymbol{a}_{max} + \boldsymbol{a}_{min}$ is concatenated after the squashing of the preliminary signal. This maps the final control actions dictating gains on the interval $[\boldsymbol{a}_{min}\ \ \boldsymbol{a}_{max}]$, where $\boldsymbol{a}_{min} > 0$ and $\boldsymbol{a}_{max} > \boldsymbol{a}_{min}$.

$$\pi(\boldsymbol{s}_t) = \boldsymbol{a}_{max} \left(\frac{\left[9 \sin \tilde{\pi}(\boldsymbol{s}_t) + 3 \sin \tilde{\pi}(\boldsymbol{s}_t)\right]}{8} + 1 \right) + \boldsymbol{a}_{min} \tag{4-23}$$

It suffices to say that in contrary to the stiffness dimensions, the reference actions from Equation 4-4 are still allowed to be negative. Although it is still desirable to saturate them to prevent dangerously large accelerations. These are thus exempted from the final translation in Equation (4-23), yielding outputs for dimension i that are bounded by $(-a^{(i)}_{max},\ a^{(i)}_{max})$. Practically these depend on the controller sampling time and the maximum Cartesian velocity:

$$a^i_{max} = v^{(i)}_{max} \Delta t_c \tag{4-24}$$

[7] A gain of zero is often not allowed on most real robots and can cause unexpected behaviour in simulation and are thus avoided.

4-4 Discussion

The steps described in this chapter present a significant computational burden for commercial computers. Two main procedures dominate computation; *model training* and *policy learning*.

The most computationally expensive step in training the model (see Chapter 3-3-3) is the inversion of the covariance matrix K at $\mathcal{O}(N^3)$. However once K is known, it takes only time $\mathcal{O}(N^2)$ to compute the gradient of the LML with respect to a specific hyper-parameter. Hence it is often justified to employ gradient-based optimization from Appendix A-2.
Since the target dimensions are considered mutually independent, for this work parallel threads are designated for training the hyper-parameters of each individual target dimension. This thus results in $\mathcal{O}_{tr}((N^3 + N^2 \cdot n_{\boldsymbol{\theta}}) \cdot n_g)$, where n_g indicates the number of gradient evaluations per optimization iteration and $n_{\boldsymbol{\theta}}$ stands for the number of hyper-parameters, which according to Eq. (3-6): $n_{\boldsymbol{\theta}} = ((N+1)D + N + 2)$ and thus $\mathcal{O}_{tr}\left((N^3(D+2)+2)n_g\right)$. We conclude that the most effective way of minimizing training time is to limit the number of (induced) inputs.

It turns out that training the model is relatively cheap compared to the computational resources required for propagating uncertainties and computing gradients during long-term predictions. Which, due to the inherent sequential nature of these predictions, can not be parallelized. As was found, the predicted mean is simply a linear combination of N basis functions: $\boldsymbol{\mu}_* = \boldsymbol{k}_* \boldsymbol{\beta}$. Hence $\boldsymbol{\beta}$ will not change for a given set of parameters and can thus be stored in memory during training, reducing the number of inversions when computing predictive means.
Looking at Equation (3-18), computing the the entries of $Q \in \mathbb{R}^{N \times N}$ requires a D-dimensional scalar product per entry, where D is number of inputs to the GP. This gives a computational complexity of $\mathcal{O}(DN^2)$. For multivariate targets, the burden scales quadratically with target dimensionality E. This is because Q has to be computed for each entry of the predictive covariance $\boldsymbol{\Sigma}_y \in \mathbb{R}^{E \times E}$, resulting in a total complexity of $\mathcal{O}(E^2 N^2 D)$.
We can thus note that for reducing the total computational burden during policy learning, reducing either the training set or target dimensionality is the most effective, while the dimensionality of the input states can be increased relatively cheaply. However when dealing with data-sets larger than roughly $N = 500$, the complexity starts getting too large. For this reason it will be necessary to use a *sparse* approximation with the help of *inducing inputs*.

Large datasets present a significant problem for making the procedures computationally tractable. Multiple measures can be put into place to reduce the computational complexity. One could for example utilize only a subset of the recorded data during training and prediction. Depending on the sampling frequency one can expect many data points to closely resemble each other. An algorithm for sorting through the dataset, removing similar data points can be useful. Also when making predictions in certain parts of the state space, data far away from there will result in near-zero entries in $\boldsymbol{k}_*$. This suggests one can reduce the dimensionality of the design matrix by only performing regression on nearby data points. Multiple ways exist of deciding which data to use for training and regression. The approach taken in the [6] is through Sparse Pseudo-Inputs Gaussian Process (SPGP), proposed by Snelson in [68] and has an elegant trade-off between under- and over-fitting For an overview of different methodologies and how they relate, the work by Quionero in [69] is an excellent resource. Chapter 5-6 presents an analysis of computational burden for multiple dimensions.

Chapter 5

Experimental Results

The following chapter presents the experimental results of the proposed algorithm for two scenarios; Variable Impedance Learning (VIL), learning variable stiffness gains along predefined motion plans; and Variable Impedance Learning with Trajectory Adaptation (VILTA), learning both stiffness gains and trajectory adaptations simultaneously. Both settings are implemented on a simulated planar robot and experimentally on a *KUKA LBR 7 R800 iiwa*. We start by formulating some general concepts and ideas related to the motion plans and the initial stiffness trajectories. Followed by a description of the experiment design for both the simulated and real system. Results are presented with a concluding discussion on them.

The original Probabilistic Inference for Learning COntrol (PILCO) algorithm developed by Rasmussen and Deisenroth in [5], [24] is implemented in MATLAB and can be found at http://mlg.eng.cam.ac.uk/pilco/release/pilcoV0.9.zip. The files needed for VIL and VILTA can be found in the GitHub repository: http://tinylink.net/PILCOT.

5-1 Trajectory Generation

Both scenarios require some form of predefined trajectories before learning can start. For VIL this entails both motion plans and stiffness trajectories, for VILTA only the latter. The following sections address some points regarding these hand-crafted trajectories.

5-1-1 Kinematic Trajectories

For VIL, the kinematic trajectory is predefined before a run. This motion plan consists of several hand-crafted via-points in the Cartesian workspace. Here these are placed to reflect the part-mating phases detailed in Chapter 2-3. Since the Gaussian Process (GP) model will attempt to model the closed-loop dynamics we want to avoid discontinuities where possible. Hence instead of a trapezoidal velocity profile, a *minimum-jerk* trajectory is preferred. This

type of path is one which minimizes the total jerk throughout the duration of the motion:

$$\int_{t=0}^{T} \dddot{\boldsymbol{x}}_e(t)^T \dddot{\boldsymbol{x}}_e(t)\,\mathrm{d}t$$

This path has the property $\frac{\mathrm{d}^6 \boldsymbol{x}_e}{\mathrm{d}t^6} = 0$ and thus is defined by a 5th-order polynomial $\boldsymbol{x}_e(t) = a_5 t^5 + \cdots + a_0$. The coefficients in this polynomial are defined through boundary conditions on the initial position, velocity and acceleration.

5-1-2 Initial Stiffness Trajectories

To start off the algorithm, an initial roll-out must be performed to supply the model with a preliminary data set. For benchmark problems such as the cart-pole system or the inverted (double) pendulum, the system is actuated with random control inputs drawn from a Gaussian or uniform distribution. For impedance control this approach is not valid. The learned policy will be suggesting high-level stiffness parameters implemented by a lower-level impedance controller. Changing the stiffness value from by two orders of magnitude between two consecutive time-steps will likely be faster than the cut-off frequency of the impedance controller. Thereby not giving an honest estimate of how the system will behave giving such inputs. Also, oscillations due to fast changing impedance targets with a stiff environment are likely to occur. Finally, commercial robots often do not allow the stiffness to change dramatically from one time step to the next for these same reasons. We would thus like the initial data-set to cover a large area of the state-action space, while still being safe to implement. To this end, initial stiffness gains are generated that follow an *Ornstein-Uhlenbeck*[1] trajectory shown in Figure 5-1. This is a stationary mean-reverting Gauss–Markov process. Since it is autoregressive, its output will depend on previous values, inhibiting drastic changes.

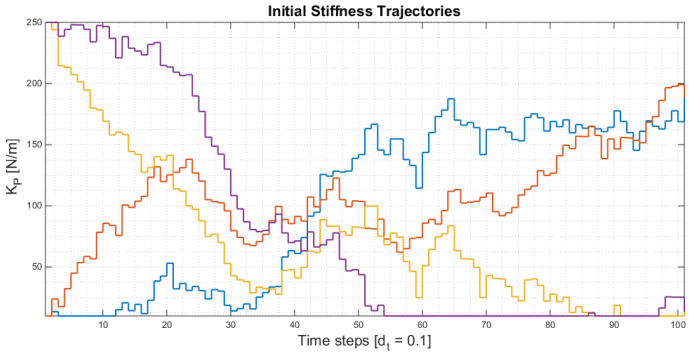


Figure 5-1: *Four initial stiffness trajectories following an Ornstein-Uhlenbeck process.*

5-2 Simulated System Setup

Simulations are performed in a Simulink environment. *The Robotics Toolbox* by Corke is used to model the robot and compute dynamics [70], [71]. The controller contains the impedance

[1] Used to describe the velocity of massive Brownian particle under the influence of friction.

law detailed in Chapter 2-2-2 and 2-2-1 acting on a frequency of $f_c = 200$ [Hz]. Resolved acceleration in joint space is achieved through a recursive Newton-Euler algorithm [3]. For increased speed, MATLAB executable (.mex) files are generated beforehand that implement the forward and backwards dynamics in C++.
Simulations are performed on a planar 3-Degrees of Freedom (DOF) manipulator as depicted in Figure 5-2. The robot's base is located at $\boldsymbol{x}_{base} = \mathbf{0}$. All three links share the same length

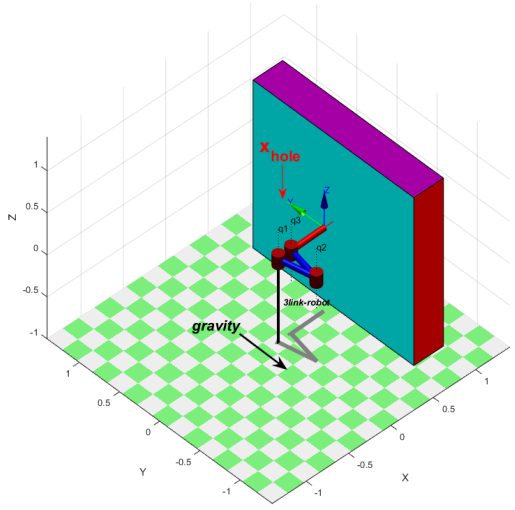


Figure 5-2: *Three dimensional overview of the simulated robot in the environment.*

$l_i = 0.40$ [m] and mass $m_i = 1$ [kg]. This gives it a total workspace of a 1.2 [m] circle around the base[2]. Each joint has a motor with inertia $I_m = 1e^{-4}$ [Nm/rad] and small viscous friction. For forward simulation of the system's dynamics, the inertial parameters are perturbed by 15% with respect to the model used for kinematics and feedback. This offers a possibility for the learning agent to compensate for a mismatch in inverse dynamics through GP modeling and ultimately its impedance gains.
The robot is assumed to be endowed with joint position encoders and tachometers. Each sensor suffers under independent and identically distributed (i.d.d.) Gaussian noise with a variance of $\sigma^2_{q^i} = 0.1$ [deg] for the encoders and tachometers and $\sigma^2_{q^i} = 0.1$ [N] for the Force/Torque-sensor. The joint positions at each time step are used for computing the Cartesian state vector according to the formalism introduced in Chapter 2-1. The overall block-diagonal noise matrix in SI units is:

5-2-1 Contact Simulation

The environment consists of a solid wall at $\boldsymbol{x}_{wall,x} = 0.65$ [m] relative to the base. It is modelled as a highly stiff spring with low damping. The force exerted by the wall is thus a function of the displacement of the TCP with respect to the wall's initial position.

$$\boldsymbol{W}_{ext} = K_{env}(\boldsymbol{x}_e - \boldsymbol{x}_{wall}) + d(\dot{\boldsymbol{x}}_e)$$

[2]The actual reachable workspace is smaller since singularities will occur near the boundaries which have to be avoided.

Where $d(\dot{\boldsymbol{x}}_e)$ is some directional linear damping term[3] and $K_{env} = I_6 \cdot 10^5$.
The additional insertion resistance eluded to earlier is modelled as linear spring with $k_{insert} = 300$ [N/m] that catches just after the centre of the hole and stops 2 [cm] before the end. Such resistances are often encountered in part mating procedures for clicking or forcing the part in place. Here it serves as an increased complexity for the desired compliance. At the end of the hole, another constraint is placed which has the same stiffness as the other surfaces. A small viscous friction effect with $C_{f,visc} = 0.5$ is simulated whenever the robot is in contact with a surface.

5-2-2 Geometric Problem Description

The goal of the agent is to insert the peg as far as possible into the hole. Geometrically this means it will try to minimize the distance between via-point 3 in Figure 5-3 and its Tool Center Point (TCP). The centre of the hole (2) is located at $\boldsymbol{x}_{hole} = [0.7\ 0.2\ 0]$. It has a radius $r_{hole} = 0.005$ and a depth of $l_{hole} = 0.1$. Since the TCP of the end-effector is simulated as having no volume, the radius of the hole is also the tolerance on either side.
One trial lasts for $T = 10$ [s]. The diagram in Figure 5-3 shows a geometric overview of

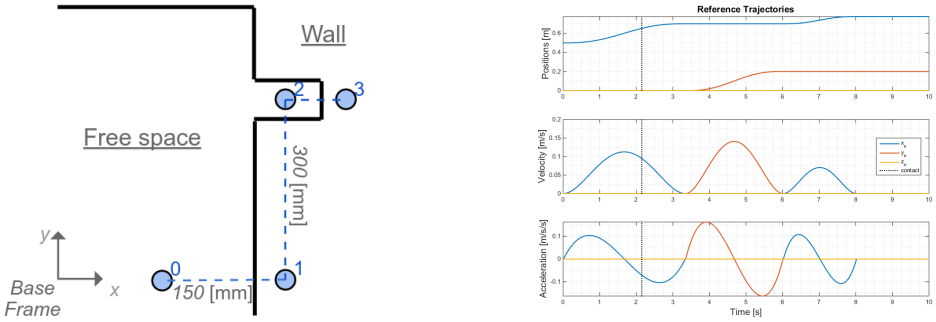


Figure 5-3: *Schematic overview of the stiffness learning task in (left) with via-points indicated by blue solid circles. Minimum-jerk trajectory profile (right) with moment of contact indicated with black dotted line.*

the experiment with the relative via-points. Firstly the manipulator must move towards the vicinity of the target, this is the *approach*. For this the TCP moves from via-point (0) at $\boldsymbol{x}_0 = [0.5\ 0\ 0]$ to (1) at 50 [mm] behind the wall, ensuring contact with the environment. for *alignment* it then moves to point (2) located at the centre of the hole $\boldsymbol{x}_{hole}$[4]. For the *insertion* phase, point (3) is placed just behind the end of the hole at $\boldsymbol{x}_3 = \boldsymbol{x}_{hole} + l_{hole}/2$. The resulting trajectory is shown Figure 5-3 (right).

5-2-3 Coupled Closed-loop Damping

As shown in Chapter 2, impedance control simplifies controller design to the selection of impedance parameters of the closed-loop system:

$$M_d\ddot{\tilde{\boldsymbol{x}}}_e + B_d\dot{\tilde{\boldsymbol{x}}}_e + K_d\tilde{\boldsymbol{x}}_e = \boldsymbol{W}_{ext} \tag{5-1}$$

[3]Since the end-effector is not rigidly connected to the environment, damping only occurs when moving in the direction of the wall.

[4]Since this simulation does not accurately model jamming and friction and the location of the hole is known precisely, *alignment* occurs relatively easy.

The selection of parameters define the frequency response variables as

$$\omega_n = \sqrt{\frac{K_d}{M_d}} \qquad \zeta = \frac{B_d}{2\sqrt{K_d M_d}} \tag{5-2}$$

For this thesis only the stiffness gains are learned. The inertial matrix is set to a predetermined value and the damping term is tuned to obtain a nicely damped system. Defining a damping ratio $\zeta = 0.7$, allows computing the desired damping matrix at each time step as $B_d = 2\zeta\omega_n M_d$. When the robot comes in contact with the environment, its dynamics will become coupled. This in turn effects the natural frequency of the coupled arm-wall system:

$$\omega_{n_{coupled}} = \sqrt{\frac{K_d + K_e}{M_d}}$$

The damping ratio is inversely proportional to this frequency and thus for stiff environments ($K_e \gg K_p$), we expect to see decreased damping and stronger oscillations during the transient between free and constrained motion. For this thesis we will assume the environment characteristics to be known and adjust the desired damping accordingly to $B_d = 2\zeta\sqrt{\frac{K_d+K_e}{M_d}}$, which achieves a nice stable transient. Schemes for on-line estimation of wall stiffness have been shown to improve performance and the interested reader is referred to [72].

5-3 KUKA iiwa Experimental Setup

Chapter 2 gave an introduction to the control theory behind ideal impedance behaviour. In practice this has been hard to accomplish. One of the reasons for this is the non-linearity of the torque-current relationship in even the best electric motors. This means that for commercial robots, impedance control is often realized by cascaded inner positional controllers. The newest wave of professional manipulators are light-weight, flexible-joint, torque controlled systems. The *KUKA LBR 7 iiwa R800* depicted on the left in Figure 5-4 is such a system. The block-diagram in Figure 5-4 gives a schematic overview of the communication protocols during the experiments. The Robotic Operating System (ROS) metapackage '*iiwa stack*' (https://github.com/SalvoVirga/iiwa_stack [73]) is used to establish a UDP communication with the KUKA operating system '*sunriseOS*' running a local ROS node in its Java environment. The 'MATLAB Robotics Toolbox' [74] is used to create a node within the MATLAB environment for easy integration with the learning algorithm.

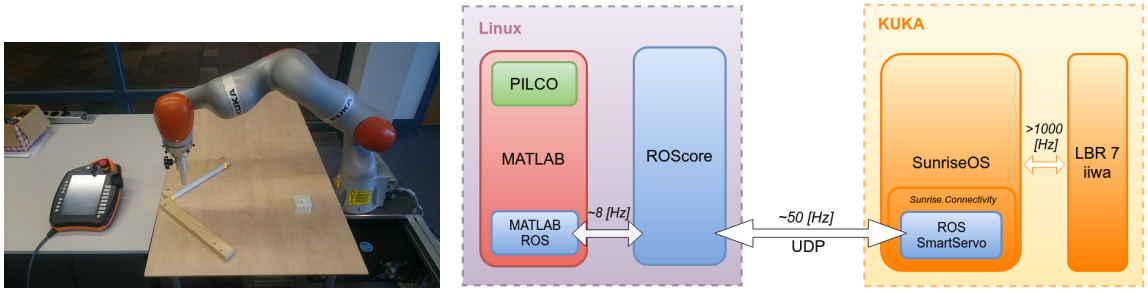


Figure 5-4: *Photograph of the KUKA iiwa experimantal setup (left) with a schematic overview of the communication protocol (right) in place to control the KUKA iiwa from MATLAB through ROS.*

Figure 5-5 shows a geometric overview of the setup with trajectory via-points. The goal for the agent is to minimize the distance between the TCP and point (2). Two walls placed at 45 degree angles constrain the motion.

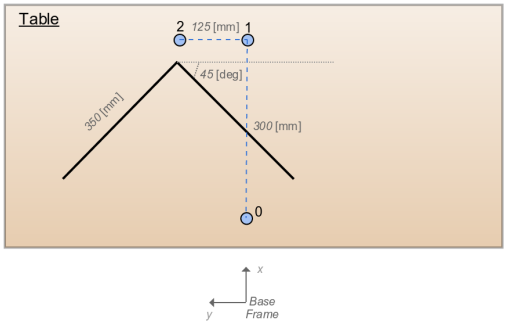


Figure 5-5: *Geometric overview of the experiment design with blue circles indicating via-points.*

5-4 Variable Impedance Learning

Below are presented the results for the stiffness learning scenario with the relevant concepts as defined above. Table 5-1 to 5-4 summarize the parameters chosen for each of the relevant parts of the learning algorithm. For the non-linear Radial Basis Function (RBF) controller, multiple numbers of basis function are implemented to asses their effectiveness compared to a linear controller.

GP Model	
Δ_t	0.1 [s]
$\boldsymbol{x}_{in}$	$x\ \ y\ \ \dot{x}\ \ \dot{y}\ \ t$
$\boldsymbol{y}_{target}$	$\Delta x^r\ \ \Delta y^r\ \ \Delta \dot{x}^r\ \ \Delta \dot{y}^r\ \ f_x\ \ f_y$
N_{ii}	300

Table 5-1: *Relevant parameters for the GP model in planar VIL. Differences are indicated by Δ, reference priors by an r superscript.*

Policy		
	Linear	RBF
N_c	-	10 25
$\boldsymbol{a}$	$K_{p_x}\ \ K_{p_y}$	
$\boldsymbol{s}_\pi$	$x\ \ y\ \ \dot{x}\ \ \dot{y}$	
Limits	(5, 250)	
n_ψ	10	68 208

Table 5-2: *Parameters relevant to the high level stiffness controller.*

Value function		
	$c_s(\boldsymbol{s}_c)$	$c_a(\boldsymbol{a}_c)$
type	sat.	quad.
input	$x\ \ y\ \ f_x\ \ f_y$	$k_{p_x}\ \ k_{p_y}$
target	0.75 0.2 0 0	0 0
width	0.05 0.05 20	-
weight	$w_2 = -0.25$	$w_3 = 0.01$

Table 5-3: *Relevant parameters for the GP model. w_i are relative to $w_1 = 1$.*

Optimization	
method	(L)BFGS
N	25
K	1
Length	50
MFEPLS	25
MSR	100

Table 5-4: *Parameters relevant to the policy optimization algorithms.*

With respect to Table 5-4: N is the number of roll-outs, K is the number of initial states for which the policy is optimized, Length the number of line-searches per iteration, MFEPLS the number of function evaluations per line-search and MSR is the maximum slope ratio.

5-4-1 General Impedance Simulation Results

We start by examining the results for General Impedance Control (GIC) detailed in Section 2-2-2. The plots in Figure 5-6 give a comparative overview of the final policies for both the linear policy and increasingly expressive non-linear ones. It also shows the predicted expected returns superimposed with the recorded values over 25 iterations. For this and all episodic cost plots, green circles indicate all trials during that episode were successful at inserting the peg fully into the hole. Yellow diamonds indicate some roll-outs were successful and red squares mean that none were. All distributions show 95% confidence intervals.

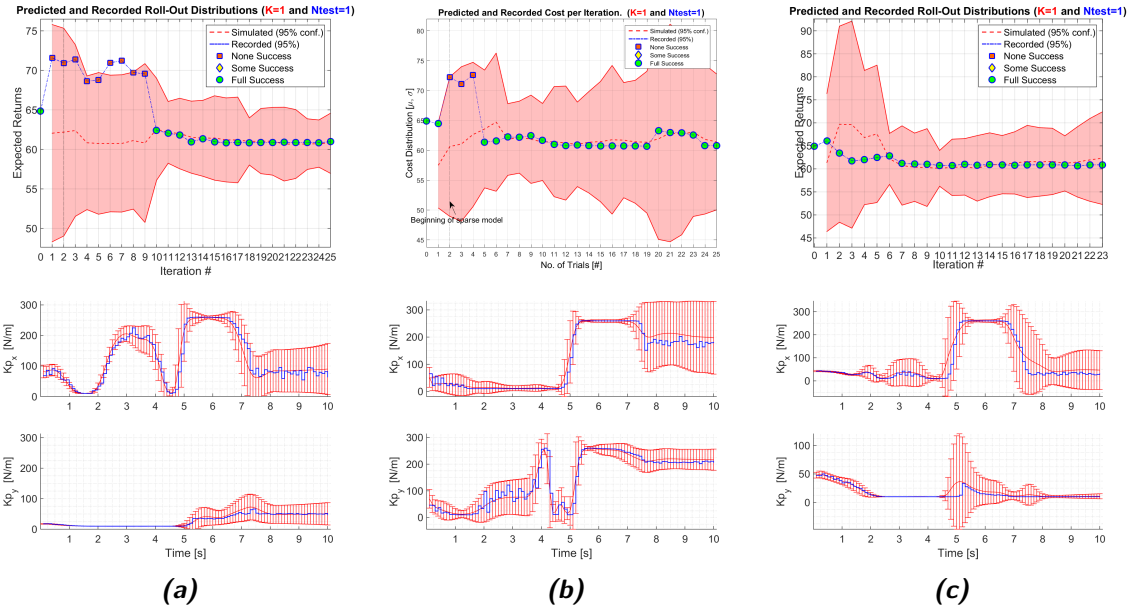


Figure 5-6: *Expected returns and final policies after 25 iterations for three policy structures; linear* ***(a)*** *and RBF with* $N_c = 10$ ***(b)*** *and* $N_c = 25$ ***(c)***.

For all types, the expected return converges to roughly 61. For the linear policy this takes almost 15 trials as the uncertainty in the model is reduced. For RBF policies the overall strategy is found before a mere 10 episodes. The linear policy fails for 9 trials before succeeding, despite the initial roll-out resulting in success. This is likely due to the initial energy penalty taking up too much of the reward. The policy thereby adjusts its (limited) expressiveness into lowering the gains too much and thus failing. Only after exploratory actions does it manage to converge to successful insertion. This behaviour is seen in 5-6b at episode 5 and is no longer present when $N_c = 25$.
Despite the similarity in values, each policy has found another local optimum for its expected return.
What is similar for every policy are low stiffness gains perpendicular to the wall both before and after insertion. Stiffness is only increased when the peg is located in front of the hole, to

increase insertion speed and overcome the resistance encountered at $t \approx 6.5$. The RBF controller with 10 support points shows a short increase in lateral stiffness while the peg moves in the y-direction, followed by a short drop in stiffness to reduce the momentary lateral forces encountered from the side of the hole's wall. This behaviour is not shared in the particular run shown here for $N_c = 25$, but was informally observed as a recurring strategy.

5-4-2 Simple Impedance Simulation Results

An interesting application is in Simple Impedance Control (SIC), detailed in Chapter 2-2-1. Naturally such control will give somewhat degraded performance over the fully compensated case, but several advantages over GIC where given in Chapter 2-2-1, such as naturally dealing with singularities. Instead, compensating for model mismatch only at places where it is beneficial for the agent by increasing its stiffness gains can be of much value in such scenarios. The maximum stiffness values are increased to $k_P^{x,y} = 500$ to allow for this. The results are discussed below. Looking at Figure 5-7 we see that the model and data trials convergences to

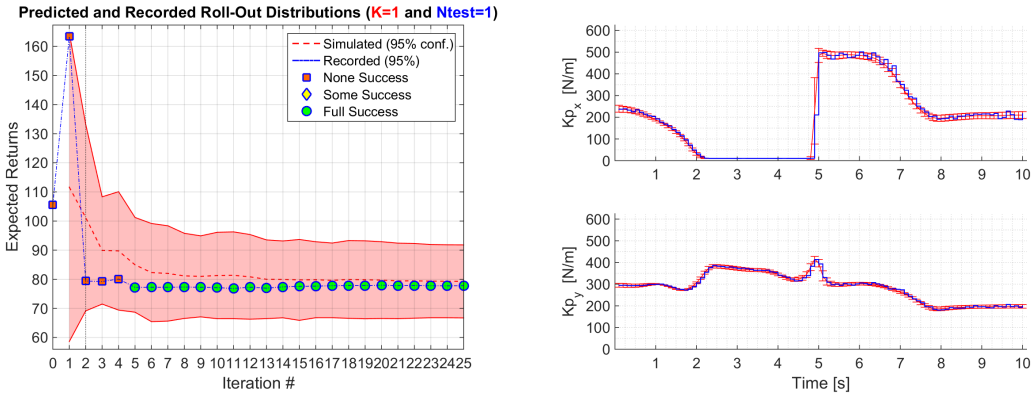


Figure 5-7: *Results of stiffness learning for SIC. Predicted and recorded returns (left) converge to a value of 80. Sparse approximation starts at $k = 4$. The final policy is shown in (right).*

a return of 78. The increased cost with respect to GIC being a result of the larger interaction forces arising from uncancelled inertial and Coriolis terms. The difference in cost over the course of a single run is now more drastic than before, since less of the reward is 'taken up' by the initial low-level controller, leaving more for the policy.
The final policy has some interesting characteristics. For the perpendicular x-direction, wee see initial high stiffness, dropping to a minimum just before contact to minimize the external forces as much as possible. When insertion starts at $t \approx 5$, stiffness is quickly increased to accommodate faster movement. For $k_P^{(y)}$, the stiffness remains higher over the course of the task, since lower stiffness values (the case in early iterations) are too small to succeed at insertion in the absence of dynamic compensation. The plots in Figure 5-8 show the immediate predicted and recorded cost after one episode and at the end of learning. Initially the model misjudges the expected return because the early policies are unsuccessful at inserting the peg. At the end of learning the predicted cost has matched the real data.

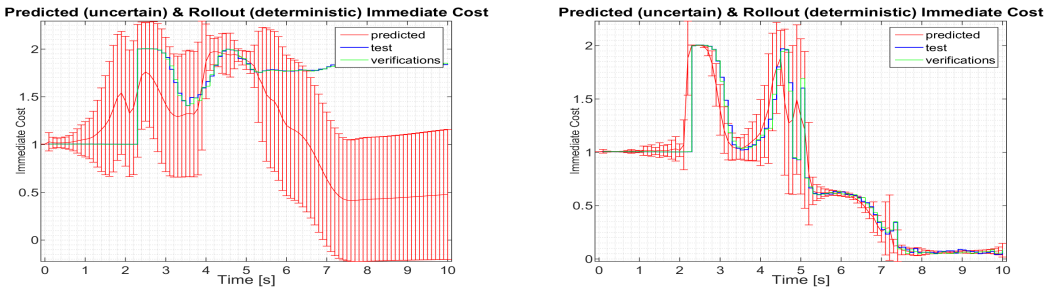


Figure 5-8: *Predicted and recorded immediate cost distributions of VIL after episode 1 (left) and after convergence (right).*

5-4-3 Experimental Results on KUKA LBR iiwa

Finally the stiffness learning algorithm was implemented on a physical KUKA iiwa robot. Due to real-time constraints in the communication chain the loop-time is increased to $\Delta_t = 0.15$ [s]. For these experiments the inputs to the GPs are set to $\boldsymbol{x}_{in} = [x \ y]$. The KUKA iiwa does not allow the definition of specific damping constants, but instead takes a damping ratio ζ for each Cartesian direction. For these experiments this was set to $\zeta = 0.7$.

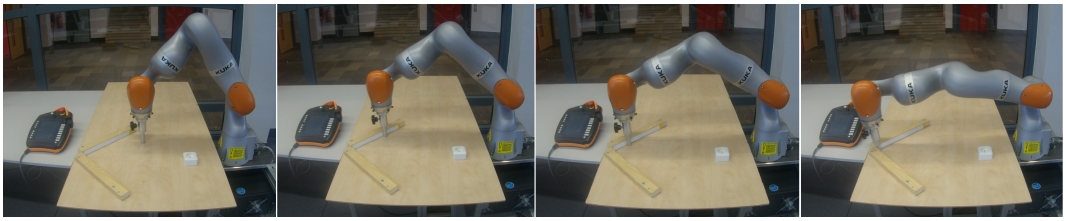


Figure 5-9: *Progression photographs of VIC experiments on the KUKA LBR iiwa.*

The left of Figure 5-10 indicates that the expected return quickly drops from an initial 44 to 34 after 6 interactions and 4 policy updates. Afterwards the policy varies around this (sub)-optimal solution. The stiffness progressions ion the right again show an initial high stiffness in x, up until the moment of contact when it is lowered to accommodate the wall before being increased to help in penetration. The stiffness in y is maximal during free motion until $t \approx 3$ when it drops sharply to already allow for movement towards the target.

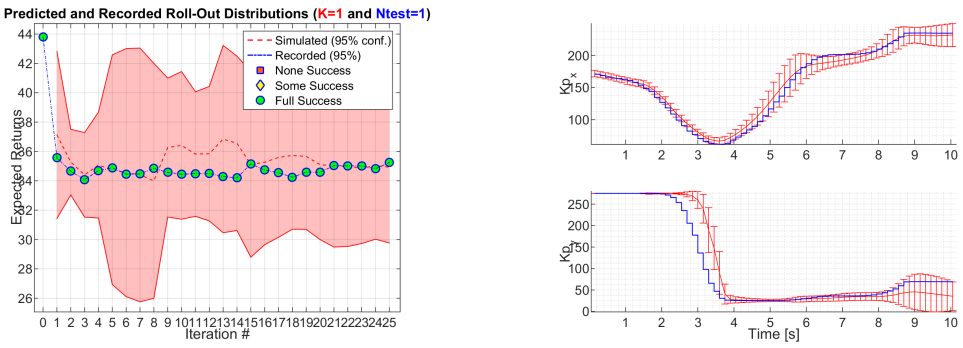


Figure 5-10: *Expected returns (left) and final policy (right) after 25 episodes of VIL on the "KUKA LBR 7" $N_c = 10$.*

The learned model for VIL on the KUKA LBR is shown in Figure 5-11. It is clear that the GPs are limited in predicting the velocities, resulting in increasing uncertainty from $t \approx 4$, possibly due to encountered static friction throwing the model off. Chapter 5-6 gives some further thoughts on this uncertainties. The cost dimensions of x, y and $f_{x,y}$ show better fits, although the final spike in f_x is not modelled properly. The predictive variance does indicate an uncertainty about those final dynamics. Judging from the low variance policy outputs on the right in Figure 5-10, the uncertain velocities are not important for computing stiffness gains.

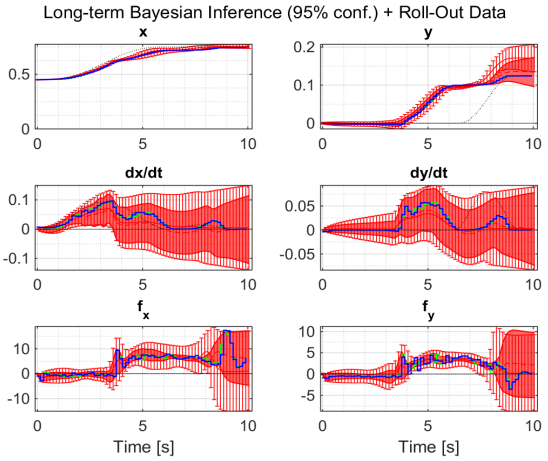


Figure 5-11: *Long-term predictions and data trials for KUKA iiwa VIL, $N_c = 10$. Shaded areas are the full models, red errorbars the sparse GPs used for policy improvements.*

5-5 Variable Impedance Learning with Trajectory Adaptation

VILTA entails learning both stiffness gains and reference trajectories simultaneously. The additional problem that the agent is presented with is that the location of the hole is unknown before learning. The only thing known is that the end of the hole is located somewhere on the interval $\boldsymbol{x}_y \in (0.1\ 0.3)$ and $\boldsymbol{x}_x \in (0.5\ 0.8)$. Analogous to the search phase described in Chapter 2-3, the initial hand-crafted references are thus shaped in a manner that scans over the interval in x and y. The following tables summarize the parameters relevant to the GPs and policy.

Value function		
	$c_s(\boldsymbol{s}_c)$	$c_a(\boldsymbol{a}_c)$
type	sat.	quad.
input	$x\ \ f_x\ \ f_x$	$k_P^{(x)}\ \ k_P^{(y)}$
target	$0.8\ \ 0\ \ 0$	$0\ \ 0$
width	$0.05\ \ 0.05\ \ 20$	-
weight	$w_2 = -0.25$	$w_3 = 0.01$

Table 5-5: *Relevant parameters for the GP model in planar VILTA. Weights relative to $w_1 = 1$.*

Policy	
Type	RBF
N_c	50
$\boldsymbol{a}$	$k_{p_x}\ \ k_{p_y}\ \ \Delta x_{ref}\ \ \Delta y_{ref}$
$\boldsymbol{s}_\pi$	$x\ \ y\ \ \dot{x}\ \ \dot{y}$
Limits	$(5,\ \ 250)\ \ (-0.0025\ \ 0.0025)$
n_ψ	416

Table 5-6: *Parameters relevant to VILTA policy.*

Notice that time has been dropped as an input since the followed trajectory is no longer predetermined and the policy now conditions the model on similar encountered states for different references.
Also, since the exact location of the hole is unknown to the agent, the inputs to the state-dependent cost function can clearly not remain the same. Instead y is dropped as input to $c_{\boldsymbol{s}}(\boldsymbol{s}_t)$ and instead only the displacement perpendicular to the wall serves as input to the cost function[5]. This cost function basically tells the agent to maximize the penetration depth into the wall, while keeping contact forces to a minimum. We do not want to kill the, already small, initial motivation for suggesting non-zero reference changes, especially since there is no direct reward to be reaped from moving in the y direction.

5-5-1 Simulation Results

Although some runs resulted in more elegant outcomes than the ones here, these were chosen for a more interesting and honest discussion on the strengths and weaknesses of the algorithm.

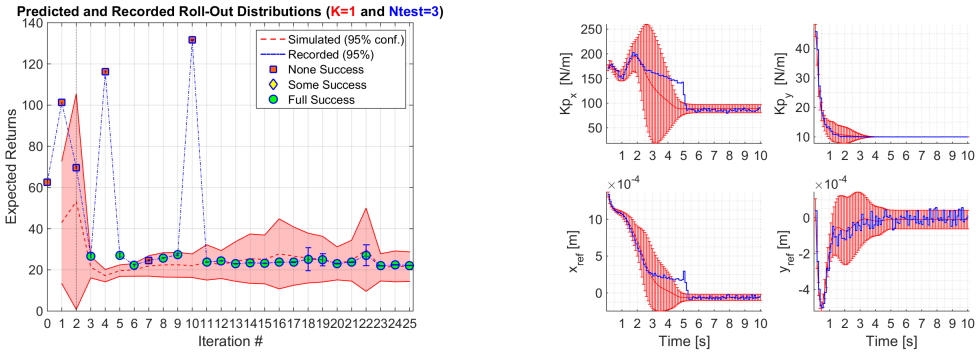


Figure 5-12: *Episodic cost (left) and final policy (right) of simulated VILTA for* $N_c = 50$.

The left graph in Figure 5-12 shows the episodic cost for VILTA for $N_c = 50$. The initial '*searching*' trial results in a return just over 60. Roughly equal to the converged returns during VIL of Section 5-4-1. During the third planning phase the policy finds a spatial trajectory capable of reaching the end of the hole, drastically decreasing the recorded and expected returns. A slight modification during iteration 4 again misses the hole. After 11 policy updates the policy converges to a solution that consistently completes insertion at a cost of 20.
Figure 5-12 reveals that it does this by immediately moving in the $+x/y$-direction towards the opening in the wall while maintaining high stiffness values. When the task no longer requires movement, the stiffness gains are lowered to values of 60 and 10 respectively.
The long-term predictions depicted in Figure 5-13 show accurate predictions for x and y, with increased uncertainty around the discontinuity. The velocities and forces are again hard to capture by the sparse model. The aggressive movement likely increases the effect of the contact discontinuity. The local solution does not properly minimize the forces encountered inside the hole by continually suggesting non-zero reference changes even after the hole's lateral position has been reached. See Section 5-6 for a discussion on these phenomena.

[5]The position and force both have their own saturated cost function with appropriately sized widths, which are later added together in a Bayesian manner.

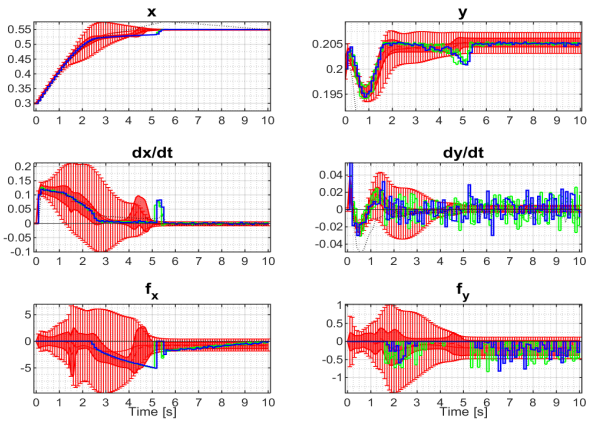


Figure 5-13: *Long-term predictions and recorded data for simulated VILTA with $N_c = 50$. Shaded area are full GPs for comparison.*

Generalizing to Variable Start States

Another interesting generalization is that of *variable start-state VILTA*. For many practical applications, even when the start state can be accurately determined, it is useful to learn trajectory policies that can generalize to a range of initial conditions. To this end, the algorithm is adjusted to allow policy optimization for start states varying over an interval of 20 [cm]. The goal still remains constant during learning, although unknown to the agent before learning[6]. In the current scenario the policy essentially learns where the hole is located and generalizes in such a way to adjust its trajectories in order to reach that point. Each iteration now consists of 4 physical roll-outs, the start state of each being uniformly drawn on the interval $x_y \in (0.1,\ 0.3)$. One of these is used for supplying the GPs with new data. Each line-search direction of every policy improvement is now determined by making $K = 3$ long-term predictions and averaging the expected return and gradient directions over them.

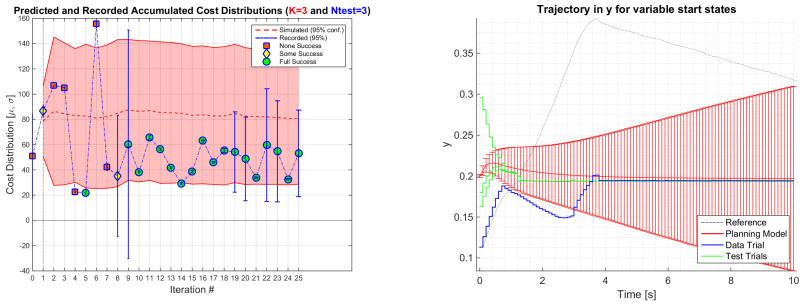


Figure 5-14: *Learned reference trajectories for y direction for variable start states, $N_c = 75$. Despite a relatively high variance fit, the policy correctly steers the peg into the hole for a range of starting conditions.*

Figure 5-14 shows the cost and y dimension model. We can see that the RBF policy has

[6]Another interesting scenario lies in variable goal states, where the policy learns to generalize insertion even when the hole is located at different locations each iteration. Force-conditioned polices would be necessary (see the discussions in Chapter 5-6 and 6-2).

learned to find the hole after 10 iterations for initial conditions on the interval $y = [0.1\ 0.3]$. However we can also see that for this run, the sparse model is very crude and does not converge to the recorded data, likely due to an increased range of different contact transitions.

5-5-2 Experimental Results on KUKA LBR iiwa

The final scenario presented is VILTA implemented on the physical KUKA iiwa robot.

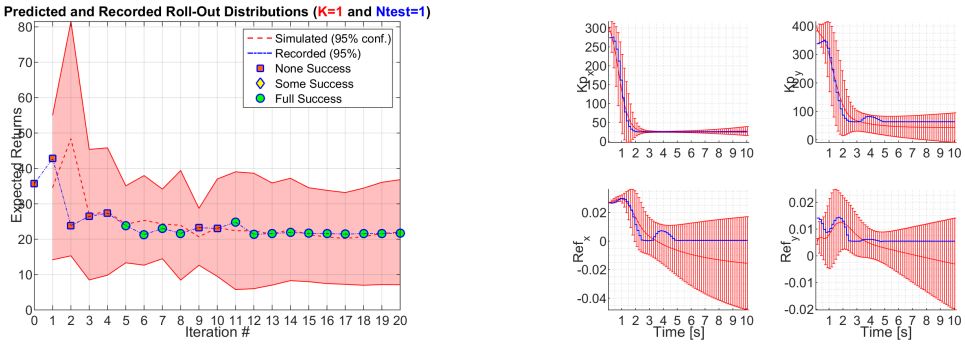


Figure 5-15: *Episodic cost and final policy for VILTA on KUKA LBR 7 iiwa, $N_c = 25$.*

The left plot in Figure 5-15 shows the cost reaching 22 from an initial of 36 after 6 roll-outs, a mere 80 seconds of interaction. The right graph shows that the policy again settles on immediately steering the end-effector towards the goal with high gains. When the goal is reached the gains are lowered and the reference settles to zero. The variance in the predicted actions stems from the cross-covariances of the policy's input states. The higher variance for the reference thus suggests they rely more heavily on their velocity inputs through decreased length-scales on those dimensions.

5-6 Discussion

What follows is brief discussion on the obtained results and an analysis on computational complexity and scalability. A summary, conclusions and possible future directions are given in Chapter 6.

Overall we may state that the proposed algorithm is capable of learning several variants of simple Peg-in-Hole (PiH) tasks. The average convergence speeds for all scenarios is promising with most runs converging to a local optimum within 100 seconds of physical interaction (10 trials). However the sparse model fit and quality of the obtained solutions are not always optimal.

For VIL detailed in Section 5-4 it was found that several policy structures, including a linear architecture, was able to converge to a (local) optimum. The RBF policies were capable of some finer adjustments after a general stiffness trajectory was found. Interestingly, the (sparse) GP model in the case of SIC was observed to often behave better than in the case of GIC. One reason for this could be the remaining presence of open-loop dynamics in the recorded data. These dynamics might be more suitable of being modelled by the chosen

squared exponential kernel. Whereas for GIC, the GPs are expected to only model the way in which the recorded data deviates from the imposed trajectory, after dynamical model compensation by the control law. This might force the hyper-parameters to try and explain the thing they are not good at fitting, the contact transition. While for SIC, the GPs capture the more general dynamics and use these for the sparse posterior. Also, GIC has the (normally beneficial) effect of a very short transient between the moment of contact and a constant desired force. SIC however shows more gradual transient forces, although levelling out at larger values. This results in higher cost, but is beneficial for the underlying GPs.

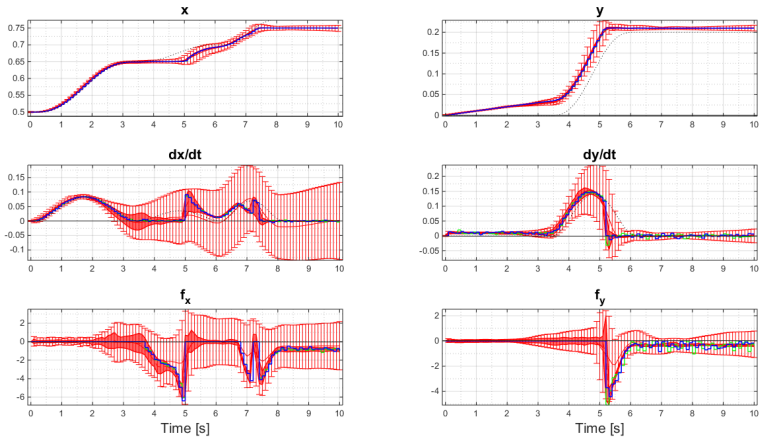


Figure 5-16: *Comparison between the full GP models and their sparse approximation for GIC VIL after 20 episodes using 300 inducing inputs.*

Looking at Figure 5-16, it is seen that the full model predicts the real-life data better and with higher certainty than the sparse approximation. This discrepancy is likely a result of the difficulty induced input models have with incorporating data points far away from the existing set [68].

For the case of VILTA the learning algorithm is again capable of finding a (locally) optimal solution. The final policies exhibit behaviour that aligns with a common sense intuition: high gain feedback during free motion towards the hole followed by large compliance in the vicinity of the hole and zero reference velocities after insertion. Convergence guarantees for VILTA are weaker than for VIL. It is not uncommon for the model to diverge into high variance estimates halfway into the learning process. It is hypothesized that the discrete nature of possible outcomes during simulation has a negative effect on model convergence. For the rather simplistic contact simulation, there are a number of discrete 'choice points' for the model. For example the references might suggest a trajectory that moves towards the wall and simultaneously laterally to the hole. In the ideal case the peg arrives at the y location of the hole at exactly the moment the peg would otherwise touch the wall. If in a subsequent trial, the robot's lateral velocity is just a fraction higher, it will miss the hole entirely and obtain widely different output data for what it perceives are virtually the same inputs. It is reasoned that because of this the hyper-parameters start to drift in non-intuitive directions, trying to account for multiple things at once, thereby loosing the ability to accurately predict the more common dynamics.

Lastly the absence of force as policy input needs to be addressed. Defining force-conditioned policies could improve the generalization power of the algorithm by for example lowering its stiffness in the case of unplanned obstructions or dealing with variable goal states. It was empirically observed that presenting forces as inputs to the policy could result in instability by venturing into dangerous parts of the action space. This happened if the policy rightfully decided to sharply lower the stiffness values at the moment of contact. This sudden change would result in the end-effector loosing contact, which in turn yielded maximum stiffness values from the policy. This behaviour would then spiral into a jittering motion, oscillations and instability, especially in the case of simulated VILTA. Intelligent reward shaping could help in steering the policy away from these parts of the action space.

5-6-1 Computational Complexity Analysis

Probabilistic inference is notoriously heavy in terms of computational complexity. The experiments presented here were for mere planar scenarios. The plots below show the average computation times for different parts of the algorithm during the fifth iteration without any parallelization. The data was recorded on a '*Windows 7 x-64*' machine sporting an '*Intel Xeon E5*' CPU with 8 cores[7] running at 3.50 [GHz].

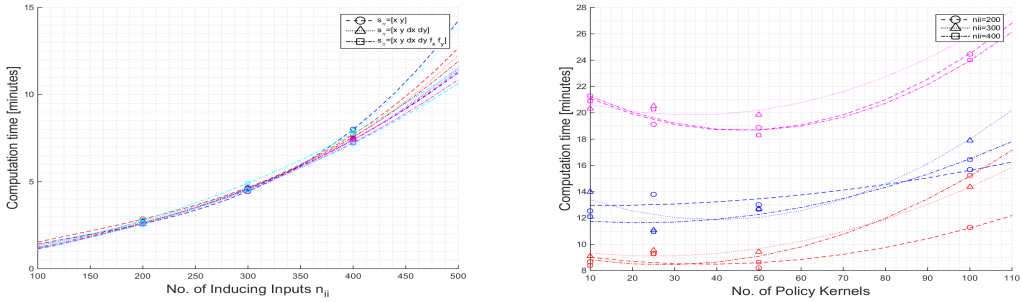


Figure 5-17: *Computational burden for training (left, colours indicate N_c) and prediction (right, colours indicate s_π) of planar VIL.*

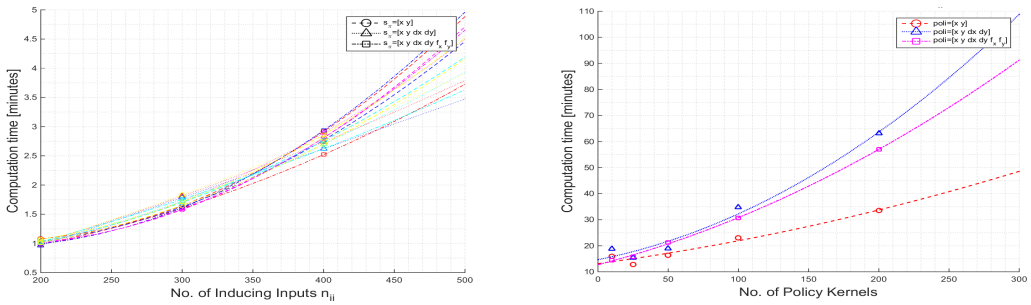


Figure 5-18: *Computational burden for training (left, colours indicate N_c) and prediction (right) of planar VILTA.*

Figure 5-17 (left) shows the average CPU time for a single model update in VIL, including 300 line-searches for fitting the sparse GP. It can be seen that both the number of policy inputs

[7] As stated earlier, the sequential nature of dynamical simulation does not allow for parallel threads and hence the physical architecture of a high core-count CPU offers limited advantage.

and basis functions have little effect as expected. The defining variable is number of inducing inputs. For the number chosen in this thesis ($N_{ii} = 300$), fitting 6 output dimensions (see Eq. (4-3)) we get $T_{train} = 2$ [min], for $N_{ii} = 400$ this has increased to almost 8 minutes. The right plot in Figure 5-17 reveals that a number of basis functions between $n_c = 10$ and $n_c = 50$ has little effect on the time needed for a single policy update, confirming the expectation that the number of policy parameters can be increaed relatively cheaply compared to other learning methods [24]. The inducing input count again does have an influence and the number of policy basis functions can interestingly be increased from 10 to 50 at almost no cost and doubled from 50 to 100 for a 25% increase in learning time. Setting the number of inducing inputs to $N_{ii} = 300$, the total episode time amounts to $T_{ep} = 17$ [min].

Scalability

What do these findings mean for the scalability of the proposed algorithm?
For the sake of this discussion, let us imagine full 6-DOF VIL with 300 sparse inputs and

$$\pi(\boldsymbol{s}_e) = \boldsymbol{k}_P \in \mathbb{R}^{D_u=6}, \quad \boldsymbol{x} = [\boldsymbol{s}_e \;\; \dot{\boldsymbol{s}}_e \;\; \boldsymbol{k}_P]^T \in \mathbb{R}^{D=18}, \quad \boldsymbol{y} = [\boldsymbol{s}_e \;\; \dot{\boldsymbol{s}}_e \;\; \boldsymbol{W}_{ext}]^T \in \mathbb{R}^{E=18} \tag{5-3}$$

For training we found[8] in Chapter 4-4 that $\mathcal{O}_{tr}\left((N_{ii}^3(D+2)+2)E\right) \propto D, E$. Without using parallel threads, for the full spatial case this suggests

$$T_{train}^{full} \approx 2 \cdot (18/4) \cdot (18/6) = 27 \text{ [min]}$$

Now for policy learning, from Figure 5-17 it is seen that $T_{learn} = 19$ [min] when $N_c = 50$ and $D_u = 6$ (right plot). For the sake of argument we can assume only the pose to serve as policy input in the spatial case to keep D_u constant. We had found that prediction complexity scales as $\mathcal{O}_{pl}(E^2N^2D)$. In other words if we consider learning quadratically proportional to the number of GPs and linearly to the number of inputs to those models and to the number of policy outputs, this would give:

$$T_{learn}^{full} \approx 19 \cdot (18/6)^2 \cdot (12/4) \cdot (6/2) = 2308 \text{ [min]} \approx 38 \text{ [h]}$$

This means every policy update would amount to over a one and a half days of computation time. Even for the imagined example of interplanetary rovers in Chapter 2-3 these time-spans are at the very least impractical.

Now it has to be said these are very crude extrapolations without any dimensionality reduction applied to the problem. For example one could imagine the rotational stiffness gains to be shared or constant, certain dimensions might not have to be modelled, or in the case of symmetric tasks policy outputs can be shared for multiple dimensions. However the spirit of this extrapolation does tell us something about the overall scalability of the proposed algorithm on current hardware. Especially in light of the conclusions presented in Chapter 6, which suggest several ways of mitigating the weaknesses of the algorithm, but will likely inject more complexity and thus computational burden into the overall architecture of the learning problem.

[8] For constant line-search optimization settings.

Chapter 6

Conclusions and Future Directions

This thesis has detailed the theoretical underpinnings, development and implementation of a model-based algorithm for learning Variable Impedance Control (VIC). What follows is a summary of, and conclusions on, the important points of this work based on the research objectives formulated in Chapter 1-2. Concluded by a rationale of the author on what further research directions are worth considering in order to move us forward based on the insights learned over the course of this thesis.

6-1 Summary and Conclusions

Chapter 1 introduced a number of ways in which Bayesian inference is used in biological learning along with a number of recent approaches to learning robotic force control. To recreate this behaviour in robotic force control with the goal of minimizing the required interaction time for learning constrained motion, a model-based approach using Gaussian Processes was proposed.

Chapter 2 presented the mathematical framework for Cartesian Impedance Control (IC). First of all it must be stated that the design choices for IC are not in general straightforward and research into new architectures is an ongoing endeavour. Two methodologies were outlined; Simple Impedance Control (SIC), suitable for back-drivable, low inertia robots with a natural ability to deal with singularities. In General Impedance Control (GIC) the system dynamics of the robot are partially decoupled and linearized through feedback. Only recently has ideal IC been significantly approximated by the development of light-weight torque-controlled manipulators.

The concepts related to Reinforcement Learning (RL) and Gaussian Process (GP) models were introduced in Chapter 3, in particular with respect to the Probabilistic Inference for Learning COntrol (PILCO) algorithm. This model-based RL algorithm uses learned GPs to perform long-term simulations in a fully Bayesian manner. Due to its high sample-complexity, the PILCO algorithm has shown to yield great improvements in the required interaction time.

A drawback from a system's point of view on GPs is that the hyper-parameters in a non-parametric model do not usually yield a mechanical or physical interpretation. This can make the resulting models opaque to engineers trying to analyze its behaviour.

Chapter 4 introduced several contributions for model-based learning of impedance control tasks, in line with the first research objective. These included the particular state and action spaces used in planning for Variable Impedance Learning (VIL) and Variable Impedance Learning with Trajectory Adaptation (VILTA) with partial non-negative saturated policies. Relative targets with informative reference priors accommodated closed-loop system identification. Another contribution was made in the form of an explicit energy penalty in the expected return in the framework of Gaussian state trajectories. Borrowing from studies into human sensorimotor control, saturated exponential accuracy cost were combined with a normalized quadratic energy cost acting on part of the action space to mimic the reward function in biological skill learning. Analytical gradients were derived for the extended return function. The concepts put forth here do not need to be constrained by the exact form investigated in this work and can be applied to any Bayesian model-based learning setting.

Finally Chapter 5 presented results for simulated and physical experiments in VIL and VILTA. The proposed learning framework succeeded in a variable impedance planar Peg-in-Hole (PiH) task without any predetermined gain schedules and, in the case of VILTA, any kinematic motions plans or knowledge of the target location.
For VIL, policy convergence to a local optimum is often reached within 10 interactions, including for linear policies. Recent work on model-free VIL of force-perturbing tasks still required up to 200 episodes for policy convergence[1] [1]. Although due to the current application being limited to a planar setting and a different type of task, we should be cautious in drawing conclusions on this decrease of interaction time in the order of a magnitude. An interesting application was found through the combination of VIL and SIC, which benefited the behaviour of a posterior model. The variable stiffness gains were able to partially mitigate the weaknesses of SIC while maintaining its benefits.
VILTA was capable of finding solutions for learning stiffness and trajectories simultaneously, deducing the hole's location in a common-sense manner, including for distributed start states.

The developed algorithm is by no means free of problems. The final objective was investigating the effects of local discontinuities and how uncertainty bookkeeping could help in mitigating their effect. It was found that singular GP system identification is not fully reliable in capturing the global dynamics for the scenarios considered. The discrete nature of the task could steer the sparse approximated model off-course. In simulation this was mainly due to simplistic contacts and in the real-world experiments likely due to friction and contact transitions. Force-conditioned policies could not implemented. Mainly due to the tendency of the policy venturing into dangerous part of the action space. Contact forces would induce abrupt stiffness changes, creating oscillations and finally instability. Better reward shaping could help in this.

[1]The advantage for learning of such tasks over scenarios presented here is the absence of obvious discontinuities during the trials. It would be interesting to see how this framework performs in such a setting.

6-2 Future Directions

It is hard to say how far away the imagined ideal of full scale humanoids operating effortlessly in the real world is. What seems to be clear is that deterministic approaches developed for well-behaved problems and clear structured rule systems will not apply to any of the current robotic frontiers. The PILCO algorithm and its extension presented here are an interesting step into more intelligent, human-like Bayesian inference and advances in computer hardware will partially offset the scalability pitfalls. However to expand the sophistication of robotic sensorimotor skill-learning algorithms, more generalized problem formulations will need to be combined with further improvements in variable impedance hardware. Research into biological physiology and neurology will help in crystallizing the way forward. We might not have build flapping-wing aircrafts, but birds are not the only things capable of generating lift in an atmosphere. For intelligent and effective manipulation though, it seems logical that biology is still our best guide.

A promising step was taken with Dynamic Movement Primitives (DMPs) in [35] and [36] combined with the path integral approaches to VIL in [38]. It was considered using DMPs in this work for encoding the learned kinematic trajectories, before deciding against it for two reasons; the added complexity of did not immediately outweigh the benefits from more generalized movement representations; secondly the current framework of DMPs is based around constant goal state attractors. For this work, the exact goal state was often unknown. An interesting extension could lie in variable, possibly learned, attractor targets for DMPs.

Another issue is the effect of input noise. The learned GPs incorporate an estimate of the output noise, but assume noiseless inputs. Since we are dealing with sequential time-series data, this is obviously untrue. This leads to estimation errors especially in parts of the state-space with fast changing dynamics [75]. One way of dealing with this is to project the input noise to the output. To render the problem tractable, this technique should be combined with sparse GPs.

For resolving some of the problems with respect to local discontinuities, several solutions can be investigated. One possibility is different choices of covariance functions. The exponential used in this thesis is attractive due to its infinite smoothness, however this is probably too stringent of a requirement, especially in the context of interaction. The *Matern* function for example is a kernel that is still twice differentiable which might allow more drastic function changes to accommodate contact transitions. To this end, non-stationary covariance functions such as in [76] are likely even more suitable to adapt to functions whose smoothness varies with the inputs. These non-stationary kernels allow the local differentiability of the GPs to be learned from data to accommodate abrupt changes. *Local Gaussian Process Models* [77], [78] can also serve this purpose. These are multiple smaller-scale GPs that model only a part of the state-action space. For contact environments, it would be interesting to see if the boundaries between the models can be learned. Predictions along the boundaries of these regions can be a mixture of surrounding local solutions. Another approach is through hierarchical GP models [79], which have already been applied to human motion studies in [80].

It is however also important to reflect on the larger applicability and consider honestly the viability of pursuing certain directions. The PILCO algorithm has scored genuinely impressive improvements in learning from scratch. It is however also a complex algorithm in terms of its meticulous analytical bookkeeping and mathematical underpinnings. Whether injecting more complexity into this framework through the use of local or hierarchical models remains to be seen.

This thesis started by citing an observation known as the "*Moravec Paradox*" [8], which states the counter-intuitive propositions that learning effective and robust sensorimotor skills is far more difficult and computationally intensive than what is normally considered sophisticated reasoning in humans. If nothing else, of these two claims, this thesis has certainly been an exhibition of the former and a useful exercise in the latter.

Appendix A

Mathematical Background

A-1 Statistical Properties

Assume x and y are two sets of random variables.
Then the *expected value* and *variance* is

$$\mathbb{E}[x] = \int xp(x)dx \qquad \mathrm{Var}[x] = \sigma_x^2 = \mathbb{E}[(x - \mathbb{E}[x])^2] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 \tag{A-1}$$

and the *covariance* is

$$\mathrm{Cov}[x, y] = \mathbb{E}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])] = \mathbb{E}[xy] - \mathbb{E}[x]\,\mathbb{E}[y] \tag{A-2}$$

The *joint probability* is given by

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x) = p(y, x) \tag{A-3}$$

The *marginal* or *evidence* is found by integrating over the joint:

$$p(x) = \int p(x, y)\,\mathrm{d}p(y) = \int p(x|y)p(y)\,\mathrm{d}p(y) \tag{A-4}$$

Which allows defining a *conditional probability* known as *Bayes' Theorem.*

$$p(y|x) = \frac{p(y, x)}{p(x)} = \frac{p(x|y)p(y)}{p(x)} \tag{A-5}$$

Adding two random variables results in

$$\mathbb{E}[x + y] = \mathbb{E}[x] + \mathbb{E}[y] \qquad \mathrm{Var}[x + y] = \mathrm{Var}[x] + \mathrm{Var}[y] + \mathrm{Cov}[x, y] + \mathrm{Cov}[y, x] \tag{A-6}$$

Other useful manipulations are the *Law of Iterated Expectations*

$$\mathbb{E}[x] = \mathbb{E}_y[\mathbb{E}_x[x|y]] \quad (= \mathbb{E}_{y,x}[x|y]) \tag{A-7}$$

and the *Law of Total (Co-)Variance*

$$\text{Var}[x] = \mathbb{E}_y[\text{Var}[x|y]] + \text{Var}_y[\mathbb{E}[x|y]] \tag{A-8}$$

In the case of a Gaussian random variable x, the *probability density function* is

$$pdf(x) = p(x) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{A} + \boldsymbol{B}|^{-1} \exp\left(-\frac{1}{2}(x - \boldsymbol{\mu}_x)^T \Sigma^{-1} (x - \boldsymbol{\mu}_x)\right) \tag{A-9}$$

The marginal distributions of the joint

$$p(x_1, x_2) = \mathcal{N}\left(\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right) \tag{A-10}$$

are arrived at by selecting the proper parts

$$p(x_1) = \mathcal{N}(\mu_1, \Sigma_{11}) \qquad\qquad p(x_2) = \mathcal{N}(\mu_2, \Sigma_{22}) \tag{A-11}$$

Now the conditional distributions are given by the following formula

$$p(x_2|x_1) = \mathcal{N}\left(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x_1 - \mu_1), \quad \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right) \tag{A-12}$$

Which is a result of completing the squares in a *Schur complement*.

The *product* of 2 Gaussians $\mathcal{N}(\boldsymbol{a}, \boldsymbol{A})\mathcal{N}(\boldsymbol{b}, \boldsymbol{B}) = \text{c} \cdot \mathcal{N}(\boldsymbol{c}, \boldsymbol{C})$ is another Gaussian distribution, multiplied by a factor c which is itself a Gaussian distribution:

$$\begin{aligned}
\boldsymbol{C} &= \left(\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1}\right)^{-1} \\
\boldsymbol{c} &= \boldsymbol{C}\left(\boldsymbol{A}^{-1}\boldsymbol{a} + \boldsymbol{B}^{-1}\boldsymbol{b}\right) \\
c &= (2\pi)^{-\frac{D}{2}} |\boldsymbol{A} + \boldsymbol{B}|^{-1} \exp\left(-\frac{1}{2}(\boldsymbol{a} - \boldsymbol{b})^T (\boldsymbol{A} + \boldsymbol{B})^{-1} (\boldsymbol{a} + \boldsymbol{b})\right)
\end{aligned} \tag{A-13}$$

Which is different from the so called *product normal distribution*.

A-2 Gradient Descent Optimization

The optimization algorithms used for conditioning the Gaussian Process (GP) and policy updates are Limited-memory Broyden-Fletcher-Gretchen-Shanno ((L)BFGS) and Conjugate Gradient (CG). Both these techniques are *line-search methods*, which are effective iterative algorithms to compute local minimizers in unconstrained optimization problems. In general, each iteration k of a line search algorithm computes a search direction $\boldsymbol{p}_k$ and a positive step length α_k along the search direction that satisfies a sufficient decrease in the function as measured by the inequality

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) \leq f(\boldsymbol{x}_k) + c_1 \alpha_k \nabla f(\boldsymbol{x}_k)^T \boldsymbol{p}_k$$

for some constant c_1. This condition is the first of the *Wolfe conditions*[1]. In the context of line search, this condition could be satisfied for all sufficiently small values of α, so it may not

[1] Also called *Armijo condition*.

be enough by itself to ensure fast convergence and thus another condition is proposed, called the curvature condition:

$$\nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)^T \boldsymbol{p}_k \leq c_2 \nabla f(\boldsymbol{x}_k)^T \boldsymbol{p}_k$$

for some constant c_2 satisfying $c_1 \leq c_2 < 1$. Once a scalar α_k has been found that satisfies the Wolfe conditions we can update the unknown $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$, $s_k = \alpha_k \boldsymbol{p}_k$ being the accepted step at the current iterate.

Most line search algorithms require $\boldsymbol{p}_k$ to be a descent direction.
In other words: $\boldsymbol{p}_k^T \nabla f(\boldsymbol{x}_k) < 0$. Which ensures reducing $f(x)$ along this direction. In steepest descent, $\boldsymbol{p}_k$ is simply the opposite of the gradient $\boldsymbol{p}_k = -\nabla f(\boldsymbol{x}_k)$.
For the quasi-Newton method (L)BFGS, $\boldsymbol{p}_k$ is computed by minimizing the predictive quadratic model

$$m_k(\boldsymbol{p}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^T \boldsymbol{p} + \frac{1}{2} \boldsymbol{p}^T H_k \boldsymbol{p}$$

leading to $\boldsymbol{p}_k = -H_k^{-1} \nabla f(\boldsymbol{x}_k)$. where H_k is a symmetric and non-singular approximation of the Hessian updated at every line search. Then when $\boldsymbol{p}_k$ is defined in this way and the matrix H_k is positive definite, we have

$$\boldsymbol{p}_k^T \nabla f(\boldsymbol{x}_k) = -\nabla f(\boldsymbol{x}_k)^T H_k^{-1} \nabla f(\boldsymbol{x}_k)$$

Which is negative definite and hence $\boldsymbol{p}_k$ is a descent direction [81].

CG relies on the fact that the solution $\boldsymbol{x}^*$ to the linear matrix equality $A\boldsymbol{x} = \boldsymbol{b}$ for the known symmetric and positive definite $A \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$ can be expressed in the basis $P = \{\boldsymbol{p}_1 \dots \boldsymbol{p}_n\}$ of n *mutually conjugate* vectors as

$$\boldsymbol{x}^* = \sum_{i=1}^{n} \alpha_i \boldsymbol{p}_i$$

Filling the known conjugate directions allows solving for $\boldsymbol{\alpha}$. Now if n is large, as is the case for us, we can approximate the solution by iteratively considering the first k entries of P. This is done by realizing that $\boldsymbol{x}^*$ also minimizes the quadratic form $f(\boldsymbol{x}) = \boldsymbol{x}^T A \boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{b}$. Then starting from an arbitrary guess of $\boldsymbol{x}_0$, the first search direction is taken equal to the gradient of $f(\boldsymbol{x})$: $\boldsymbol{p}_0 = \nabla f(\boldsymbol{x}_0) = A\boldsymbol{x}_0 - \boldsymbol{b}$. This leaves a residual of $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k$, which is the negative gradient of $f(\boldsymbol{x}_k)$. Thus in steepest descent the next direction would be along $\boldsymbol{r}_k$. Instead in CG we insist that the new direction $\boldsymbol{p}_{k+1}$ be conjugate to the gradient and hence the name *Conjugate Gradient*. This gives an expression for the next direction:

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} - \sum_{i=1}^{k} \frac{\boldsymbol{p}_i^t A \boldsymbol{r}_{k+1}}{\boldsymbol{p}^T A \boldsymbol{p}_i} \boldsymbol{p}_i = \boldsymbol{r}_{k+1} + \frac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^T \boldsymbol{r}_k} \boldsymbol{p}_k$$

Where the last equality holds because $\boldsymbol{r}_{k+1}$ is orthogonal to all $\boldsymbol{p}_i$ for $i < k$ and we thus only need to maintain memory of the previous iterate. $\boldsymbol{p}_{k+1}$ can now be used as the direction for a line search $\boldsymbol{x}_{k+1} = \boldsymbol{x}(k) + \alpha_k \boldsymbol{p}_k$, where

$$\alpha_k = \frac{\boldsymbol{r}_k^T \boldsymbol{r}_k}{\boldsymbol{p}_k^T A \boldsymbol{p}_k}$$

Appendix B

Orientation Representation

A point that needs to be addressed in operational space control is the particular choice of orientation representation. For an analysis and comparison of all methods for orientation representation the interested reader is referred to [82]. What follows is a derivation for the representation used in this thesis, namely the *relative Euler angles*, based on the work in [51].

The choice of axes can be divided into two categories: *proper Euler angles* and *Tait-Bryan angles*. As eluded to in Chapter 2-2-2, one can see that all Euler angles suffer from two representation singularities occurring when the first and last axes of rotation in the sequence lie along the same direction. For proper Euler angles this occurs at $\beta = 0$. For the Tait-Bryan angles their singularity lies at $\beta = \pm\frac{\pi}{2}$. When articulating the relative spatial orientation of the end-effector with respect to a certain reference in a specific manner, the chances of encountering this singularity can be minimized.

Proper Euler angles perform the first and last rotation about the same axis (zxz or $zx'z''$), while the latter performs all three rotations about distinct axes (xyz or $xy'z''$). In both cases, each successive rotation may be performed about the axes of the original coordinate system[1], or they may be performed about the axes of the coordinate system resulting from the previous rotation, indicated by an apostrophe.
The most widely used proper angles are $zx'z''$. However this formulation has a representational singularity at $\beta = 0$ and for this reason are not ideal for spatial impedance control. The following Tait-Bryan angles are used in this thesis:

$$R_e(\boldsymbol{\phi}) = R_x(\alpha)R_y^{'}(\beta)R_z^{''}(\gamma) \tag{B-1}$$

This succession of rotations is often referred to as aerospace-, or roll-pitch-yaw angles. The angular velocity $\boldsymbol{\omega}_e$ is related to the time derivative of the rotation matrix R_e through the skew symmetric operator S:

$$\dot{R}_e = S(\boldsymbol{\omega}_e)R_e \tag{B-2}$$

[1]These types of Euler angles are called *extrinsic rotations* and its counterpart as *intrinsic* rotations.

Differentiating the mapping between Euler angle derivatives and angular velocity yields the angular acceleration of the end-effector in terms of its Euler angles:

$$\dot{\boldsymbol{\omega}}_e = T(\boldsymbol{\phi}_e)\ddot{\boldsymbol{\phi}}_e + \dot{T}(\boldsymbol{\phi}_e)\dot{\boldsymbol{\phi}}_e \tag{B-3}$$

The vector $\boldsymbol{a}$ represents a resolved acceleration that can be decomposed into a linear and angular part $\boldsymbol{a} = \begin{bmatrix}\boldsymbol{a}_p^T & \boldsymbol{a}_o^T\end{bmatrix}^T$. Which will result in the resolved acceleration $\ddot{\boldsymbol{x}}_e = \begin{bmatrix}\ddot{\boldsymbol{p}}_e^T & \dot{\boldsymbol{\omega}}_e^T\end{bmatrix}^T$. For the positional impedance control, the choice of Euler angles naturally does not make a difference. Now according to traditional operational space control, the orientation error is defined as the difference in Euler angles between the desired and end-effector frame:

$$\boldsymbol{\Delta\phi}_{de} = \boldsymbol{\phi}_d - \boldsymbol{\phi}_e$$

In view of (B-3), the resolved acceleration is then chosen as

$$\boldsymbol{a}_o = T(\boldsymbol{\phi}_e)\left(\ddot{\boldsymbol{\phi}}_d + K_{D_o}\dot{\boldsymbol{\phi}}_{de} + K_{P_o}\boldsymbol{\phi}_{de}\right) + \dot{T}(\boldsymbol{\phi}_e, \dot{\boldsymbol{\phi}}_e)\dot{\boldsymbol{\phi}}_e$$

The above manner of Euler angles feedback becomes ill-conditioned when the actual (absolute) end-effector orientation approaches a representation singularity. Since for a general trajectory this is bound to happen despite the choice of Euler angles, an alternative approach is adopted that described in Appendix B.
Instead of the absolute orientation, the Euler angles are extracted from the relative orientation between the end-effector and a desired reference frame.

$$R_d^e = R_e^T R_d \tag{B-4}$$

We now define $\boldsymbol{\phi}_{de}$ as the set of Euler angles extracted from R_d^e, who's temporal derivative is $\dot{R}_d^e = S(\Delta\boldsymbol{\omega}_{de})R_d^e$. Where $\boldsymbol{\Delta\omega}_{de}$ is the end-effector angular velocity error. Which is related to relative Euler angle derivative through

$$\boldsymbol{\Delta\omega}_d e = T(\boldsymbol{\phi}_{de})\dot{\boldsymbol{\phi}}_{de}$$

The resolved angular acceleration can now be set to

$$\boldsymbol{a}_o = \boldsymbol{\omega}_d - \dot{T}_e(\boldsymbol{\phi}_{de})\dot{\boldsymbol{\phi}}_{de} + T_e(\boldsymbol{\phi}_{de})\left(K_{D_o}\dot{\boldsymbol{\phi}}_{de} + K_{P_o}\boldsymbol{\phi}_{de}\right) \tag{B-5}$$

Where $T_e(\boldsymbol{\phi}_{de})$ is the transformation between the angular acceleration and the derivative of the Euler angles [51], referred to the static base-frame by $T_e(\boldsymbol{\phi}_{de}) = R_e T(\boldsymbol{\phi}_{de})$.

The clear advantage of the alternative over the classical Euler angles feedback is that, by adopting a representation for which $T(\mathbf{0})$ is nonsingular, representation singularities occur only for large orientation errors. For this reason, the widely adopted choice zyz-angles is not recommended, since it's singularity will occur at $\boldsymbol{\phi}_{de} = \mathbf{0}$. This motivates the choice for roll-pitch-yaw angles, whose singularity lies at a relative orientation error of $|\beta_{de}| = \frac{\pi}{2}$.

As a final remark, the above operational space control schemes have been derived with reference to a minimal description of orientation in terms of Euler angles. It is understood that it is possible to adopt different definitions of orientation error such as the angle/axis or the unit quaternion. The advantage is the use of the geometric Jacobian in lieu of the analytical Jacobian, thereby completely eliminating representational singularities. The price to pay, however, is a more complex analysis of the stability and convergence characteristics of the closed-loop system. Even the inverse dynamics control scheme will not lead to a homogeneous error equation, and a Lyapunov argument should be invoked to ascertain its stability. Due to its added complexity these representations are not considered here, but are used on the KUKA robot.

Bibliography

[1] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics*, vol. 30, pp. 432–450, June 2011.

[2] F. Stulp and O. Sigaud, "Policy improvement methods: Between black-box optimization and episodic reinforcement learning," *HAL Archives-Ouvertes*, vol. 14, p. 34, October 2012.

[3] B. Siciliano, L. Sciavicco, and L. Villani, *Robotics: modelling, planning and control*. Advanced Textbooks in Control and Signal Processing, London: Springer, 2009.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning - An Introduction*. The M.I.T. Press, 1998.

[5] M. P. Deisenroth, *Dissertation: Efficient Reinforcement Learning using Gaussian Processes*. KIT Scientific Publishing, Cambridge, December 2009.

[6] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 408–423, February 2015.

[7] S. Pinker, *The language Instinct*. William Morrow and Company, 1994.

[8] H. Moravec, *Mind Children - The Future of Robot and Human Intelligence*. Harvard University Press, 1988.

[9] K. Tee, D. Franklin, M. Kawato, T. Milner, and E. Burdet, "Concurrent adaptation of force and impedance in the redundant muscle system," *Biological Cybernetics*, vol. 102, no. 1, pp. 31–44, 2010.

[10] E. Burdet, R. Osu, D. W. Franklin, T. Milner, and M. Kawato, "The central nervous system stabilizes unstable dynamics by learning optimal impedance," *Nature*, vol. 33, pp. 446–449, 2001.

[11] A. A. Blank, A. M. Okamura, and L. Whitcomb, "Task-dependent impedance and implications for upper-limb prosthesis control," *The International Journal of Robotics Research*, vol. 33, no. 6, pp. 827–846, 2014.

[12] N. Hogan, "Impedance control - an approach to manipulation part 1, 2 & 3," *Journal of Dynamic Systems, Measurement and Control*, vol. 107, pp. 1–24, March 1985.

[13] M. P. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *In Proceedings of the International Conference on Machine Learning*, 2011.

[14] D. Barry, "Nonparametric bayesian regression," *The Annals of Statistics*, vol. 14, pp. 934–953, September 1986.

[15] J. Kober, J. Peters, and J. A. Bagnell, "Reinforcement learning in robotics - a survey," *The International Journal of Robotics*, vol. 32, September 2013.

[16] J. F. Broenink and M. L. J. Tiernego, "Peg-in-hole assembly using impedance control with a 6 dof robot," in *Simulations in Industry - Proceedings 8th European Simulation Symposium*, pp. 504–508, October 1996.

[17] T. Meitinger and F. Pfeiffer, "The spatial peg-in-hole problem," in *Proceedings of the World Automation Congress*, May 1996.

[18] I. F. of Robotics, "World robotics 2015 industrial robots - http://www.ifr.org/industrial-robots/statistics/," December 2015.

[19] D. M. Wolpert, J. Diedrichsen, and J. R. Flanagan, "Principles of sensorimotor learning," *Nature Reviews Neuroscience*, vol. 12, pp. 739–751, December 2011.

[20] J. W. Krakauer and P. Mazzoni, "Human sensorimotor learning: adaptation, skill and beyond," *Current Opinion in Neurobiology*, vol. 21, p. 636–644, August 2011.

[21] K. Kording and D. Wolpert, "Bayesian integration in sensorimotor learning," *Nature*, vol. 427, pp. 244–247, January 2004.

[22] M. Kawato, Y. Uno, M. Isobe, and R. Suzuki, "Hierarchical neural network model for voluntary movement with application to robotics," *Control Systems Magazine*, vol. 8, pp. 8 – 15, April 1988.

[23] V. Gullapalli, R. A. Grupen, and A. G. Barto, "Learning reactive admittance control," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, vol. 2, pp. 1475–1480, May 1992.

[24] M. P. Deisenroth, G. Neumann, and J. Peters, "Direct policy search in robotics," *Foundations and Trends in Robotics*, vol. 2, pp. 1–142, January 2011.

[25] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[26] R. J. Williams, "Simple statistical gradient following algorithm for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 23, pp. 229–256, 1992.

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

[28] S. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, p. 251–276, February 1998.

[29] S. Kakade, "A natural policy gradient," *Advances in neural information processing systems*, vol. 14, 2001.

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

[31] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1-2, pp. 171–203, 2011.

[32] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber, "Parameter-exploring policy gradients," *Neural Networks*, vol. 23, pp. 551–559, October 2010.

[33] T. Ruckstiess, F. Sehnke, T. Schaul, D. Wierstra, Y. Sun, and J. Schmidhuber, "Exploring parameter space in reinforcement learning," *Paladyn*, vol. 1, no. 1, pp. 14–24, 2010.

[34] J. Peters and S. Schaal, "Reinforcement learning by reward-weighted regression for operational space control," in *In: Proceedings of the International Conference on Machine Learning (ICML*, pp. 745–750, 2007.

[35] E. A.Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 11, November 2010.

[36] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, pp. 328–373, February 2013.

[37] F. Stulp, J. Buchli, A. Ellmer, M. Mistry, E. Theodorou, and S. Schaal, "Model-free reinforcement learning of impedance control in stochastic environments," *IEEE Transactions on Autonoous Mental Development*, vol. 4, pp. 330–341, December 2012.

[38] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4639 – 4644, IEEE, September 2003.

[39] J. F. Abu-Dakka, B. Nemec, J. Jorgensen, T. Savarimuthu, N. Kruger, and A. Ude, "Adaptation of manipulation skills in physical contact with the environment to reference force profiles," *Autonomous Robots*, vol. 39, no. 2, pp. 199–217, 2015.

[40] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 4, pp. 3557–3564 vol.4, Apr 1997.

[41] E. W. Aboaf, S. M. Drucker, and C. G. Atkeson, "Task-level robot learning: Juggling a tennis ball more accurately," in *Proceedings 1989 IEEE Int Conf Robotics and Automation*, pp. 1290–1295, Springer, 1989.

[42] T. Hester and P. Stone, *Learning and Using Models*. Springer-Verlag, 2012.

[43] P. Abbeel, M. Quigley, and A. Ng, "Using inaccurate models in reinforcement learning," in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 1–8, ACM, 2006.

[44] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[45] M. T. Mason, "Compliance and force control for computer controlled manipulators," *Systems, Man and Cybernetics*, vol. 11, pp. 418 – 432, June 1981.

[46] M. Vukobratovic, *Dynamics and Robust Control of Robot-Environment Interaction*. World Scientific Publishing Company, March 2009.

[47] S. Chiaverini and B. Siciliano, "A survey of robot interaction control schemes with experimental comparison," *IEEEE/ASME Transactions on Mechatronics*, vol. 4, September 1999.

[48] S. G. Khan, G. Herrmann, M. A. Grafi, T. Pipe, and C. Melhuish, "Compliance control and human-robot interaction," *International Journal of Humanoid Robotics*, vol. 11, September 2014.

[49] D. E. Whitney, "Historical perspective and state of the art in robot force control," in *Robotics and Automation*, vol. 2, pp. 262 – 268, IEEE, March 1985.

[50] J. de Schutter, H. Buryninck, and M. W. Spong, "Force control - a birds eye view," in *Control Problems in Robotics and Automation: Future Directions*, pp. 1–17, Springer Verlag, 1997.

[51] F. Caccavale, B. Siciliano, and L. Villani, "The role of euler parameters in robot control," *Asian Journal of Control*, vol. 1, pp. 25–34, March 1999.

[52] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 43–53, February 1987.

[53] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control - the Computer Control of Robot Manipulators*. MIT Press Series, 1981.

[54] C. O. R. Mukherjee and Y. Nakamura, "Unified impedance and admittance control," in *International Conference on Robotics and Automation*, pp. 554 – 561, IEEE, May 2010.

[55] A. Albu-Schaffer, C. Ott, and G. Hirzinger, *A Unified Passivity Based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots*. Springer Berlin Heidelberg, March 2007.

[56] Y. Seung-kook, "Compliant manipulation for peg-in-hole: Is passive compliance a key to learn contact motion?," in *Robotics and Automation*, pp. 1647 – 1652, IEEE, May 2008.

[57] Y. Xu, H. B. Yue, and C. H. Lei, "Precision peg-in-hole assembly strategy using force-guided robot," in *3rd International Conference on Machinery, Materials and Information Technology Applications*, Atlantis Press, January 2015.

[58] K. Sharma, V. Shirwalkar, and P. K. Pal, "Intelligent and environment-independent peg-in-hole search strategies," in *Control, Automation, Robotics and Embedded Systems (CARE), 2013 International Conference on*, pp. 1–6, December 2013.

[59] H. Park, J. Bae, J. Park, M. Baeg, and J. Park, "Intuitive peg-in-hole assembly strategy with a compliant manipulator," in *Robotics (ISR), 2013 44th International Symposium on*, pp. 1–5, October 2013.

[60] W. Newman, Y. Zhao, and Y.-H. Pao, "Reinforcement learning by reward-weighted regression for operational space control," in *Proceedings of the International Conference on Robotics and Automation*, pp. 571–577, IEEE, May 2001.

[61] R. A. Howard, *Dynamic Programming and Markov Processes.* The M.I.T. Press, 1960.

[62] A. Ng and M. Jordan, "Pegasus: A policy search method for large mdps and pomdps," in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.

[63] T. Bayes, "An essay towards solving a problem in the doctrine of chances," *Philosophical Transactions*, vol. 53, p. 370–418, January 1763.

[64] R. Neal, *Dissertation: Bayesian Learning for Neural Networks.* University of Toronto, 1995.

[65] K. Kording and D. Wolpert, "The loss function of sensorimotor learning," in *Proceedings of the National Academy of Sciences of the United States of America*, PNAS, 2004.

[66] J. Diedrichsen, "Optimal task-dependent changes of bimanual feedback control and adaptation," *Current Biology*, vol. 17, p. 1675–1679, October 2007.

[67] S. Schaal, *Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics.* Springer Tokyo, 2006.

[68] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Neural Information Processing Systems*, vol. 18, 2006.

[69] J. Quinonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, December 2005.

[70] P. I. Corke, "A robotics toolbox for matlab," *IEEE Robotics Automation Magazine*, vol. 3, pp. 24–32, March 1996.

[71] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in MATLAB.* Springer, 2011. ISBN 978-3-642-20143-1.

[72] L. J. Love and W. J. Book, "Environment estimation for enhanced impedance control," in *Robotics and Automation*, pp. 1854 –1859, IEEE, May 1995.

[73] S. Virga and M. Esposito, "Iiwa ros stack," July 2016.

[74] Mathworks, "Matlab robotics system toolbox," 2015.

[75] A. McHutchon, *Dissertation - Nonlinear Modelling and Control using Gaussian Processes*. University of Cambridge, August 2009.

[76] C. J. Paciorek and M. J. Schervish, "Nonstationary covariance functions for gaussian process regression," in *In Proceedings of the Conference on Neural Information Processing Systems*, MIT Press, 2004.

[77] F. Meier, P. Hennig, and S. Schaal, "Local gaussian regression," *CoRR*, vol. abs/1402.0645, March 2014.

[78] D. Nguyen-tuong, J. R. Peters, and M. Seeger, "Local gaussian process regression for real time online model learning," in *Advances in Neural Information Processing Systems 21*, pp. 1193–1200, Curran Associates, Inc., 2009.

[79] J. Shi, R. Murray-Smith, and D. Titterington, "Hierarchical gaussian process mixtures for regression," *Statistics and Computing*, vol. 15, no. 1, pp. 31–41, 2005.

[80] N. D. Lawrence and A. J. Moore, "Hierarchical gaussian process latent variable models," in *Proceedings of the 24th International Conference on Machine Learning*, Inernational Conference on Machine Learning, (New York, NY, USA), pp. 481–488, ACM, 2007.

[81] A. Tien, M. Fabian, and B. M. Toulouse, *On Optimization Algorithms for Maximum Likelihood Estimation*, December 2014.

[82] J. Diebel, *Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors*. Stanford, October 2006.

Glossary

List of Acronyms

CAGR	Compound Annual Growth Rate
IC	Impedance Control
DOF	Degrees of Freedom
IFR	International Federation of Robotics
CNS	Central Nervous System
NN	Neural Network
TCP	Tool Center Point
PiH	Peg-in-Hole
VIC	Variable Impedance Control
GIC	General Impedance Control
SIC	Simple Impedance Control
pHRI	physical Human-Robot-Interaction
AI	Artificial Intelligence
PI^2	Policy Improvements with Path Integrals
DMP	Dynamic Movement Primitive
pde	partial differential equations
HJB	Hamilton-Jacobi-Bellman
PoWER	Policy learning by Weighting Exploration with the Returns
NAC	Natural Actor-Critic

PEGASUS Policy Evaluation-of-Goodness And Search Using Scenarios

REINFORCE REward Increment = Non-negative Factor × Offset Reinforcement × Characteristic Eligibility

GP Gaussian Process

RL Reinforcement Learning

i.d.d. independent and identically distributed

ARD Automatic Relevance Determination

EM Evidence Maximization

LML Log Marginal Likelihood

(L)BFGS Limited-memory Broyden-Fletcher-Gretchen-Shanno

CG Conjugate Gradient

RBF Radial Basis Function

SPGP Sparse Pseudo-Inputs Gaussian Process

VIL Variable Impedance Learning

VILTA Variable Impedance Learning with Trajectory Adaptation

PILCO Probabilistic Inference for Learning COntrol

MDP Markov Decision Process

pdf probability density function

DLR Deutsches Zentrum für Luft- und Raumfahrt - German Aerospace Center

SNR Signal-to-Noise Ratio

ROS Robotic Operating System

List of Symbols

Abbreviations

$\boldsymbol{a}$	Action vector
$\boldsymbol{\eta}$	Gravity vector in Cartesian space
$\boldsymbol{\mu}$	Mean of multivariate distribution
$\boldsymbol{\omega}$	Angular velocity vector
$\boldsymbol{\phi}_e$	Euler angles of end-effector w.r.t. base
$\boldsymbol{g}$	Gravity matrix in joint space
$\boldsymbol{u}$	Input torque vector
$\boldsymbol{\psi}$	Hyper-parameter vector of policy
$\boldsymbol{q}$	Vector of robot joint angles.
$\boldsymbol{s}$	State vector
$\boldsymbol{s}_e$	Minimal Cartesian state vector
$\boldsymbol{s}_\pi$	Input states for the policy, subset of $\boldsymbol{y}$.
$\boldsymbol{\Sigma}$	Variance of multivariate distribution
$\boldsymbol{\theta}$	Hyper-parameter vector of Gaussian Process (GP) model
$\boldsymbol{W}_{ext}$	Wrench vector of external forces
$\boldsymbol{x}$	Independent variable (regressor)
$\boldsymbol{x}_e$	Pose of end-effector in operational space
$\boldsymbol{y}$	Dependent variable (regressand)
Δ	Discrete difference
$\dot{\boldsymbol{q}}$	Vector of robot joint angular velocities.
Γ	Coriolis/Centrifugal matrix in Cartesian space
Λ	Diagonal matrix of length-scales
$\mathbb{C}$	Set of all complex numbers
$\mathbb{R}$	Set of all real numbers
$\mathbb{Z}$	Set of all integer numbers
$\mathcal{A}$	Expected value of the energy cost function
$\mathcal{E}$	Expected value of the accuracy cost function
$\mathcal{S}$	Variance of the accuracy cost function
∇	Gradient operator
$\pi(\cdot)$	Policy
σ_f	GP function variance
σ_o	GP output noise hyper-parameter
Θ	Inertia matrix in Cartesian space
$\tilde{\pi}$	Unsaturated policy
C	Coriolis/Centrifugal matrix in joint space
D_u	Policy dimensionality
E	GP Target dimensionality

H_a^b	Homogeneous transformation matrix from frame a to b, belonging to SE(3).
J_a	Analytic Jacobian
J_g	Geometric Jacobian
J_π	Expected return
K_D	Desired Cartesian damping matrix
K_P	Desired Cartesian stiffness mtrix
M	Mass inertia matrix in joint space
m	Joint-space input dimensionality
N	Number of GP input points
n	Joint-space state dimensionality
$p(\cdot)$	Distribution
R_e	Rotation matrix in $\mathbb{R}^{3\times3}$ of end-effector orientation w.r.t. base
S	Selection and scaling matrix
t	Time index