Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Martin Hagen Myrestrand

Learning Compliant Robotic Manipulation

A Model-Based Approach Using Gaussian Processes

Master's thesis in Cybernetics and Robotics
Supervisor: Jan Tommy Gravdahl
Co-supervisor: Akhil S. Anand
July 2021

NTNU
Norwegian University of
Science and Technology

Martin Hagen Myrestrand

Learning Compliant Robotic Manipulation

A Model-Based Approach Using Gaussian Processes

Master’s thesis in Cybernetics and Robotics
Supervisor: Jan Tommy Gravdahl
Co-supervisor: Akhil S. Anand
July 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Preface

This thesis is written for the degree of Master of Science in Engineering Cybernetics, and is a continuation of the specialization project [1] from the fall of 2020. The thesis and the related work is part of SINTEF's project *ROMO*. The projects full working title being *"RObotics for Moving Objects within manufacturing and healthcare – enabling the future internet of things"*. The scope of the project is development of methods and algorithms that allow robots to interact with and handle moving objects in real time [2].

Being a continuation of my specialization project, there is some overlap in terms of relevant background information, software, and hardware. Thus, the following sections are adapted from [1]:

- Sections 2.1 - 2.4
- The subsection *Impedance Control* in 2.5
- Sections 3.1 - 3.4
- The subsection *Setup of the extended FCI* in 4.4

Acknowledgements

I want to thank my co-supervisor, PhD Candidate Akhil S. Anand, for a very thorough follow-up, providing exceptional assistance and guidance. I also want to thank my supervisor, Professor Jan Tommy Gravdahl, for valuable feedback and support. Additionally, I want to thank Postdoctoral Fellow Mathias H. Arbo for his assistance in learning how to operate the Franka Emika Panda robot.

Abstract

The area of compliant robotic manipulation is one of increasing importance in the field of robotic control. For the performance of complex real world interaction tasks, requiring human-like skills of adaptability, it is fundamentally important. Robotic ultrasound examination on humans is an example of such a task, demanding highly adaptable force control. In this thesis, we introduce learning techniques in a selection of low-level force controllers, setting out to achieve force tracking behaviour in robotic interaction tasks.

Most recent research in the field of robotic interaction control is aimed at achieving high flexibility and robustness. Prominent approaches often fall into the category of *Variable Impedance Control* (VIC), achieving flexibility by adjusting its dynamical properties during the execution of a task. Especially, there is an increasing focus on learning-based VIC, utilizing *Machine Learning* techniques to construct adaptive strategies. However, to achieve good performance, these methods usually require a lot of interactions during learning, entailing high wear and tear of the robot. Thus, the aim of this thesis is to develop compliant learning-based controllers that perform well in challenging scenarios, utilizing efficient sampling methods. For this purpose, the field of model-based *Reinforcement Learning* is of particular interest. By utilizing models of the system, model-based algorithms can predict states and actions, thereby improve policies through model simulation. Consequently, making it more sampling efficient, requiring fewer physical interactions.

In our approach, three different force controllers are implemented and assessed in the state of the art, sampling efficient PILCO framework, using Gaussian Processes (GP's) to incorporate model uncertainty into long-term planning. The controllers represent three fundamental approaches in robotic interaction control: *Admittance Control*, *Hybrid Force/Motion Control* and *Force-based VIC*. The approach is evaluated in the use case of performing robotic interaction with object surfaces, where the manipulator executes a motion trajectory while maintaining a desired contact force. Based on results from simulation and experimental studies, the controllers are analysed for their advantages and limitations in the learning-framework. Furthermore, the pros and cons of GP-modelling are investigated, leading to a discussion on future directions.

Table of contents

Mathematical notations

$\boldsymbol{q} \in \mathbb{R}^{n\times 1}$	joint angles
$\dot{\boldsymbol{q}} \in \mathbb{R}^{n\times 1}$	joint velocities
$\boldsymbol{p} \in \mathbb{R}^{3\times 1}$	position
$\boldsymbol{v} \in \mathbb{R}^{6\times 1}$	velocity, consisting of translational and rotational velocity
$\Delta\boldsymbol{p}_{de} \in \mathbb{R}^{3\times 1}$	error in position
$\Delta\boldsymbol{v}_{de} \in \mathbb{R}^{6\times 1}$	error in velocity
$\boldsymbol{h}_e \in \mathbb{R}^{6\times 1}$	external wrench
$\boldsymbol{h}_c \in \mathbb{R}^{6\times 1}$	applied wrench
$\boldsymbol{\tau} \in \mathbb{R}^{n\times 1}$	applied torque
$\boldsymbol{x} \in \mathbb{R}^{6\times 1}$	pose, consisting of position and orientation

Acronyms

BNN Bayesian Neural Network. 19, 68, 69

C-PI^2 Coordination Policy Improvement with Path Integral. 2, 19

DMP Dynamic Movement Primitive. 18

DOF Degrees of freedom. 5, 7, 8, 10, 13, 19, 22, 27

EMG Electromyography. 18

FCI Franka Control Interface. i, ii, viii, 27–29, 31, 36, 43, 62

GMR Gaussian Mixture Regression. 17

GP Gaussian Process. i, viii–x, 2, 19, 20, 23–26, 47, 55, 56, 64–69

HFMC Hybrid Force/Motion Control. 13

Hybrid Control Hybrid Force/Motion Control. viii–x, 13, 31, 36, 38, 39, 41, 44, 45, 47, 48, 50, 52, 54, 56, 57, 59–67, 69

IL Imitation Learning. 17

ILC Iterative Learning Control. 18

LfD Learning from Demonstration. 17

MSE Mean Square Error. viii–x, 40, 47–53, 55–61, 66

NN Neural Network. 19

PETS Probabilistic Ensembles with Trajectory Sampling. 19, 68

PI^2 Policy Improvement with Path Integrals. 2, 18, 19

PILCO Probabilistic Inference for Learning Control. i–iii, 2–4, 22, 31, 34–37, 47, 59, 62, 63, 66–69

RBF Radial Basis Function. ix, 22, 35, 47, 48, 63, 68

RL Reinforcement Learning. viii, 1, 2, 18–22, 30, 34, 35, 66

ROS Robot Operating System. vii, 27, 28, 36, 62

VIC Variable Impedance Control. i, viii–x, 1, 2, 15, 17, 19, 32, 36, 42, 44–46, 51, 53, 60–62, 65, 67–69

VIL Variable Impedance Learning. 17, 18

VILC Variable Impedance Learning Control. 17, 18

Glossary

Approximate inference Computationally efficient method to predict future rewards in model-based Reinforcement Learning [3].. 22

Gazebo An open-source 3D robotics simulator.. viii, 28, 29, 38, 62

GPflow v2 A package for building Gaussian Process models in Python (builds on Tensorflow v2) [4]. 30

MoveIt! Motion planning framework running on top of ROS, utilizing some of their common tools like the ROS Visualizer (Rviz) and the ROS robot format (URDF).. 28, 29

OpenAI An artificial intelligence research and deployment company [5].. 30, 34, 37

SINTEF One of Europe's largest independent research organisations [6].. i

Tensorflow v2 An end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources [7].. vii, 30

URDF (Unified Robotic Description Format) is an XML specification used in academia and industry to model multibody systems [8].. 28

List of Figures

List of Algorithms

1 Introduction

In the field of robotics, mechanical interaction with objects is arguably one of the fundamentally important functions. Numerous robot applications depend on it; for example, mechanical interaction is vital for manipulation, the principal task of assembly systems [15]. Moreover, new robot applications may be enabled by improved control of mechanical interactions. Proficient interaction control can for example be used in healthcare to aid people with reduced mobility, and to assist in patient assessments. Automatic ultrasound examination, where a manipulator autonomously is controlling a probe, is an example of the latter. In such cases, there are placed high demands on the flexibility of the robotic control. Furthermore, while finding and following organs and other bodily structures, it must, despite unexpected changes in its surroundings, provide safe force-constrained interactions.

In applications like this one, the "feel" of the robot becomes an important factor. The "feel", defined by the mechanical interaction dynamics, is therefore something we want to control and modulate [15]. This is the central idea of *Impedance Control* [16], a prominent approach in interaction control. When an impedance-controlled end-effector is exposed to contact forces, its dynamic response is designed to mimic one of a suitable mass-spring-damper system. In traditional impedance controllers, this dynamical response is fixed, meaning that the response is given by constant mass-, damping-, and stiffness parameters. However, for increased flexibility, these parameters can be altered during the execution of a task. Motivated by the great application-potential of highly flexible interaction controllers, a lot of research is aimed at the development of adaptive strategies.

Methods striving for this flexibility often fall into the broad category of *Variable Impedance Control* (VIC). Already in 1992, H.P. Huang and S.S. Chen presented both a position-based- and a force-based VIC [17]. These implementations, like many that have come in hindsight, are dependent on explicit adaptation laws, making parameter-adjustments based on state-errors. The derivation of such adaptation laws is far from trivial, requiring advanced knowledge of both the robot, the task, and the process of designing and parameterizing such controllers [18].

The tiresome process of designing explicit adaptation laws has motivated the employment of *Machine Learning* techniques. For example, in the approach of *Imitation of Human Impedance*, the controller is to replicate human bio-mechanical impedance strategies, developed over years of experience. The more universal approach of *Reinforcement Learning* (RL) is a promising way to learn impedance strategies without human intervention. In RL, the general idea is that a virtual *agent* learns optimal actions through trial and error. By specifying some desired behaviour, which the agent will receive rewards for attaining, the agent is guided towards an optimal policy.

RL algorithms can be classified into two categories, being either model-free or model-based. Aimed at increasing sampling-efficiency, the model-based algorithms are utilizing models of the transition dynamics to predict the effect of actions. However, as they often are computationally intensive, they do not usually extend to high-dimensional state- or action-spaces. This limitation of model-based RL is one of the reasons why model-free methods are used more frequently in practice [19]. C-PI^2 [20], an extension of PI^2 [21], is a state of the art model-free method, granting high learning speeds for VIC. Despite being sampling efficient in the context of model-free methods, it still requires more than 100 rollouts to achieve good performance [19].

When performing interaction control, doing a lot of trials in training brings with it adverse consequences, such as high wear and tear of the robot [11]. This motivates the use of model-based RL, having a higher potential of sample efficiency. Today, PILCO [12] is considered the state of the art approach for achieving high sample efficiency [11]. The great efficiency is achieved by optimizing the policy based on internal simulation, enabled by the underlying model. To reduce model-bias, one of the key problems of model-based RL, it is using Gaussian Processes (GPs) to learn stochastic transition dynamics. As PILCO is quite computationally extensive and does not scale to high dimensional state- or action-spaces, it is beneficial to combine it with conventional interaction control. This way the learning can be reserved to optimize a few decisive parameters, such as the impedance characteristics. This drastically improves sampling efficiency and safety while learning. Li et al. did this in their development of a data-efficient RL method [19] using policy search and probabilistic GP models of the system, similar to PILCO.

Contribution

The high-level aim for this thesis is development of compliant learning-based controllers that are able to perform well in challenging scenarios. More specifically, linked to the above-mentioned use case of robotic ultrasound examinations, where the manipulator is to perform a motion trajectory while maintaining a desired contact force. In the attempt to achieve this goal, a selection of force controllers are implemented and assessed in the sampling efficient PILCO framework. The thesis constitutes four contributions to the field of robotic interaction control:

- Literature review
- Implementing and testing three fundamental approaches within force control:
    1. Admittance Control
    2. Hybrid Force/Motion Control
    3. Force-based Variable Impedance Control
- Providing compatibility between the robot environment and modern machine learning libraries
- Analysis of the respective controllers' performance in the learning-based framework

Outline

The thesis is divided into a total of seven sections, this being the first one, serving as an introduction to the task at hand. Next section is providing a more in-depth coverage of the field of robotic interaction control and the fundamental principles it is based upon. Then the hardware and software essential to the presented work will be reviewed. Accordingly the implementations are presented, covering force controllers and the interface connecting the PILCO-algorithm with the robot-environment. Finally, the results are presented and discussed, leading up to a conclusion in the final section.

2 Background

In this section, the reader is presented with the theoretical foundation which the thesis is built upon. First, the methods used to represent orientation are reviewed. Then the focus is directed at the field of *Robotic Interaction Control*. This is followed by some important concepts in force control, along with the review of four different force controllers. The last subsections are dedicated to the field of learning-based interaction control, with emphasis on *model-based Reinforcement Learning* and the PILCO-algorithm.

2.1 Orientation in 3D Space

In this report, three different representations of orientation and rotation are treated.
Rotation matrices, *Euler angles* and *quaternions* respectively.

Rotation matrices

A rotation matrix, describing a three dimensional rotation, transforming the frame from orientation a to orientation b is denoted $\boldsymbol{R}_a^b \in \mathbb{R}^{3\times3}$. Being an orthogonal matrix per definition, the reverse rotation $\boldsymbol{R}_b^a$ can be found as $\boldsymbol{R}_a^{b^{-1}} = \boldsymbol{R}_a^{b^T}$. The rotation matrix operates as a transformation matrix, e.g., with the ability to represent positions and velocities with respect to different frames, Σ. In particular, in this thesis it is used to transform forces, positions, velocities, accelerations and rotations to the appropriate frame, whether it may be in the base frame or the end-effector frame.

Euler angles

Euler angles, $\begin{pmatrix}\alpha & \beta & \gamma\end{pmatrix}^T \in \mathbb{R}^{3\times1}$, are offering a more minimal form of representation, describing three simple rotations relative to the axes of its own moving coordinate frame Σ. When operating with Euler angles, the order of the rotations matter, and there are several conventions. Among the 12 possible orders of rotations Z-Y-X, Z-Y-Z and Z-X-Z are most commonly used [22].

Quaternions

A drawback of the Euler angle-based representation is occurrence of singularities in the vector notation. The *quaternion* representation is extremely useful as it avoids this problem. A quaternion has the form

$$\boldsymbol{\epsilon} = \eta + \epsilon_1 i + \epsilon_2 j + \epsilon_3 k\,. \tag{1}$$

η, ϵ_1, ϵ_2 and ϵ_3 being scalars, and i, j and k being operators satisfying the following combinatory rules

$$ii = jj = kk = -1, \qquad ij = k,\ jk = i,\ ki = j, \qquad ji = -k,\ kj = -i,\ ik = -j. \tag{2}$$

The *conjugate* of the quaternion is defined as

$$\tilde{\boldsymbol{\epsilon}} = \eta - \epsilon_1 i - \epsilon_2 j - \epsilon_3 k \,, \tag{3}$$

and is useful for defining the *unit quaternion*, satisfying $\tilde{\boldsymbol{\epsilon}}\boldsymbol{\epsilon} = \eta^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 = 1$. Often η and $\begin{pmatrix}\epsilon_1 & \epsilon_2 & \epsilon_3\end{pmatrix}^T$ is referred to as the scalar- and vector part of the quaternion respectively.

2.2 Robot Manipulator Dynamics

Jacobian

The jacobian, $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$, is a mapping between joint space and task space based on the forward kinematics of the manipulator. Accordingly, it is used to calculate the cartesian (task space/operational space) velocity as a function of joint velocities

$$\boldsymbol{v}_e = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{4}$$

and reversely the actuator torque $\boldsymbol{\tau}$ as a function of the task space wrench $\boldsymbol{h}_c$ from the controller

$$\boldsymbol{\tau} = \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{h}_c. \tag{5}$$

Task space formulation of the dynamic model

For the task at hand, it is most convenient to consider the task space formulation of the dynamical system. For a rigid 6-DOF robot manipulator in contact with the environment, the model can be described by

$$\boldsymbol{\Lambda}(\boldsymbol{q})\dot{\boldsymbol{v}}_e + \boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{v}_e + \boldsymbol{\eta}(\boldsymbol{q}) = \boldsymbol{h}_c - \boldsymbol{h}_e, \tag{6}$$

where $\boldsymbol{h}_c$ is the controller's force output, $\boldsymbol{h}_e$ is the external wrench, $\boldsymbol{\Lambda}(\boldsymbol{q}) \in \mathbb{R}^{6\times 6}$ is the cartesian inertia matrix, $\boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \mathbb{R}^{6\times 6}$ is the wrench caused by centrifugal and Coriolis effects, and $\boldsymbol{\eta}(\boldsymbol{q}) \in \mathbb{R}^{6\times 1}$ is the wrench of the gravitational effects. The cartesian inertia matrix, $\boldsymbol{\Lambda}(\boldsymbol{q})$, is calculated as

$$\boldsymbol{\Lambda}(\boldsymbol{q}) = (\boldsymbol{J}\boldsymbol{H}(\boldsymbol{q})^{-1}\boldsymbol{J}^T)^{-1}, \tag{7}$$

where $\boldsymbol{H}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the symmetric and positive-definite joint space inertia matrix. This inertia matrix is representing the mass distribution of the manipulator and is highly state-dependent. By additionally knowing the joint space formulation of the centrifugal and Coriolis effects, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$, the corresponding wrench, $\boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})$, is

$$\boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{J}^{-T}\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{J}^{-1} - \boldsymbol{\Lambda}(\boldsymbol{q})\dot{\boldsymbol{J}}\boldsymbol{J}^{-1}. \tag{8}$$

The last component on the left side of (6), the wrench of the gravitational effects, is found as $\boldsymbol{\eta}(\boldsymbol{q}) = \boldsymbol{J}^{-T}\boldsymbol{g}(\boldsymbol{q})$, where $\boldsymbol{g}(\boldsymbol{q})$ is the joint space quantity.

2.3 Robotic Interaction Control

Robotic interaction control is the concept of controlling the interaction between a manipulator and its environment. Combined with methods of motion control, manipulators can be made capable of following desired trajectories while ensuring a *compliant behavior* with respect to external forces, providing safe and stable control. Furthermore, interaction control make it possible to perform advanced manipulation tasks, requiring adaptability that is enabled through force feedback. The methods to achieve interaction control is divided into two broad approaches, *passive*- and *active interaction control* (Fig. 1). In the passive approach, the compliant behavior is inbuilt into the agent's hardware. For example, by structural compliance of the joints, links, position servo or end-effector. A big drawback with this method is its lack of flexibility as the compliant structure must be adapted to fit each special use case. Moreover, it can only handle small deviations from its pre-programmed trajectory [9]. Active interaction control is a more sophisticated approach where the control system is responsible for securing compliance. Although usually being slower and more expensive than the passive one, it is capable of overcoming the previously mentioned drawbacks of passive interaction control [9]. In this thesis we specifically focus on active interaction control.

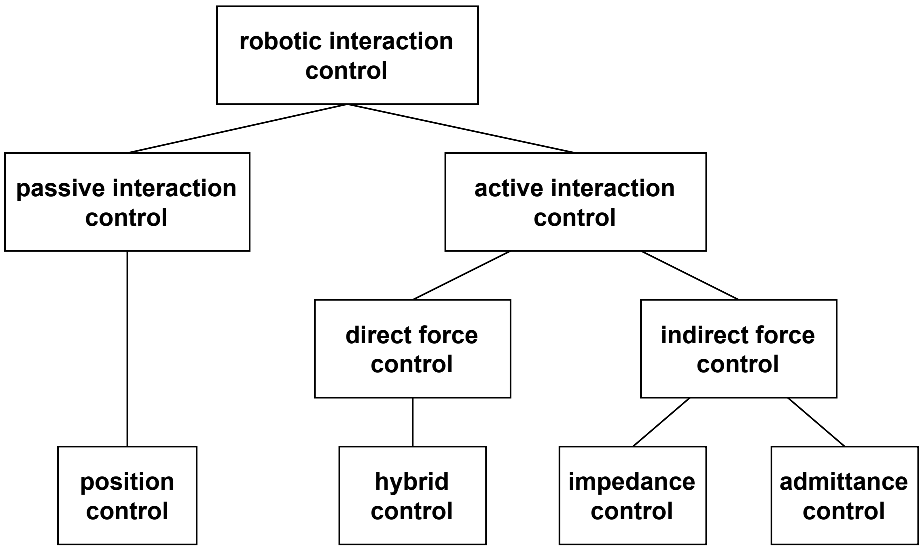


Figure 1: Interaction control classification as proposed in [9]

Active interaction control can be divided into two subcategories, *indirect*- and *direct force control.* The latter is recognized by the use of force feedback in closed-loop. Opposed to the indirect method, it is by design achieving force regulation, controlling the contact force and moment to a desired value. The control method *Hybrid Force/Motion Control* falls under this category. Without any explicit closed force feedback loop, the indirect methods are instead achieving force control through motion control [9] e.g., by changing the reference position to comply with the interaction force. *Impedance Control* and *Admittance Control*, both covered in this section, are examples of approaches within indirect force control.

2.4 General Concepts of Force Control

Stiffness control

Starting off with the basics, the position and orientation of the end-effector (*pose*) can be described by the vector $\boldsymbol{x}_e = \begin{pmatrix}\boldsymbol{p}_e^T & \boldsymbol{\phi}_e^T\end{pmatrix}^T \in \mathbb{R}^{6\times 1}$, where $\boldsymbol{p}_e$ is the position and $\boldsymbol{\phi}_e$ is a set of Euler angles describing the orientation. Similarly, $\boldsymbol{x}_d$ is defined as the desired pose. Assuming a constant $\boldsymbol{x}_d$, the end-effector's deviation from the desired pose is denoted $\Delta\boldsymbol{x}_{de} = \boldsymbol{x}_d - \boldsymbol{x}_e$, and the velocity error is $\Delta\dot{\boldsymbol{x}}_{de} = -\dot{\boldsymbol{x}}_e = -\boldsymbol{A}^{-1}(\boldsymbol{\phi}_e)\boldsymbol{v}_e$, with

$$\boldsymbol{A}(\boldsymbol{\phi}_e) = \begin{pmatrix}\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T}(\boldsymbol{\phi}_e)\end{pmatrix}, \tag{9}$$

$\boldsymbol{I} \in \mathbb{R}^{3\times 3}$ being the identity matrix, $\boldsymbol{0} \in \mathbb{R}^{3\times 3}$ being the null matrix and $\boldsymbol{T} \in \mathbb{R}^{3\times 3}$ being the mapping function of $\boldsymbol{\omega}_e = \boldsymbol{T}(\boldsymbol{\phi}_e)\dot{\boldsymbol{\phi}}_e$, where $\boldsymbol{\omega}_e$ is the angular velocity of the end-effector. If we now introduce the following proportional-derivative motion control law with gravity-compensation

$$\boldsymbol{h}_c = \boldsymbol{A}^{-T}(\boldsymbol{\phi}_e)\boldsymbol{K}_P\Delta\boldsymbol{x}_{de} - \boldsymbol{K}_D\boldsymbol{v}_e + \boldsymbol{\eta}(\boldsymbol{q}), \tag{10}$$

where $\boldsymbol{K}_P \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{K}_D \in \mathbb{R}^{6\times 6}$ are symmetric positive-definite matrices, we find the following asymptotically stable equilibrium

$$\boldsymbol{h}_e = \boldsymbol{A}^{-T}(\boldsymbol{\phi}_e)\boldsymbol{K}_P\Delta\boldsymbol{x}_{de}. \tag{11}$$

According to (11), the end-effector will behave as a 6-DOF spring in respect of the external wrench $\boldsymbol{h}_e$. Furthermore, (11) shows that the matrix $\boldsymbol{K}_P$ is playing the role of an *active stiffness*, controlling the compliance of the manipulator. The choice of $\boldsymbol{K}_P$ is then important for ensuring a suitable elastic behaviour. For example, you can design this stiffness matrix so that you have high positional accuracy in the xy-plane, and allowing more compliance in the z-direction, reducing interaction-forces along this axis. Establishing a second-order dynamical relationship between the deviation of the end-effector pose and the external wrench is what is defined as stiffness control [9].

Mechanical springs

The compliant behavior of two elastically coupled rigid bodies, A and B, with coinciding reference frames Σ_a and Σ_b, can near the equilibrium be described by the linear mapping

$$\boldsymbol{h}_b^b = \boldsymbol{K}\delta\boldsymbol{x}_{ab}^b = \begin{pmatrix}\boldsymbol{K}_t & \boldsymbol{K}_c \\ \boldsymbol{K}_c^T & \boldsymbol{K}_o\end{pmatrix}\delta\boldsymbol{x}_{ab}^b, \tag{12}$$

where $\boldsymbol{h}_b^b$ is the elastic wrench applied to body B, expressed in Σ_b, while being exposed to an infinitesimal twist displacement $\delta \boldsymbol{x}_{ab}^b$, also expressed in Σ_b [9]. $\boldsymbol{K} \in \mathbb{R}^{6\times 6}$ in (12) is the symmetric positive-semidefinite stiffness matrix, describing the behavior of an ideal 6-DOF spring. It consists of the *translational stiffness*, $\boldsymbol{K}_t \in \mathbb{R}^{3\times 3}$, the *rotational stiffness*, $\boldsymbol{K}_o \in \mathbb{R}^{3\times 3}$, and the *coupling stiffness*, $\boldsymbol{K}_c \in \mathbb{R}^{3\times 3}$. In the case of a symmetric coupling stiffness matrix, there is a maximum decoupling between rotation and translation. Accordingly, there is a *center of stiffness* at the point where the bodies' reference frames coincide. Similarly a *center of compliance* can be defined if the compliance matrix, $\boldsymbol{C} = \boldsymbol{K}^{-1}$, has symmetric off-diagonal blocks. If the center of stiffness and the center of compliance coincide, there is no coupling between translation and rotation. Meaning that a relative translation of the bodies results in a pure force along an axis, through the center off stiffness. And also meaning that a relative rotation would lead to a pure torque about an axis through the center of stiffness.

Geometrically consistent active stiffness

To realize a geometrically consistent 6-DOF active stiffness, it is required to find an appropriate control law with the correct proportional control action. In the case of having a finite displacement of the end-effector frame Σ_e in respect to the desired frame Σ_d, the resulting control action can be interpreted as the elastic wrench applied on the end-effector. As to guarantee asymptotic stability in terms of Lyapunov, there is need for a fitting potential elastic energy function. The expression for mechanical stiffness in (12) is simplified by assuming that the coupling stiffness matrix, $\boldsymbol{K}_c$, is zero. The simplification means that the potential elastic energy can be expressed as the sum of translational- and rotational potential energy. The translational potential energy is defined as

$$V_t = \frac{1}{2}\Delta \boldsymbol{p}_{de}^T \boldsymbol{K}'_{Pt} \Delta \boldsymbol{p}_{de}, \tag{13}$$

with

$$\boldsymbol{K}'_{Pt} = \frac{1}{2}\boldsymbol{R}_d \boldsymbol{K}_{Pt} \boldsymbol{R}_d^T + \frac{1}{2}\boldsymbol{R}_e \boldsymbol{K}_{Pt} \boldsymbol{R}_e^T, \tag{14}$$

where $\boldsymbol{K}_{Pt} \in \mathbb{R}^{3\times 3}$ is a symmetric positive-definite matrix. By using $\boldsymbol{K}'_{Pt}$ instead of $\boldsymbol{K}_{Pt}$ in (13), the potential energy is guaranteed to be port symmetric, also in the case of finite displacements [9]. The following power $\dot{V}_t$ becomes

$$\dot{V}_t = \Delta \dot{\boldsymbol{p}}_{de}^{e^T} \boldsymbol{f}_{\Delta t}^e + \Delta \boldsymbol{\omega}_{de}^{e^T} \boldsymbol{m}_{\Delta t}^e, \tag{15}$$

where $\Delta \dot{\boldsymbol{p}}_{de}^e$ is the time derivative of the positional deviation, and $\Delta \boldsymbol{\omega}_{de}^e$ is the error in angular velocity, both with respect to the end-effector frame. The vectors $\boldsymbol{f}_{\Delta t}^e$ and $\boldsymbol{m}_{\Delta t}^e$ are the the elastic force and moment respectively, applied to the end-effector at a finite position displacement $\Delta \boldsymbol{p}_{de}^e$. Expressed in the base frame they are computed as

$$\boldsymbol{f}_{\Delta t} = \boldsymbol{K}'_{Pt} \Delta \boldsymbol{p}_{de}, \qquad \boldsymbol{m}_{\Delta t} = \boldsymbol{K}''_{Pt} \Delta \boldsymbol{p}_{de}, \tag{16}$$

with

$$\boldsymbol{K}''_{Pt} = \frac{1}{2}\boldsymbol{S}(\Delta\boldsymbol{p}_{de})\boldsymbol{R}_d\boldsymbol{K}_{Pt}\boldsymbol{R}_d^T, \tag{17}$$

where $\boldsymbol{S}(\Delta\boldsymbol{p}_{de}) \in \mathbb{R}^{3\times3}$ is the *skew matrix* of the positional displacement vector $\Delta\boldsymbol{p}_{de} \in \mathbb{R}^{3\times1}$

$$\boldsymbol{S}(\boldsymbol{v}) = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}. \tag{18}$$

In sum, the elastic wrench caused by a pure translational displacement is denoted

$$\boldsymbol{h}_{\Delta t} = \begin{pmatrix}\boldsymbol{f}_{\Delta t}^T & \boldsymbol{m}_{\Delta t}^T\end{pmatrix}^T. \tag{19}$$

The rotational potential energy in turn can be defined as

$$V_o = 2\boldsymbol{\epsilon}_{d_e}^{eT}\boldsymbol{K}_{Po}\boldsymbol{\epsilon}_{de}^e, \tag{20}$$

where $\boldsymbol{\epsilon}_{de}^e \in \mathbb{R}^{3\times1}$ is the vector part of the unit quaternion extracted from the rotation $\boldsymbol{R}_d^e$, taking the end-effector to the desired orientation. Similar to $\boldsymbol{K}_{Pt}$ in (17), $\boldsymbol{K}_{Po}$ is a symmetric positive-definite matrix $\in \mathbb{R}^{3\times3}$. Since $\boldsymbol{\epsilon}_{d_e}^e = -\boldsymbol{\epsilon}_{d_e}^e$, the function V_o is port symmetric. $\dot{V}_o$ yields

$$\dot{V}_o = \Delta\boldsymbol{\omega}_{de}^{eT}\boldsymbol{m}_{\Delta o}^e, \tag{21}$$

with

$$\boldsymbol{m}_{\Delta o} = \boldsymbol{K}'_{Po}\boldsymbol{\epsilon}_{de}, \tag{22}$$

where

$$\boldsymbol{K}'_{Po} = 2\boldsymbol{E}^T(\boldsymbol{\eta}_{de}, \boldsymbol{\epsilon}_{de})\boldsymbol{R}_e\boldsymbol{K}_{Po}\boldsymbol{R}_e^T, \tag{23}$$

and $\boldsymbol{E}(\boldsymbol{\eta}_{de}, \boldsymbol{\epsilon}_{de}) = \boldsymbol{\eta}_{de}\boldsymbol{I} - \boldsymbol{S}(\boldsymbol{\epsilon}_{de})$. Accordingly a finite orientation displacement $\boldsymbol{\epsilon}_{de}$ produces an elastic wrench

$$\boldsymbol{h}_{\Delta o} = \begin{pmatrix}\boldsymbol{0}^T & \boldsymbol{m}_{\Delta o}^T\end{pmatrix}^T, \tag{24}$$

equivalent to a pure moment. The total elastic wrench can then be expressed as

$$\boldsymbol{h}_{\Delta} = \boldsymbol{h}_{\Delta t} + \boldsymbol{h}_{\Delta o}, \tag{25}$$

in line with (19) and (24), as a function of both a finite position- and orientation displacement. Discarding the high order infinitesimal terms, (25) yields the mapping

$$\boldsymbol{h}_e^e = \boldsymbol{K}_P \delta \boldsymbol{x}_{de}^e = \begin{pmatrix} \boldsymbol{K}_{Pt} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{K}_{Po} \end{pmatrix} \delta \boldsymbol{x}_{de}^e. \tag{26}$$

This shows that $\boldsymbol{K}_P$ represents an ideal spring with respect to the frame of the end-effector, the origin being the center of stiffness. Also in the case of large displacements, the geometrical and physical meaning of $\boldsymbol{K}_{Pt}$ and $\boldsymbol{K}_{Po}$ remains the same. Meaning e.g., that the upper left element in $\boldsymbol{K}_{Pt} \in \mathbb{R}^{3\times 3}$ will always decide the stiffness in x-direction with respect to Σ_e.

2.5 Indirect Force Controllers

Impedance Control

In order to achieve a desired *dynamic* behavior, stiffness control is not sufficient. The problem extends to achieving a desired second-order system with 6-DOF, characterized by a certain mass, damping and stiffness. Achieving such a system, known as mechanical *impedance*, can be tedious as the dynamics depends on the nonlinear and coupled ones of the manipulator [9]. By doing the acceleration-resolved approach associated with motion control, one aim to decouple and linearize the nonlinear robot dynamics at the acceleration level. Furthermore, in the presence of a force and torque sensor (FT sensor) measuring $\boldsymbol{h}_e$, a more complete form of impedance control can be implemented by enabling inertia shaping, meaning that you are able to determine the apparent inertia of the closed system. By casting the control law

$$\boldsymbol{h}_c = \boldsymbol{\Lambda}(\boldsymbol{q})\boldsymbol{\alpha} + \boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\eta}(\boldsymbol{q}) + \boldsymbol{h}_e, \tag{27}$$

into the dynamic model in (6), it reduces to $\dot{\boldsymbol{v}}_e = \boldsymbol{\alpha}$, $\boldsymbol{\alpha}$ being the control input with the meaning of an acceleration with respect to the base frame. Identifying $\dot{\boldsymbol{v}}_e = \bar{\boldsymbol{R}}_e^T \dot{\boldsymbol{v}}_e^e + \dot{\bar{\boldsymbol{R}}}_e^T \boldsymbol{v}_e^e$ with

$$\bar{\boldsymbol{R}}_e = \begin{pmatrix} \boldsymbol{R}_e & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_e \end{pmatrix}, \tag{28}$$

choosing

$$\boldsymbol{\alpha} = \bar{\boldsymbol{R}}_e^T \boldsymbol{\alpha}^e + \dot{\bar{\boldsymbol{R}}}_e^T \boldsymbol{v}_e^e, \tag{29}$$

leads to $\dot{\boldsymbol{v}}_e^e = \boldsymbol{\alpha}^e$, with the control input $\boldsymbol{\alpha}^e$ having the meaning of an acceleration relative to the end-effector frame. Now, setting

$$\boldsymbol{\alpha}^e = \dot{\boldsymbol{v}}_d^e + \boldsymbol{K}_M^{-1}(\boldsymbol{K}_D \Delta \boldsymbol{v}_{de}^e + \boldsymbol{h}_\Delta^e - \boldsymbol{h}_e^e), \tag{30}$$

the closed loop expression is found to be

$$\boldsymbol{K}_M \Delta \dot{\boldsymbol{v}}^e_{de} + \boldsymbol{K}_D \Delta \boldsymbol{v}^e_{de} + \boldsymbol{h}^e_\Delta = \boldsymbol{h}^e_e, \tag{31}$$

$\boldsymbol{K}_M \in \mathbb{R}^{3\times3}$ and $\boldsymbol{K}_D \in \mathbb{R}^{3\times3}$ being symmetric positive-definite matrices, $\Delta\dot{\boldsymbol{v}}^e_{de}$ and $\Delta\boldsymbol{v}^e_{de}$ being the error in acceleration and velocity, and $\boldsymbol{h}^e_\Delta$ being the elastic wrench defined in (25), all relative to Σ_e. $\boldsymbol{K}_M$, having the meaning of the apparent inertia matrix, is like $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ an adjustable parameter. With no external wrench working on the manipulator, this control scheme guarantees that that the end-effector frame Σ_e asymptotically follows the desired frame Σ_d. In the presence of external forces, the compliant behavior of the end-effector is described by (31), limiting the contact wrench at the expense of a finite displacement in position and orientation.

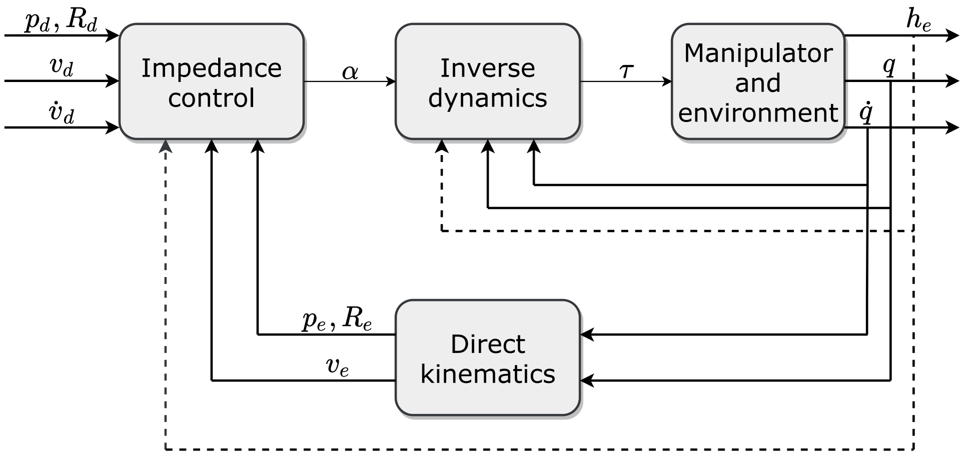


Figure 2: High level block diagram of the suggested impedance controller

Admittance Control

Whereas an Impedance Controller receives motion as input and imposes an effort as output, the opposite is the case for the Admittance Controller [23]. Principally, force-readings are used to calculate appropriate responses from a position controller. As there is no need for robotic torque control, the control method is applicable to traditional industrial robots using position control.

For the Admittance Controller, the relationship between a one-dimensional interaction force F_{int} and the corresponding motion-displacement $(\ddot{x}, \dot{x}, x)$ is described by the following second order system

$$M\ddot{x}(t) + B\dot{x}(t) + Kx(t) = F_{int}(t), \tag{32}$$

where M, B and $K \in \mathbb{R}$ is the system's inertia, damping and stiffness, respectively. As computers works at discrete time, a transformation to discrete domain via the Laplace domain is presented. With null initial conditions, the resulting Laplace transform is given by

$$X(s)(Ms^2 + Bs + K) = F_{int}(s). \tag{33}$$

As admittance deals with force as input and motion as output, the transfer function is denoted

$$\frac{X}{F}(s) = \frac{1}{Ms^2 + Bs + K}. \tag{34}$$

Using Tustin's approximation (35), with T_s being the controller's time step, the equation can be transformed from Laplace- to Z-domain.

$$s \approx \frac{2}{T_s}\frac{z-1}{z+1} \tag{35}$$

The method results in the following equation

$$\begin{aligned} x(k) =& [T_s^2 f(k) + 2T_s^2 f(k-1) + T_s^2(k-2) - (2KT_s^2 - 8M)x(k-1) \\ &- (4M - 2BT_s + KT_s^2)x(k-2)] * 1/(4M + 2BT_s + KT_s^2), \end{aligned} \tag{36}$$

where $f(k)$, $f(k-1)$ and $f(k-2)$ are the most recent interaction forces, $x(k) \in \mathbb{R}$ is the adjustment to the reference position $X_d \in \mathbb{R}^{3\times1}$, and $x(k-1)$ and $x(k-2)$ are the previous adjustments. The full deduction is presented in the appendix of [23]. The adjustment of the reference position, $x(k)$, or ΔX_d if you will, is added to the appropriate index of $\boldsymbol{X}_d \in \mathbb{R}^{3\times1}$ and results in a compliant position X_c in z. The full admittance controller is illustrated in Fig. 3.

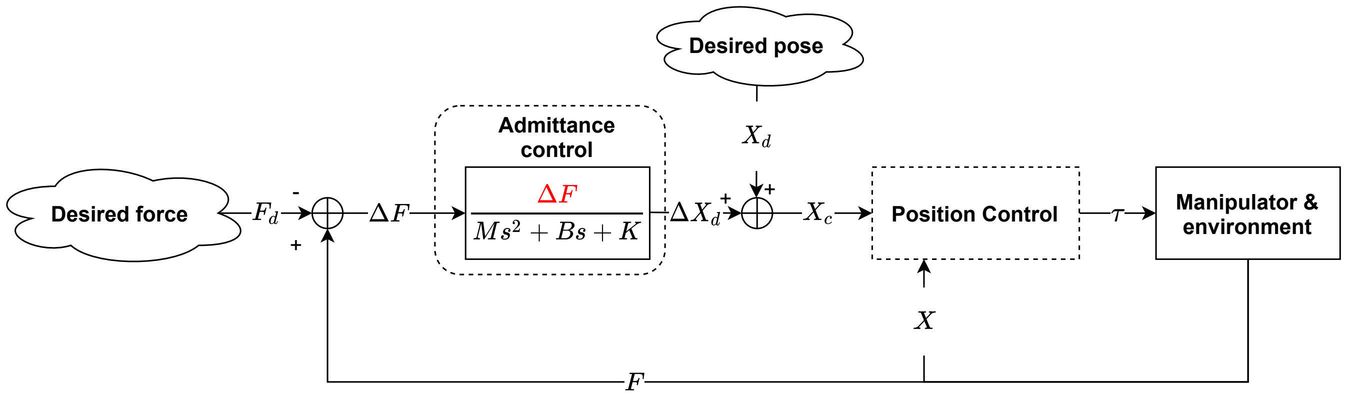


Figure 3: Block diagram of the Admittance Controller

2.6 Direct Force Controllers

Hybrid Force/Motion Control

Unlike Impedance- and Admittance Control, the Hybrid Force/Motion Controller (HFMC or Hybrid Control) is performing direct force control (Fig. 1). The aim of this control method is to achieve both motion and force control by dividing the task into two separate, decoupled subproblems [9]. By specifying which subspaces should be controlled by a motion- and force controller respectively, the Hybrid Control intend to simultaneously solve the two separate control tasks. The matrices $\boldsymbol{S}_v$ and $\boldsymbol{S}_f$ are used to specify these subspaces. In the case of doing force control along the z-axis, and motion control in the remaining five dimensions, $\boldsymbol{S}_v$ and $\boldsymbol{S}_f$ are defined as

$$\boldsymbol{S}_f = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{S}_v = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{37}$$

When dealing with a compliant environment, the end-effector displacement caused by environmental deformation in the presence of a wrench $\boldsymbol{h}_s$, can be modeled as

$$\delta \boldsymbol{x} = \boldsymbol{C}\boldsymbol{h}_s, \tag{38}$$

where $\boldsymbol{C} = \boldsymbol{K}^{-1} \in \mathbb{R}^{6\times 6}$ is an ideal 6-DOF spring of compliance [9]. This displacement can be decomposed as

$$\delta \boldsymbol{x} = \delta \boldsymbol{x}_v + \delta \boldsymbol{x}_f, \tag{39}$$

where $\delta \boldsymbol{x}_v$ and $\delta \boldsymbol{x}_f$ are the twist displacements in each of the respective controllers' subspaces. Similarly, the end-effector velocity can be decomposed as

$$\boldsymbol{v}_e = \boldsymbol{S}_v \boldsymbol{v} + \boldsymbol{C}' \boldsymbol{S}_f \dot{\boldsymbol{\lambda}}, \tag{40}$$

where $\boldsymbol{\lambda}$ is the force multiplier, and $\boldsymbol{C}' = (\boldsymbol{I} - \boldsymbol{P}_v)\boldsymbol{C}$, with $\boldsymbol{P}_v$ being a projection matrix that filters out all the end-effector twists that are not in the range space of $\boldsymbol{S}_v$ [9]. $\boldsymbol{I} - \boldsymbol{P}_v$ thus has the opposite effect of filtering out the twists that *are* in the range space of $\boldsymbol{S}_v$. $\boldsymbol{P}_v$ itself is defined as $\boldsymbol{P}_v = \boldsymbol{S}_v \boldsymbol{S}_v^\dagger$, where $\boldsymbol{S}_v^\dagger$ is a suitable weighted pseudoinverse of $\boldsymbol{S}_v$,

$$\boldsymbol{S}_v^\dagger = (\boldsymbol{S}_v^T \boldsymbol{W} \boldsymbol{S}_v)^{-1} \boldsymbol{S}_v^T \boldsymbol{W}. \tag{41}$$

Setting $\boldsymbol{W}$ equal to the inertia matrix $\boldsymbol{M} \in \mathbb{R}^{6\times 6}$ corresponds to defining a norm in the space of twists based on the kinetic energy [9]. Assuming the contact geometry $\boldsymbol{S}_v$ and compliance $\boldsymbol{C}'$ to be constant, (40) leads to the following decomposition of acceleration

$$\dot{\boldsymbol{v}}_e = \boldsymbol{S}_v \dot{\boldsymbol{v}} + \boldsymbol{C}' \boldsymbol{S}_f \ddot{\boldsymbol{\lambda}}. \tag{42}$$

The inverse-dynamics control law in (27):

$$\boldsymbol{h}_c = \boldsymbol{\Lambda}(\boldsymbol{q})\boldsymbol{\alpha} + \boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\eta}(\boldsymbol{q}) + \boldsymbol{h}_e,$$

used in the impedance controller, can be adopted, again leading to the closed loop expression $\dot{\boldsymbol{v}}_e = \boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is a properly designed control input [9]. The choice

$$\boldsymbol{\alpha} = \boldsymbol{S}_v \boldsymbol{\alpha}_v + \boldsymbol{C}' \boldsymbol{S}_f \boldsymbol{f}_\lambda \tag{43}$$

allows decoupling of the respective controllers, $\boldsymbol{\alpha}_v$ relating to motion control and $\boldsymbol{f}_\lambda$ to force control. By choosing

$$\boldsymbol{\alpha}_v = \ddot{\boldsymbol{r}}_d(t) + \boldsymbol{K}_{Dr}[\dot{\boldsymbol{r}}_d(t) - \boldsymbol{v}(t)] + \boldsymbol{K}_{Pr}[\boldsymbol{r}_d - \boldsymbol{r}(t)], \tag{44}$$

asymptotic tracking of a desired velocity $\boldsymbol{v}_d$ and acceleration $\dot{\boldsymbol{v}}_d$ is guaranteed, with exponential convergence [9]. The choice

$$\boldsymbol{f}_\lambda = \ddot{\boldsymbol{\lambda}}_d(t) + \boldsymbol{K}_{D\lambda}[\dot{\boldsymbol{\lambda}}_d - \dot{\boldsymbol{\lambda}}(t)] + \boldsymbol{K}_{P\lambda}[\boldsymbol{\lambda}_d(t) - \boldsymbol{\lambda}(t)] \tag{45}$$

with positive-definite matrices $\boldsymbol{K}_{D\lambda}$ and $\boldsymbol{K}_{P\lambda}$ ensures asymptotic tracking of a desired force trajectory $(\ddot{\boldsymbol{\lambda}}_d(t), \dot{\boldsymbol{\lambda}}_d(t), \boldsymbol{\lambda}_d(t))$, also with exponential convergence [9].
The quantity $\dot{\boldsymbol{\lambda}}$ in (45) can be computed from the force measurements of the end-effector $\boldsymbol{h}_e$ as

$$\dot{\boldsymbol{\lambda}} = \boldsymbol{S}_f^{\dagger} \dot{\boldsymbol{h}}_e. \tag{46}$$

where $\boldsymbol{S}_f^{\dagger}$ is the pseudoinverse of $\boldsymbol{S}_f$, computed as in (41) with $\boldsymbol{W} = \boldsymbol{C}$. Due to the noisy nature of the force-readings however, the estimate

$$\dot{\boldsymbol{\lambda}} = \boldsymbol{S}_f^{\dagger} \boldsymbol{K}' \boldsymbol{J}(\boldsymbol{q}) \dot{\boldsymbol{q}} \tag{47}$$

is often preferred, where $\boldsymbol{K}' = \boldsymbol{P}_f \boldsymbol{K}$ and $\boldsymbol{P}_f = \boldsymbol{S}_f \boldsymbol{S}_f^{\dagger}$. The full hybrid controller is illustrated in Fig. 4.

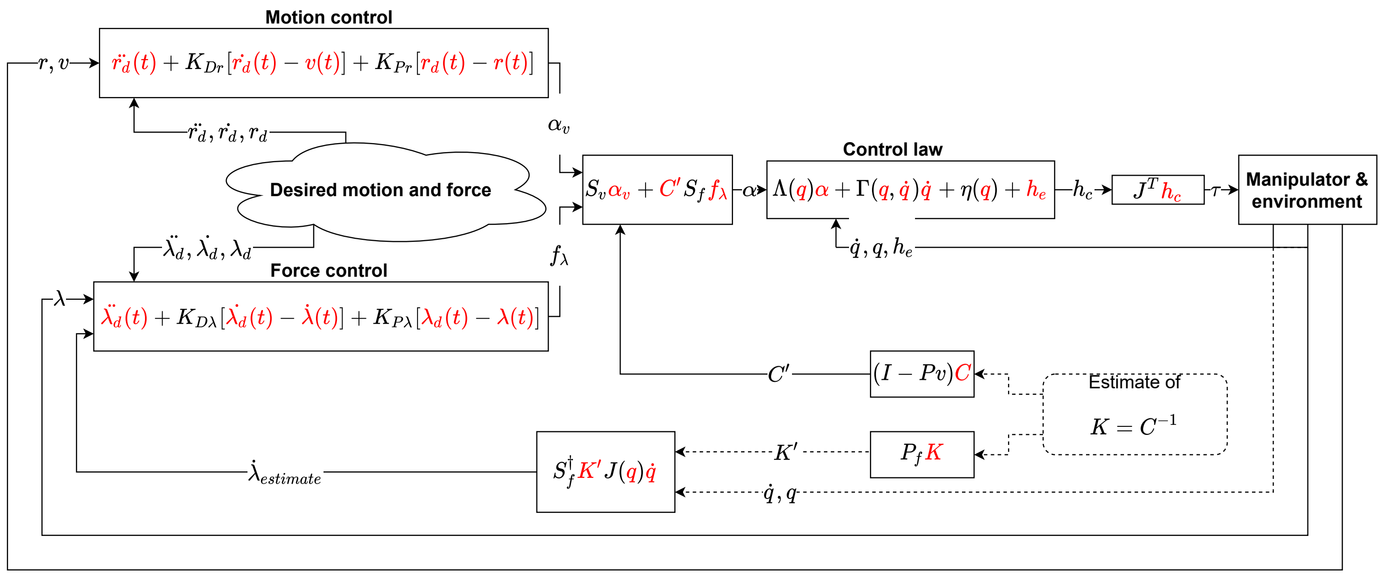

Motion control
r, v
$\ddot{r}_d(t) + K_{Dr}[\dot{r}_d(t) - v(t)] + K_{Pr}[r_d(t) - r(t)]$
$\ddot{r}_d, \dot{r}_d, r_d$
α_v
Desired motion and force
Control law
$S_v\alpha_v + C' S_f f_\lambda$
α
$\Lambda(q)\alpha + \Gamma(q,\dot{q})\dot{q} + \eta(q) + h_e$
h_c
$J^T h_c$
τ
Manipulator & environment
$\dot{q}, q, h_e$
$\ddot{\lambda}_d, \dot{\lambda}_d, \lambda_d$
f_λ
Force control
λ
$\ddot{\lambda}_d(t) + K_{D\lambda}[\dot{\lambda}_d(t) - \dot{\lambda}(t)] + K_{P\lambda}[\lambda_d(t) - \lambda(t)]$
C'
$(I - Pv)C$
Estimate of
$K = C^{-1}$
K'
$P_f K$
$\dot{\lambda}_{estimate}$
$S_f^\dagger K' J(q)\dot{q}$
$\dot{q}, q$

Figure 4: Block diagram of the Hybrid Force/Motion Controller

Force-based Variable Impedance Controller

Opposed to regular impedance control, Force-based Variable Impedance Control (VIC) is designed to achieve force regulation by adjusting the system impedance. Such a controller is presented in [17], introducing adaptation laws for both the inertia $\boldsymbol{M}$, damping $\boldsymbol{B}$ and stiffness $\boldsymbol{K}$. The proposed control method is based on adjusting the system impedance

$$\boldsymbol{M}\ddot{\boldsymbol{E}}_1 + \boldsymbol{B}\dot{\boldsymbol{E}}_1 + \boldsymbol{K}\boldsymbol{E}_1 = -\boldsymbol{F}_{ext}, \tag{48}$$

where $\boldsymbol{E}_1 = \boldsymbol{X} - \boldsymbol{X}_d$ is the error in position. Supposing uncertainties in $\boldsymbol{M}$, $\boldsymbol{B}$ and $\boldsymbol{K}$, the controller is designed as

$$\hat{\boldsymbol{M}}\ddot{\boldsymbol{E}}_1 + \hat{\boldsymbol{B}}\dot{\boldsymbol{E}}_1 + \hat{\boldsymbol{K}}\boldsymbol{E}_1 = -\boldsymbol{F}^* \tag{49}$$

with

$$\boldsymbol{F}^* = \boldsymbol{F}_d + \boldsymbol{K}_v(\dot{\boldsymbol{F}}_d - \dot{\boldsymbol{F}}_{ext}), \tag{50}$$

$\hat{\boldsymbol{M}}$, $\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{K}}$ being the actual, dynamic impedance of the system, $\boldsymbol{F}^*$ being the auxiliary force input and $\boldsymbol{K}_v \in \mathbb{R}^{n\times n}$ being a gain matrix. Equation (50) assumes the interacting force to be continuously differentiable. As a general rule it is at least piecewise continuous [17]. The goal of the controller is to ensure that $\boldsymbol{F}^*$ approaches $\boldsymbol{F}_d$ by the means of adjusting the inertia $\hat{\boldsymbol{M}}$, damping $\hat{\boldsymbol{B}}$ and stiffness $\hat{\boldsymbol{K}}$. To do so, it is dependent on suitable adaptation laws. By defining $\boldsymbol{F}_{ext} - \boldsymbol{F}_d = \boldsymbol{E}_f$ where $\boldsymbol{F}_{ext}$ and $\boldsymbol{F}_d$ are defined as in (48), (49) and (50), we obtain

$$(\hat{\boldsymbol{M}} - \boldsymbol{M})\ddot{\boldsymbol{E}}_1 + (\hat{\boldsymbol{B}} - \boldsymbol{B})\dot{\boldsymbol{E}}_1 + (\hat{\boldsymbol{K}} - \boldsymbol{K})\boldsymbol{E}_1 = \boldsymbol{K}_v\dot{\boldsymbol{E}}_f + \boldsymbol{E}_f \tag{51}$$

Additionally defining

$$\tilde{\boldsymbol{M}} = \hat{\boldsymbol{M}} - \boldsymbol{M}, \qquad \tilde{\boldsymbol{B}} = \hat{\boldsymbol{B}} - \boldsymbol{B}, \qquad \tilde{\boldsymbol{K}} = \hat{\boldsymbol{K}} - \boldsymbol{K}, \tag{52}$$

allow us to rewrite (51) as

$$\dot{\boldsymbol{E}}_f = -\boldsymbol{K}_v^{-1}\boldsymbol{E}_f + \boldsymbol{K}_v^{-1}(\tilde{\boldsymbol{M}}\ddot{\boldsymbol{E}}_1 + \tilde{\boldsymbol{B}}\dot{\boldsymbol{E}}_1 + \tilde{\boldsymbol{K}}\boldsymbol{E}_1). \tag{53}$$

By setting

$$\tilde{\boldsymbol{M}}\ddot{\boldsymbol{E}}_1 + \tilde{\boldsymbol{B}}\dot{\boldsymbol{E}}_1 + \tilde{\boldsymbol{K}}\boldsymbol{E}_1 = \boldsymbol{\xi}\boldsymbol{\lambda} \tag{54}$$

where $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{E}_1, \dot{\boldsymbol{E}}_1, \ddot{\boldsymbol{E}}_1)$ is a $n \times 3n$ matrix and $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\tilde{\boldsymbol{M}}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{K}})$ is a $3n \times 1$ vector, (53) is reduced to

$$\dot{\boldsymbol{E}}_f = -\boldsymbol{K}_v^{-1}\boldsymbol{E}_f + \boldsymbol{K}_v^{-1}\boldsymbol{\xi}\boldsymbol{\lambda}. \tag{55}$$

Now, investigating stability, the Lyapunov function $\boldsymbol{V}$ can be chosen as

$$\boldsymbol{V} = \boldsymbol{E}_f^T\boldsymbol{P}\boldsymbol{E}_f + \boldsymbol{\lambda}^T\boldsymbol{\Gamma}\boldsymbol{\lambda}, \tag{56}$$

where $\boldsymbol{P}$ and $\boldsymbol{\Gamma}$ are symmetric positive definite matrices [17]. The choice generates the following derivative

$$\dot{\boldsymbol{V}} = -\boldsymbol{E}_f^T(\boldsymbol{K}_v^{-T}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{K}_v^{-1})\boldsymbol{E}_f + 2(\dot{\boldsymbol{\lambda}}^T\boldsymbol{\Gamma}\boldsymbol{\lambda} + \boldsymbol{E}_f^T\boldsymbol{P}\boldsymbol{K}_v^{-1}\boldsymbol{\xi}\boldsymbol{\lambda}). \tag{57}$$

By setting $\boldsymbol{P}$, $\boldsymbol{\Gamma}$ and $\boldsymbol{\lambda}$ so that they satisfy both

$$\boldsymbol{K}_v^{-T}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{K}_v^{-1} = \boldsymbol{Q}_0 \tag{58}$$

and

$$\dot{\boldsymbol{\lambda}}^T\boldsymbol{\Gamma}\boldsymbol{\lambda} + \boldsymbol{E}_f^T\boldsymbol{P}\boldsymbol{K}_v^{-1}\boldsymbol{\xi}\boldsymbol{\lambda} = \boldsymbol{0}, \tag{59}$$

where $\boldsymbol{Q}_0$ is a positive definite matrix, then $\dot{\boldsymbol{V}} = -\boldsymbol{E}_f^T\boldsymbol{Q}_0\boldsymbol{E}_f < \boldsymbol{0}$. This proves the closed loop system to be stable. The criterion in (59) produces the following adaptation law for $\hat{\boldsymbol{M}}$, $\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{K}}$

$$\dot{\boldsymbol{\lambda}} = -(\boldsymbol{E}_f^T\boldsymbol{P}\boldsymbol{K}_v^{-1}\boldsymbol{\xi}\boldsymbol{\Gamma}^{-1})^T = -\boldsymbol{\Gamma}^{-1}\boldsymbol{\xi}^T\boldsymbol{K}_v^{-1}\boldsymbol{P}\boldsymbol{E}_f. \tag{60}$$

Since $\boldsymbol{V} > \mathbf{0}$ and $\dot{\boldsymbol{V}} < \mathbf{0}$, (60) will ensure that the force error $\boldsymbol{E}_f$ will approach zero, meaning that $\boldsymbol{F}^* \rightarrow \boldsymbol{F}_d$. However, as the contact force at steady-state is decided by the position error $\boldsymbol{E}_1$ as well as the stiffness $\hat{\boldsymbol{K}}$, the desired position $\boldsymbol{X}_d$ play an important part. If the desired contact force $\boldsymbol{F}_d$ is large and the position error $\boldsymbol{E}_1$ is small, the adaptive law will adjust $\hat{\boldsymbol{M}}$, $\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{K}}$ until $\boldsymbol{F}^* \rightarrow \boldsymbol{F}_d$, potentially reaching values capable of causing instability issues. Hence, upper bounds should be set for $\hat{\boldsymbol{M}}$, $\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{K}}$, avoiding instability at the expense of force tracking ability [17]. A high level block diagram of the Force-based VIC is shown in Fig. 5.

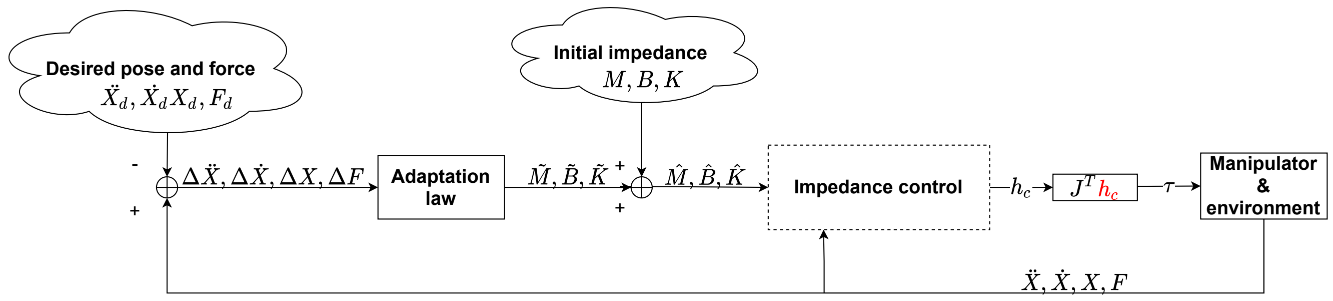

Desired pose and force
$\ddot{X}_d, \dot{X}_d X_d, F_d$
Initial impedance
M, B, K
-
+
$\Delta\ddot{X}, \Delta\dot{X}, \Delta X, \Delta F$
Adaptation law
$\tilde{M}, \tilde{B}, \tilde{K}$
+
+
$\hat{M}, \hat{B}, \hat{K}$
Impedance control
h_c
$J^T h_c$
τ
Manipulator & environment
$\ddot{X}, \dot{X}, X, F$

Figure 5: Block diagram of the Force-based Variable Impedance Controller. The adaptation law is specified in (60)

2.7 Learning-based Interaction Control

In 2020, Abu-Dakka and Saveriano presented a review of existing learning-based approaches in VIC [10]. Furthermore, the authors proposed a new taxonomy for mechanical impedance, based on variability, learning, and control. The learning-based methods for variable impedance were divided into two categories; *Variable Impedance Learning* (VIL) and *Variable Impedance Learning Control* (VILC). The extended taxonomy of the learning-based methods is illustrated in Fig. 6. Special to the VIL-approaches are that they formulate the problem of finding variable impedance gains as a supervised learning problem [10]. By using human demonstrations as training data, the algorithms try to reproduce the exhibited impedance behaviour. To do so, these methods normally depend on existing controllers. VILC-approaches on the other hand, seek to directly learn a variable impedance control law [10].

The approaches classified as VIL are usually presented by the authors as *Imitation Learning* (IL) methods or *Learning from Demonstration* (LfD) methods. In the attempt to pass on human bio-mechanical impedance skills, task-relevant information is extracted from several demonstrations [10]. One approach in LfD is *Kinesthetic Teaching*, where the user demonstrates the desired behaviour by physically moving the robot around [24]. Such frameworks have been presented in [25], [26] and [27], all estimating full stiffness matrices using *Gaussian Mixture Regression* (GMR). Other frameworks, as the one presented in [28], additionally utilizes visual information in the learning process. In this case, the extracted information was used to learn the correct stiffness profile for a cooperative assembly task.

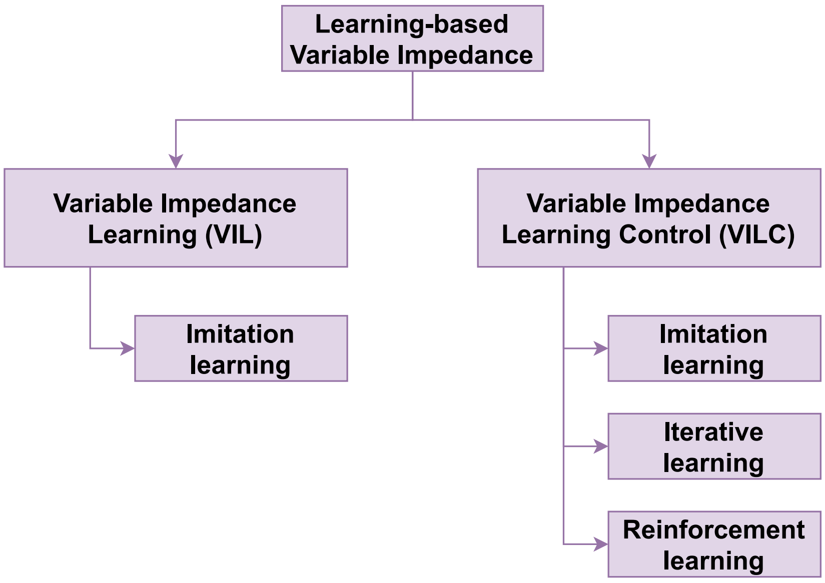

Learning-based Variable Impedance
Variable Impedance Learning (VIL)
Variable Impedance Learning Control (VILC)
Imitation learning
Imitation learning
Iterative learning
Reinforcement learning

Figure 6: A taxonomy of existing approaches for learning-based variable impedance presented in [10]

Still following the taxonomy in Fig. 6, the focus is shifted to the approaches classified as VILC. Whereas the underlying control strategy and the learning algorithm are separated in VIL, there is no clear boundary between them in VILC. A key difference is that for VILC, the process of data collection is dependent on the underlying control [10]. As Fig. 6 shows, there are methods of Imitation Learning that classifies as VILC as well. This is the case for methods with tightly integrated learning and control. An example of this is Yang et al.'s approach for transferring human limb impedance to robots. Their method was to combine haptic feedback with processing of electromyography (EMG) signals collected from human muscles. A limitation of such methods is that they usually require complex setups and tedious calibration.

In the subcategory of *Iterative Learning*, the idea is to adjust the impedance based on experience from past executions. In [29], Bristow et al. presented a framework by the name of *Iterative Learning Control* (ILC), which a lot of methods have relied on since [10]. Kramberger et al.'s approach [30] of achieving trajectory and force tracking in changing environments is one of them. They used an Iterative Learning approach based on monitoring the overall system passivity analysis in terms of reference power tracking. As a means of generating a compact task representation, able to handle sensor-based goal adaptations, they encoded the desired motions into *Dynamic Movement Primitives* (DMPs) [30].

The idea of repeated learning is key in the widely studied topic of Reinforcement Learning (RL) as well. As an introduction to RL was given in Section 1, this subsection is reserved to present specific novel approaches. Now, starting with model-free methods, Buchli et al.'s *Policy Improvement with Path Integrals* (PI^2) [21] is an important one. It realized variable impedance control by learning adjustment strategies for both motion trajectory and impedance gains using DMPs. Using diagonal stiffness matrices and a DMP for each dimension, the proposed algorithm optimized the behaviour in each direction independently. In [20], Winter et al. proposed an extension to this algorithm by the name *Coordination Policy Improvement with Path Integral* (C-PI^2). The algorithm learn variable impedance behaviours considering synergies among DOFs. This way it achieves a better exploitation of the robot's dynamic capabilities. Furthermore, its learning speed outperforms the one of PI^2 for tasks where the coupling of DOF is not negligible [20]. Granting convergence after just above 100 rollouts in several interaction control assessments [19] [20], it is reported as the most efficient model-free RL method.

In terms of novel model-*based* RL methods, there are two recurring characteristics; they are highly sample-efficient, and their models captures uncertainties. These traits are key for reducing model-bias, and wear and tear of the robot. The procedure used to achieve these properties however, varies. Several approaches are using GPs to learn the transition dynamics. Deisenroth and Rasmussen's PILCO algorithm [12] is one of them. Spaandonk made an extension of this algorithm, for learning VIC in his master thesis [31]. Li et al. also used GP's when they learned a probabilistic representation of the interaction dynamics in the context of VIC [19] [32]. In all these cases, the GP models were combined with the efficient *Gradient-Based Policy Search method.* Whereas the method of using GPs is extremely sample-efficient, it does not scale with big datasets [10]. This limitation of GP-based RL has motivated the use of (Artificial) *Neural Networks* (NNs) for learning dynamics models. This is essential components for model-based RL algorithms such as PETS [33] and Deep PILCO [34]. To capture uncertainties, some approaches resort to *Bayesian Neural Networks* (BNNs), and NN-techniques such as *dropout*, preventing overfitting. In [35], Roveda et al. applied NNs to learn models of the state transitions in the use case of VIC. While being better than the GPs in terms of dataset scalability, the GP-based methods comes out on top regarding sampling-efficiency.

Model-based Reinforcement Learning

Compared to other forms of machine learning, RL differs by having an *agent* interacting with its environment as its learning procedure [11]. In each state $\boldsymbol{x}$ the agent applies an action u that results in some state-change. These actions are outputs of the policy function $\pi(\cdot)$ which, in the deterministic case, maps a state to a specific action, $\pi(\boldsymbol{x}) \mapsto u$. To be able to evaluate the performance of the current policy, rewards/costs are collected based on the ability to reach or maintain favourable states. The goal of the RL algorithms is to maximize the expected return by finding the right policy $\pi(\cdot)$. The expected return can be denoted

$$J^{\pi}(\cdot) = \sum_{t=0}^{T} \mathrm{E}_{\mathrm{x}_t}\left[c\left(\mathbf{x}_t\right)\right], \tag{61}$$

where T is the time horizon, and $\mathrm{E}_{\mathrm{x}_t}\left[c\left(\mathbf{x}_t\right)\right]$ is the expected cost c of being in state $\boldsymbol{x}$ at time t.

As it has been mentioned in previous sections, RL algorithms come in two distinct forms, being either model-free (direct RL) or model-based (indirect RL). While the model-free methods solely rely on experience, the model-based ones are utilizing models of the transition dynamics to predict the effect of actions. This way, model-based RL algorithms can forecast rewards and derive optimal actions [11]. The optimality of the actions are however limited by the accuracy of the model, which is going through an optimization process of its own. The pipeline of the model-based RL algorithms is illustrated in Fig. 7.

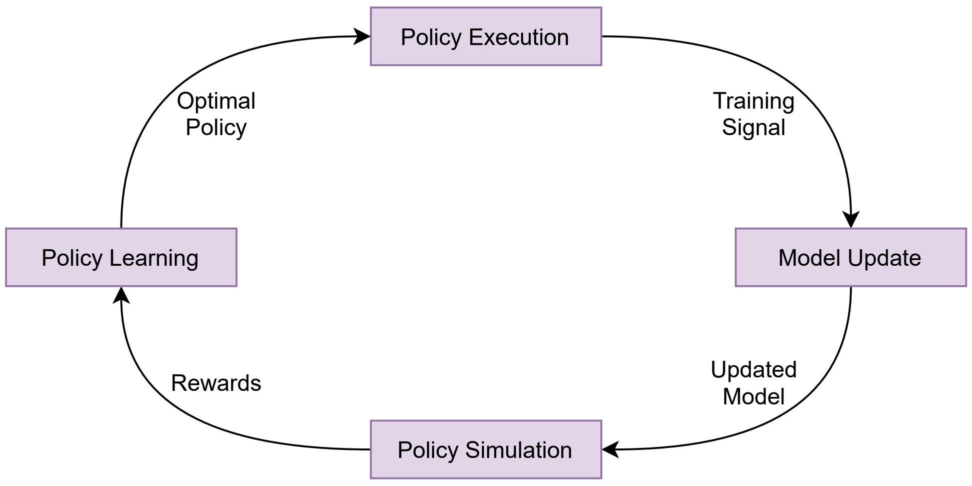


Figure 7: Model-based RL pipeline as presented in [11]

In robotics, the internal simulation of the model significantly reduces the required amount of interactions between the robot and the environment. In general, there is a faster convergence to an optimal solution. The biggest drawback is that the models' accuracy sets an upper limit on performance. If the model is inadequate, so is the policy.

Transition models are distinguished in two main classes: deterministic and stochastic. The predictions of the deterministic models do not depend on random variables, hence they are always predicting the same transition given a state and action: $f(s, a) \mapsto s'$. Stochastic models on the other hand, give predictions defined by a probability distribution over the future states. An overview of popular transition models is given in Fig. 8.
The method of using Gaussian Processes (GP's) is highlighted as it is considered the state of the art approach for sample efficiency [11], and is the one used in our approach. Compared to other stochastic methods, it stands out by building a distribution over functions instead of random variables. Consequently, it is not making any assumptions about the function f. This trait makes GP a powerful learning method. More specifically, a GP is defined by its mean and a kernel (covariance function). This mean is usually assumed to be zero. The parameters making up the kernel can be set through a number of different methods, e.g., *greedy search* and *marginal likelihood* based methods, the latter being more widely used. In the context of transition models, the objective of the GP is to deduce the function f that generates the observations s'. Assuming Gaussian noise, $s' = f(s, a) + N(0, \sigma_n^2 I)$. Using the shorthand notation $\boldsymbol{x} = (s, a)$, the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of f are defined as

$$\boldsymbol{\mu}(\mathbf{x}) = \mathbf{E}[f(\mathbf{x})], \tag{62}$$

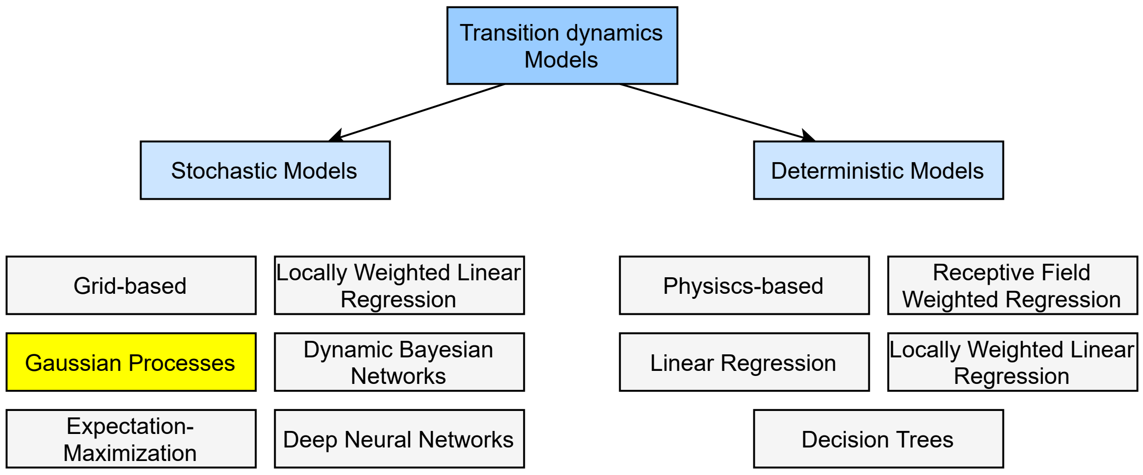

Transition dynamics Models
Stochastic Models
Deterministic Models
Grid-based
Locally Weighted Linear Regression
Gaussian Processes
Dynamic Bayesian Networks
Expectation-Maximization
Deep Neural Networks
Physiscs-based
Receptive Field Weighted Regression
Linear Regression
Locally Weighted Linear Regression
Decision Trees

Figure 8: Overview of transition models as presented in [11]

$$\mathbf{\Sigma}\left(\mathbf{x}, \mathbf{x}^{\star}\right)=\mathbf{E}\left[(f(\mathbf{x})-\boldsymbol{\mu}(\mathbf{x}))\left(f\left(\mathbf{x}^{\star}\right)-\boldsymbol{\mu}\left(\mathbf{x}^{\star}\right)\right)\right], \tag{63}$$

where $\boldsymbol{x}^*$ denotes the testing samples. The most practiced kernel is the *squared exponential*, specified by its variance σ^2 and length-scale λ as in (64). The length-scale decides how far the algorithm extrapolates away from the training data.

$$\mathbf{\Sigma}\left(\mathbf{x}, \mathbf{x}^{\star}\right)=\sigma^2 \exp \left(-\frac{\left\|\mathbf{x}-\mathbf{x}^{\star}\right\|_2^2}{2 \lambda^2}\right) \tag{64}$$

The main approaches within model-based RL are illustrated in Fig. 9.

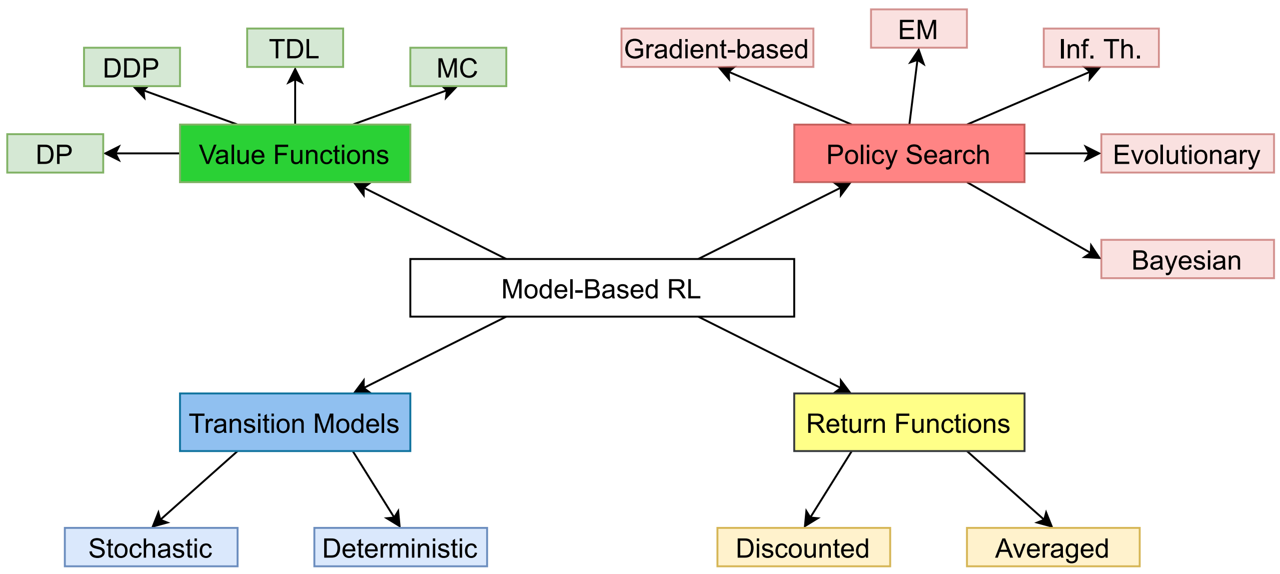


Figure 9: Model-based RL overview as presented in [11]

In terms of policy learning, this thesis is focused on the Gradient-based Policy Search method. In traditional *Value Function* methods, an optimal value function $V^{\pi}(s)$ is estimated in order to obtain the optimal actions in each state. This is not the case for policy search methods. Instead of constructing an optimal policy based on the optimal value function, the optimal policy is learned directly. Compared to value function methods, this trait allows state of the art policy search methods to converge faster in high-DOF robotic systems [11]. In terms of policy representations, there are a wide range of approaches. In this thesis, both a *linear* policy and a *Radial Basis Function* (RBF) policy is assessed. The linear policy model can be denoted

$$\pi\left(\mathbf{x}_*\right) = \mathbf{u}_{\max}\sin\left(\tilde{\pi}\left(\mathbf{x}_*\right)\right), \quad \tilde{\pi}\left(\mathbf{x}_*\right) = \Psi\mathbf{x}_* + \boldsymbol{\nu}, \quad \mathbf{x}_* \in \mathbb{R}^D, \tag{65}$$

where Ψ is a matrix of weights, $\boldsymbol{\nu}$ is a bias vector and D is the state dimension [3]. The RBF policy is represented as

$$\pi\left(\mathbf{x}_*\right) = \mathbf{u}_{\max}\sin\left(\tilde{\pi}\left(\mathbf{x}_*\right)\right), \quad \tilde{\pi}\left(\mathbf{x}_*\right) = \boldsymbol{\beta}_{\pi}^{\top} k_{\pi}\left(\mathbf{X}_{\pi}, \mathbf{x}_*\right), \quad \mathbf{x}_* \in \mathbb{R}^D, \tag{66}$$

where k_{π} is the squared exponential kernel k in (72) plus a noise kernel, and $\boldsymbol{\beta}_{\pi}$ is a weight vector explained in detail in [3]. Common to the policies used in policy search methods are that they are parametrized by a set of parameters. By modifying these parameters, the optimal policy is calculated directly. In the subcategory of *Gradient-based* Policy Search methods, the policy is optimized by employing a gradient ascent approach on the expected rewards. The *Probabilistic Inference for Learning Control* (PILCO) framework falls into this category, and is considered the state of the art approach for solving RL problems with only a small amount of data available [11].

2.8 PILCO

In 2011, Deisenroth and Rasmussen introduced PILCO [12], a practical, data-efficient model-based Policy Search method. As a key attribute, it reduces model bias, one of the main problems of model-based reinforcement learning. This is achieved by learning a probabilistic dynamics model and explicitly incorporating model uncertainty into long-term planning. This way PILCO can cope with a small amount of data, facilitating learning from scratch in only a few trials [12]. For its policy evaluation it is using *Approximate inference*, and for policy improvement it is computing policy gradients analytically. When publishing, they reported unprecedented learning efficiency on challenging and high-dimensional *control* tasks. However, PILCO's time complexity scales with $\mathcal{O}\left(N^2Q^2D^2\right)$, where N is the number of trials, and Q and D are the input- and output dimensionality respectively [34]. Consequently, PILCO will become much slower in high-dimensional *state* tasks and/or tasks that require many trials.

PILCO considers a dynamic system on the form

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}), \tag{67}$$

with unknown transition dynamics f, and continuous-valued states $\boldsymbol{x} \in \mathbb{R}^D$ and control-outputs $\boldsymbol{u} \in \mathbb{R}^F$, where D and F are the dimensions of the state space and the controls respectively. The intention is to find a policy π that minimizes the expected return

$$J^\pi(\theta) = \sum_{t=0}^{T} \mathrm{E}_{\mathrm{x}_t}\left[c\left(\mathrm{x}_t\right)\right], \quad \mathrm{x}_0 \sim \mathcal{N}\left(\mu_0, \boldsymbol{\Sigma}_0\right), \tag{68}$$

over the next T time steps, where $c(\boldsymbol{x}_t)$ is the cost associated with the state $\boldsymbol{x}$ at time t. It is assumed that the policy π is a function parameterized by Θ and that the cost function c encodes information about the given target state $\boldsymbol{x}_{target}$.

Dynamics model learning

The learning of the dynamics model is a key component of the PILCO policy-search framework. The probabilistic model is implemented as a GP, where the tuples $(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}) \in \mathbb{R}^{D+F}$ are used as training inputs and the state transition $\boldsymbol{\Delta}_t = \boldsymbol{x}_t - \boldsymbol{x}_{t-1} + \boldsymbol{\varepsilon} \in \mathbb{R}^D$ as targets, with $\varepsilon \sim \mathcal{N}\left(0, \boldsymbol{\Sigma}_\varepsilon\right), \boldsymbol{\Sigma}_\varepsilon = \operatorname{diag}\left([\sigma_{\varepsilon_1}, \ldots, \sigma_{\varepsilon_D}]\right)$. The one-step prediction of the GP is thus described by the following equations

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t|\mu_t, \Sigma_t), \tag{69}$$

$$\mu_t = \boldsymbol{x}_{t-1} + \mathbb{E}_f[\Delta_t], \tag{70}$$

$$\boldsymbol{\Sigma}_t = var_f[\Delta_t]. \tag{71}$$

In the GP-model, both a prior mean function and a prior covariance function must be specified. The authors of [12] are considering the prior mean function $m \equiv 0$ and use the squared exponential (SE) covariance function k, with automatic relevance determination, defined as

$$k(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}') = \alpha^2 exp(-\frac{1}{2}(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}')^T \boldsymbol{\Lambda}^{-1}(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}')). \tag{72}$$

Here $\tilde{\boldsymbol{x}} := \begin{pmatrix}\boldsymbol{x}^T & \boldsymbol{u}^T\end{pmatrix}^T$ and α^2 is the variance of the latent function f. Finally, $\Lambda := diag([l_1^2, ..., l_D^2])$, where l_i are the characteristic length-scales. These length-scales along with the other hyperparameters (signal variances α^2 and noise variances $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}$) are learned by evidence maximization, using n training inputs $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1, ..., \tilde{\boldsymbol{x}}_n]$ and targets $\boldsymbol{y} = [\boldsymbol{\Delta}_1, ..., \boldsymbol{\Delta}_n]$ [12].

The posterior predictive distribution $p(\boldsymbol{\Delta}_*|\tilde{\boldsymbol{x}}_*)$ of some test input $\tilde{\boldsymbol{x}}_*$ is described by the following mean and variance

$$m_f(\tilde{\boldsymbol{x}}) = \mathbb{E}_f[\boldsymbol{\Delta}_*] = \boldsymbol{K}_*^T(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{y} = \boldsymbol{K}_*^T \beta, \tag{73}$$

$$\sigma_f^2(\Delta_*) = var_f[\Delta_*] = k_{**} - \boldsymbol{k}_*^T(\boldsymbol{K} + \sigma_\epsilon^2\boldsymbol{I})^{-1}\boldsymbol{k}_*. \tag{74}$$

Here $\mathbf{k}_* := k\left(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_*\right), k_{**} := k\left(\tilde{\mathbf{x}}_*, \tilde{\mathbf{x}}_*\right)\ \beta := \left(\mathbf{K} + \sigma_\varepsilon^2\mathbf{I}\right)^{-1}\mathbf{y}$, and $\mathbf{K}$ is the *Gram matrix* with entries $K_{ij} = k\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\right)$. When dealing with multivariate targets, a dedicated GP is assigned to each target dimension. These GP's are independent given deterministic test inputs, but covaries if the inputs are uncertain.

Policy evaluation

To be able to minimize and evaluate the cost J^π in (68) you are dependent on long-term predictions of the state progression. The subsequent state distributions $p(\boldsymbol{x}_1), ..., p(\boldsymbol{x}_T)$, are obtained by cascading one-step predictions as in Eqs. (69)-(71). Accounting for the variability, the uncertain test inputs are mapped through the GP dynamics model. For simplicity, these test inputs are considered to be Gaussian distributed.

As the next state is dependent on the action of the control, to predict $\boldsymbol{x}_t$ from $p(\boldsymbol{x}_{t-1})$, a joint distribution $p(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})$ is required. Again, as the control $\boldsymbol{u}_{t-1} = \pi(\boldsymbol{x}_{t-1}, \Theta)$ is a function of the state, its distribution $p(\boldsymbol{u}_{t-1})$ must be found in advance. This is done by integrating out the state and computing the mean μ_u and covariance Σ_u. Thereafter, the cross-covariance $cov[\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}]$ is determined. Finally, the joint distribution $p(\tilde{\boldsymbol{x}}_{t-1}) = p(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})$ is approximated by a Gaussian with the right mean and covariance. As the control distribution $p(\boldsymbol{u}_{t-1})$ is found from the policy function $\pi(\boldsymbol{x}_{t-1}, \Theta)$, the computations depend on the parametrization of the policy π [12].

The distribution of the state changes are given by

$$p(\Delta_t) = \int p(f(\tilde{\boldsymbol{x}}_{t-1})|\tilde{\boldsymbol{x}}_{t-1})p(\tilde{\boldsymbol{x}}_{t-1})d\tilde{\boldsymbol{x}}_{t-1}, \tag{75}$$

where the random variable $\tilde{\boldsymbol{x}}_{t-1}$ is integrated out. Here $p(f(\tilde{\boldsymbol{x}}_{t-1})|\tilde{\boldsymbol{x}}_{t-1})$, noting the transition probability, is acquired from the posterior GP distribution. $p(\tilde{\boldsymbol{x}}_{t-1})$ is assumed to be a joint Gaussian distribution, $p\left(\tilde{\mathbf{x}}_{t-1}\right) = \mathcal{N}\left(\tilde{\mathbf{x}}_{t-1} \mid \tilde{\mu}_{t-1}, \tilde{\boldsymbol{\Sigma}}_{t-1}\right)$. As it is analytically intractable to compute the exact predictive distribution in (75), $p(\Delta_t)$ is approximated by a Gaussian using exact moment matching [12], illustrated in Fig. 10.

Given a known mean μ_Δ and covariance $\boldsymbol{\Sigma}_\Delta$ of the predictive distribution $p(\Delta_t)$, the Gaussian approximation of the desired distribution $p(\boldsymbol{x}_t)$ is identified as $\mathcal{N}(\boldsymbol{x}_t|\mu_t, \boldsymbol{\Sigma}_t)$ with

$$\mu_t = \mu_{t-1} + \mu_\Delta, \tag{76}$$

$$\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_{t-1} + \boldsymbol{\Sigma}_\Delta + \operatorname{cov}\left[\mathbf{x}_{t-1}, \Delta_t\right] + \operatorname{cov}\left[\Delta_t, \mathbf{x}_{t-1}\right], \tag{77}$$

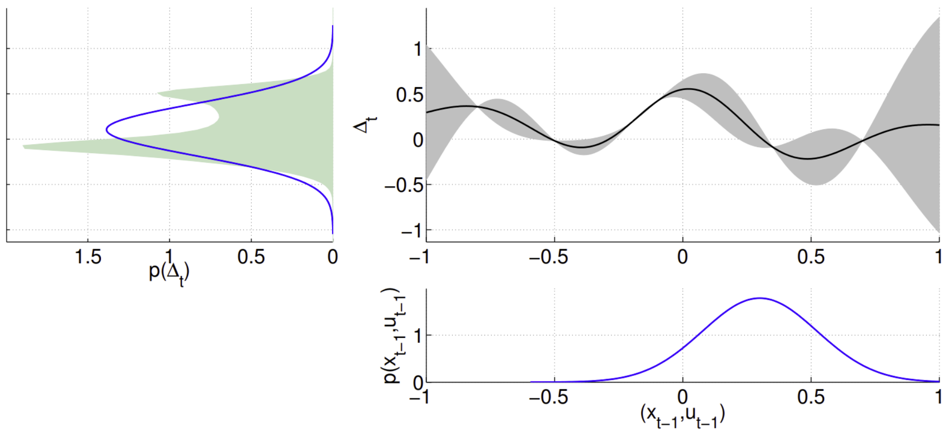


Figure 10: GP prediction at an uncertain input [12]. The lower right plot is illustrating the input distribution $p(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})$, assumed to be Gaussian. Propagating the uncertain input through the GP model (upper right), the shaded distribution $p(\Delta_t)$ is obtained (upper left). $p(\Delta_t)$ is approximated by a Gaussian with the exact mean and variance (blue outline)

$$\operatorname{cov}\left[\mathbf{x}_{t-1}, \Delta_t\right] = \operatorname{cov}\left[\mathbf{x}_{t-1}, \mathbf{u}_{t-1}\right] \boldsymbol{\Sigma}_u^{-1} \operatorname{cov}\left[\mathbf{u}_{t-1}, \Delta_t\right]. \tag{78}$$

The computation of the cross-variances in (78) is decided by the policy parameterization, but can often be determined analytically. For the computation of $\operatorname{cov}\left[\mathbf{x}_{t-1}, \Delta_t\right]$, the reader is referred to [3]. The computation of μ_Δ and $\boldsymbol{\Sigma}_\Delta$ is detailed in [3] and [12]. Having computed these respective components, making up the Gaussian approximation $\mathcal{N}(\boldsymbol{x}_t|\mu_t, \boldsymbol{\Sigma}_t)$ of $p(\boldsymbol{x}_t)$, the expected return J^π in (68) can be found through

$$\mathbb{E}_{\mathbf{x}_t}\left[c\left(\mathbf{x}_t\right)\right] = \int c\left(\mathbf{x}_t\right) \mathcal{N}\left(\mathbf{x}_t \mid \mu_t, \boldsymbol{\Sigma}_t\right) \mathrm{d}\mathbf{x}_t, \tag{79}$$

where the cost function c is chosen so that (79) can be solved analytically. The authors use

$$c(\mathbf{x}) = 1 - \exp\left(-\left\|\mathbf{x} - \mathbf{x}_{\text{target}}\right\|^2 / \sigma_c^2\right) \in [0, 1], \tag{80}$$

where $\boldsymbol{x}_{target}$ is the target state and σ_c^2 decides the width of c.

Analytic gradients for policy improvement

As both μ_t and $\boldsymbol{\Sigma}_t$ are dependent on the mean and covariance of the control signal (and Θ), the gradients of the expected rewards J^π with respect to the policy parameters can be computed analytically. The derivative $dJ^\pi/d\Theta$ is found by repeated chain-rule computations. Defining $\mathcal{E} := \mathbb{E}_{\mathbf{x}_t}\left[c\left(\mathbf{x}_t\right)\right]$ and swapping the order of summing and differentiating leads to the equation

$$\frac{\mathrm{d}\mathcal{E}_t}{\mathrm{d}\theta} = \frac{\mathrm{d}\mathcal{E}_t}{\mathrm{d}p\left(\mathbf{x}_t\right)} \frac{\mathrm{d}p\left(\mathbf{x}_t\right)}{\mathrm{d}\theta} := \frac{\partial \mathcal{E}_t}{\partial \mu_t} \frac{\mathrm{d}\mu_t}{\mathrm{d}\theta} + \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{\Sigma}_t} \frac{\mathrm{d}\boldsymbol{\Sigma}_t}{\mathrm{d}\theta}, \tag{81}$$

where $\mathrm{d}\mathcal{E}_t / \mathrm{d}p(\mathbf{x}_t)$ is a short hand notation for for taking the derivative of $\mathcal{E}_t$ with respect to both the mean μ_t and covariance $\mathbf{\Sigma}_t$ of $\boldsymbol{x}_t$. Applying the chain-rule to (81) results in

$$\frac{\mathrm{d}p(\mathbf{x}_t)}{\mathrm{d}\theta} = \frac{\partial p(\mathbf{x}_t)}{\partial p(\mathbf{x}_{t-1})} \frac{\mathrm{d}p(\mathbf{x}_{t-1})}{\mathrm{d}\theta} + \frac{\partial p(\mathbf{x}_t)}{\partial \theta}, \tag{82}$$

$$\frac{\partial p(\mathbf{x}_t)}{\partial p(\mathbf{x}_{t-1})} = \left\{ \frac{\partial \mu_t}{\partial p(\mathbf{x}_{t-1})}, \frac{\partial \mathbf{\Sigma}_t}{\partial p(\mathbf{x}_{t-1})} \right\}. \tag{83}$$

Furthermore, we have that

$$\frac{\mathrm{d}\mu_t}{\mathrm{d}\theta} = \frac{\partial \mu_t}{\partial \mu_{t-1}} \frac{\mathrm{d}\mu_{t-1}}{\mathrm{d}\theta} + \frac{\partial \mu_t}{\partial \mathbf{\Sigma}_{t-1}} \frac{\mathrm{d}\mathbf{\Sigma}_{t-1}}{\mathrm{d}\theta} + \frac{\partial \mu_t}{\partial \theta}, \tag{84}$$

finally leading to

$$\frac{\partial \mu_t}{\partial \theta} = \frac{\partial \mu_\Delta}{\partial p(\mathbf{u}_{t-1})} \frac{\partial p(\mathbf{u}_{t-1})}{\partial \theta} = \frac{\partial \mu_\Delta}{\partial \mu_u} \frac{\partial \mu_u}{\partial \theta} + \frac{\partial \mu_\Delta}{\partial \mathbf{\Sigma}_u} \frac{\partial \mathbf{\Sigma}_u}{\partial \theta}. \tag{85}$$

This method of analytic gradient computation of J^π is a lot more efficient than the alternative of estimating policy gradients via sampling [12]. For a more detailed explanation the reader is referred to [3].

Algorithm 1: PILCO

```
init: Sample control parameters $\Theta \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
Apply random control signals and record data;
repeat
    Learn probabilistic (GP) dynamics model, using all data.;
    Model-based policy search;
    repeat
        Approximate inference for policy evaluation ;
        Gradient-based policy improvement;
        Update parameters $\Theta$;
    until convergence; return $\Theta^*$;
    Set $\pi^* \leftarrow \pi(\Theta^*)$ ;
    Apply $\pi^*$ to system (single trial/episode) and record data;
until task learned;
```

3 Robot-Hardware and Software

This section aims to describe the hardware and software enabling the implementations and assessments treated in Section 4 and 5. First, the basic characteristics of the robot are presented. This is followed by reviews of the interfaces allowing intuitive and straightforward implementation of control. Finally, the software most essential to the introduction of learning is presented.

3.1 Franka Emika Panda Robot

The Panda robot, manufactured by Franka Emika, is a collaborative lightweight robot designed to act like an agile human arm. It has 7-DOF, a payload of 3 kg, a weight of 18 kg and a radius (reach) of 855 mm [36]. Through its joint torque sensors and forward kinematics, it can calculate an estimate of the external wrench at the end-effector. In addition to the robot arm itself, the setup also consists of an external activation device, a shop floor controller and a shared network switch, connecting everything to a workstation running the respective control-scripts (Fig. 11).

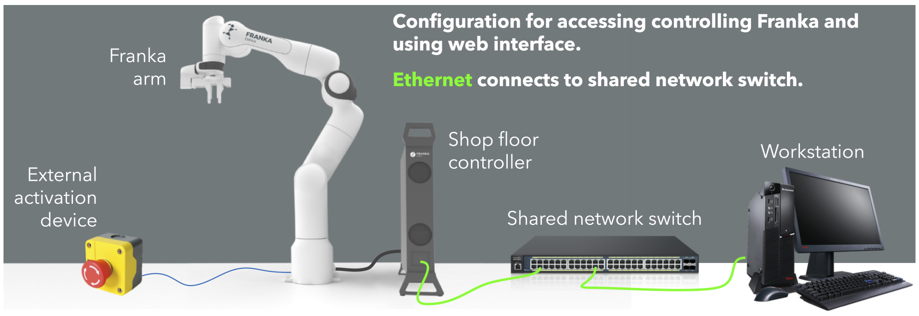

Franka arm
Configuration for accessing controlling Franka and using web interface.
Ethernet connects to shared network switch.
Shop floor controller
Workstation
External activation device
Shared network switch

Figure 11: Setup for controlling the Panda robot using web interface [13]

3.2 Robot Operating System

Robot Operating System (ROS) is a flexible framework for writing robot software. Specifically, it is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms [37]. In sum, these components provide services such as hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management [38]. Today, there exists several versioned sets of ROS packages, known as ROS *Distributions*. In this setup, ROS Melodic Morenia is the one being used, released in May 2018. This Distribution is primarily targeting Ubuntu 18.04 (Bionic Beaver).

3.3 Franka Control Interface

Franka Control Interface (FCI) allows a fast and direct low-level bidirectional connection to the arm and hand [14]. The schematic overview of this interaction is depicted in Fig. 12.

Furthermore, the figure shows the components that make up the FCI, namely the C++ library *libfranka* and the ROS integration *franka_ros*. Out of the two, *libfranka* is the one providing low-level control of the Franka robots. It is handling the network communication with the control unit (Fig. 12) and provides interfaces to easily execute commands, read robot states and access the model library [39]. *franka_ros* on the other hand, is a metapackage for all Franka Emika ROS packages, providing the robots with ROS support [40]. In addition to its ROS contribution, it ensures support for the motion planning framework MoveIt!. Also *franka_description*, a collection of URDF models and 3D meshes of the Franka robots, is a part of this component, having a utility that extends beyond ROS [41].

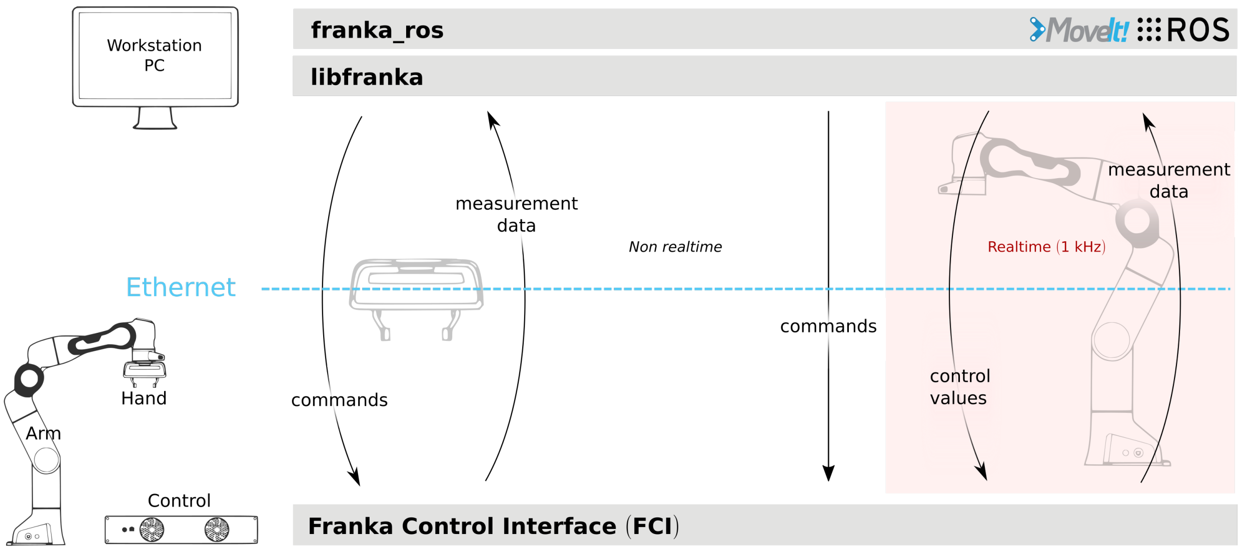


Figure 12: Schematic overview of the FCI's workflow [14]

3.4 Extensions to FCI

With the goal of making it easier and more intuitive to make controllers for the Panda robot, several independent developers have created their own, extended interfaces. Saif Sidhik, behind the Github account "justagist" is providing such an interface. His repositories *franka_ros_interface*, *panda_robot* and *panda_simulator* [42] are built upon the FCI, and make up a unified interface for controlling the Franka Emika Panda robot in Python (Fig. 13).

franka_ros_interface

Out of the three repositories, *franka_ros_interface* is the one working closest to the FCI. In short, it is a ROS package extending the *franka_ros* to expose more information about the robot, and providing low-level control of the robot using ROS and Python API. [43]. To do so, it is dependent on the package *franka_panda_description*, operating as an extension of *franka_ros*' package *franka_description*. This package is providing dynamics parameters for the robot arm and support for simulation in Gazebo.

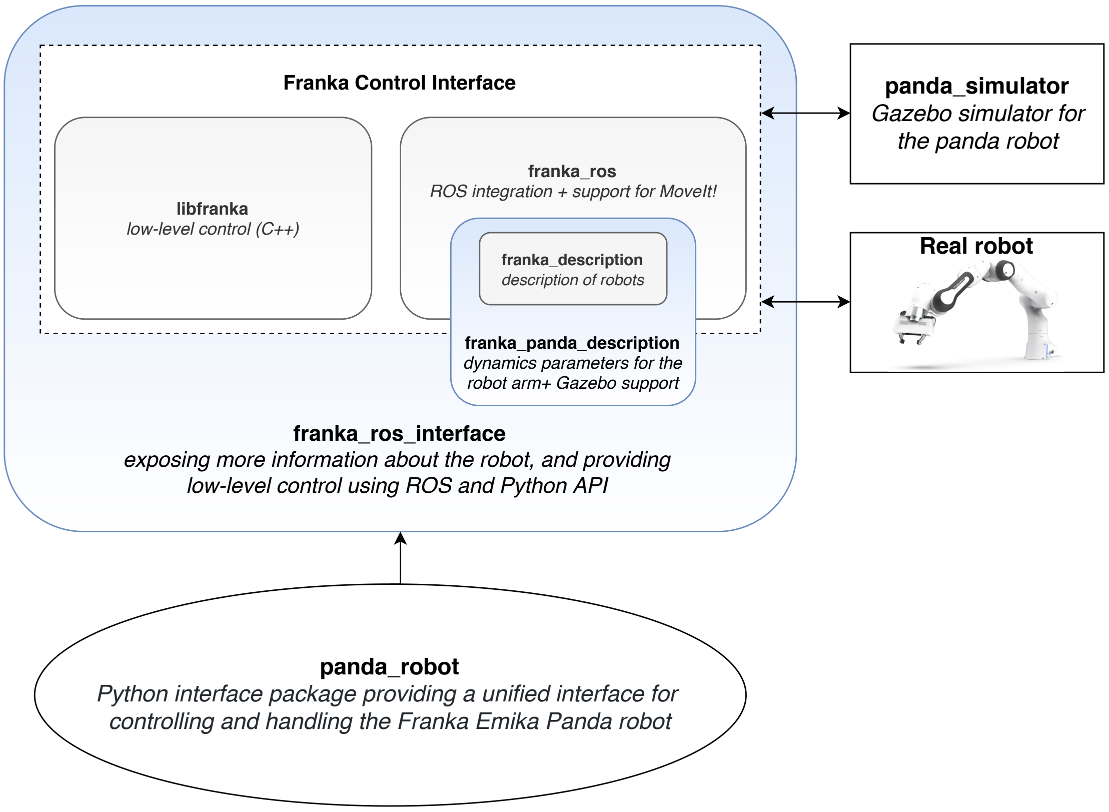

Franka Control Interface
libfranka
low-level control (C++)
franka_ros
ROS integration + support for MoveIt!
franka_description
description of robots
franka_panda_description
dynamics parameters for the robot arm+ Gazebo support
franka_ros_interface
exposing more information about the robot, and providing low-level control using ROS and Python API
panda_simulator
Gazebo simulator for the panda robot
Real robot
panda_robot
Python interface package providing a unified interface for controlling and handling the Franka Emika Panda robot

Figure 13: Illustration of dependencies and extensions among the most essential libraries

panda_simulator

panda_simulator is a Gazebo-based simulator for the robot, using *franka_ros_interface* to provide exposed controllers and real-time robot state feedback, similar to that of the real robot when using FCI's *franka_ros* package [44]. Furthermore, it has support for MoveIt! planning.

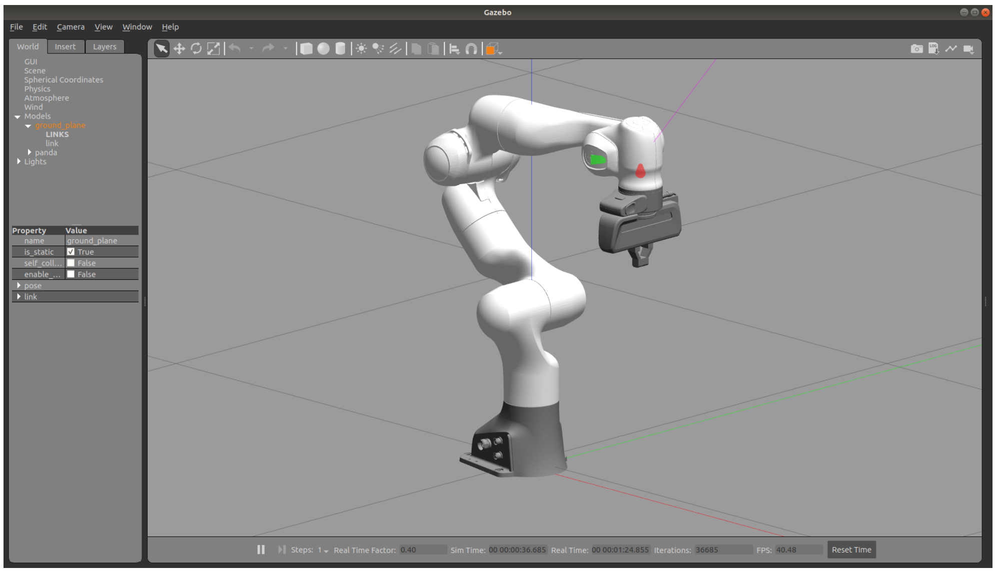


Figure 14: Screenshot of the running Gazebo simulator

panda_ robot

panda_ robot is a Python interface package built over the *franka_ ros_ interface* package. It is combining its different classes to provide a unified interface for controlling and handling the Franka Emika Panda robot [45]. Moreover, it provides a direct *sim-to-real* code transfer, making it possible to run the same script both in the simulator and on the real robot.

3.5 PILCO Implementation

Regarding implementations of the PILCO algorithm, the Github repository `https://github.com/nrontsis/PILCO` [46] proved to be a convenient choice. The package aims to provide a clean implementation by heavy use of modern learning libraries in Python 3. Specifically it uses Tensorflow v2 to avoid the need for hardcoded gradients and scale to GPU architectures. For Gaussian Process Regression it uses GPflow v2. The repository provides the user with reward functions, controllers (policy models), and functionality for model- and policy optimization.

3.6 OpenAI Gym

OpenAI Gym [47] is a toolkit for developing and comparing RL algorithms. In terms of development, it provides a generic interface that facilitates the introduction of RL in any given environment. The interface is accessed by structuring your environment as a *Gym environment*, essentially as a high level Python class with a few required functions. The functions *init*, *step* and *reset* are requisites, responsible for initialisation, interaction with the environment, and resetting, respectively.

4 Implementation

In this section, the implementation of control and learning environment is explained, followed by details on how to run the complete setup. In the first subsection, the implementations of the three different force controllers are described. Then the process of making the robot environment PILCO-compatible is explained. Next, the details on introducing learning-based variability in the force controllers are reviewed. Finally, the process of setting up the environment and testing the controllers both in simulation and the real robot is described.

4.1 Implementation of Force Controllers

Using the extended FCI described in Section 3.4, the states of the robot are fetched in the python interface. In terms of reaching the desired state, these state-measurements are used directly to compute an appropriate response from the respective controllers. States, like the position x and force F, are fetched in every control step along with state-dependent properties such as the jacobian $J(\boldsymbol{q})$, the coriolis compensation term $\boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$, and the joint inertia matrix $\boldsymbol{\Lambda}(\boldsymbol{q})$. However, tests have shown that this fetching of states through the extended FCI is inefficient. To obtain the required state information can take as much as 20 ms, acting as a limiting factor for the control frequencies. The slow data-fetching is caused by the additional abstraction layer that the extended FCI is introducing. Even though the robot is communicating at 1 kHz, the functions of the python interface is only fetching a few states for every fetch-function. Thus, several cycles of communication are spent on data-fetching at every control step.

Admittance Control implementation

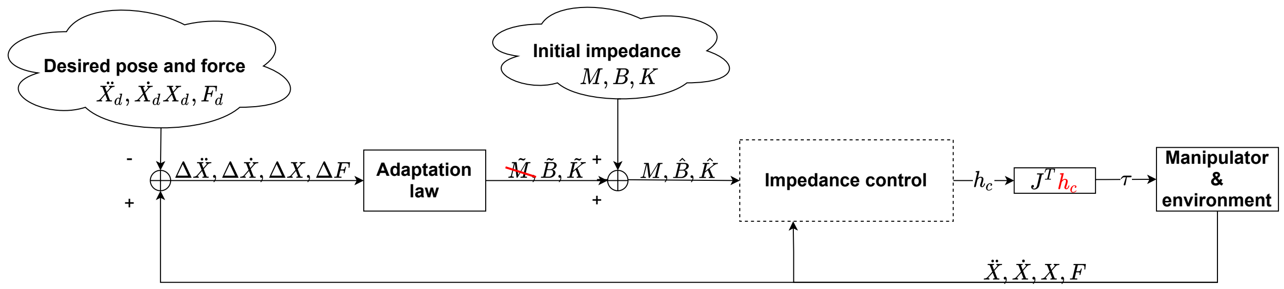
The implementation of the admittance controller is quite straightforward with respect to the information given in Section 2.5. With regard to the use case, the translation in the z dimension is chosen to be compliant, meaning that all parameters in (36) relates to the z dimension. M, B and K are therefore tunable scalars, specifying the force tracking ability and compliance in z through (36). The compliant position $\boldsymbol{X}_c$, consisting of the desired position $\boldsymbol{X}_d$ with some adjustment in z, is then fed to the position controller. The position controller is provided in the *panda_robot* package described in Section 3.4.

Hybrid Control implementation

Also the Hybrid Controller is implemented as described in the background section. By choosing $\boldsymbol{S}_f$ and $\boldsymbol{S}_v$ as in (37), the Hybrid Control is specified to perform force control in z and compliant motion control in the remaining five dimensions. With the exception of the desired force and motion, the environmental compliance matrix $\boldsymbol{C} = \boldsymbol{K}^{-1}$, the gains of the motion controller ($\boldsymbol{K}_{Dr}, \boldsymbol{K}_{Pr} \in \mathbb{R}^{5\times5}$), and the gains of the force controller ($K_{D\lambda}, K_{P\lambda} \in \mathbb{R}$), all parameters of the control laws are purely state-dependent. Leaving $\boldsymbol{K}_{Dr}, \boldsymbol{K}_{Pr}, K_{D\lambda}, K_{P\lambda}$ and $\boldsymbol{C}$ for tuning. All matrices are chosen to be diagonal, with values derived from testing in simulation and in experiments.

Force-based VIC implementation

The implementation of the Force-based VIC requires a more thorough review than the previous controllers. The final controller is a combination of the impedance control law in (27) from Section 2.5 and a modified version of the adaptive impedance law (60) from Section 2.6. H.P. Huang and S.S. Chen, presenting the adaptive impedance law in [17], also proposed a control law for the impedance controller itself, but this implementation did not satisfy the requirement of geometrically consistent active stiffness. Consequently, it performed sub-optimally in the simulated tests, motivating the use of the implementation reported in Section 2.5. The Force-based VIC is thus implemented as in Fig. 15 with the impedance controller itself implemented as in Fig. 16.



Desired pose and force
$\ddot{X}_d, \dot{X}_d X_d, F_d$
Initial impedance
M, B, K
-
+
$\Delta\ddot{X}, \Delta\dot{X}, \Delta X, \Delta F$
Adaptation law
$\cancel{\tilde{M}}, \tilde{B}, \tilde{K}$
+
+
$M, \hat{B}, \hat{K}$
Impedance control
h_c
$J^T h_c$
τ
Manipulator & environment
$\ddot{X}, \dot{X}, X, F$

Figure 15: High level block diagram of the adjusted Force-based VIC. The adaptation law is specified in (60)

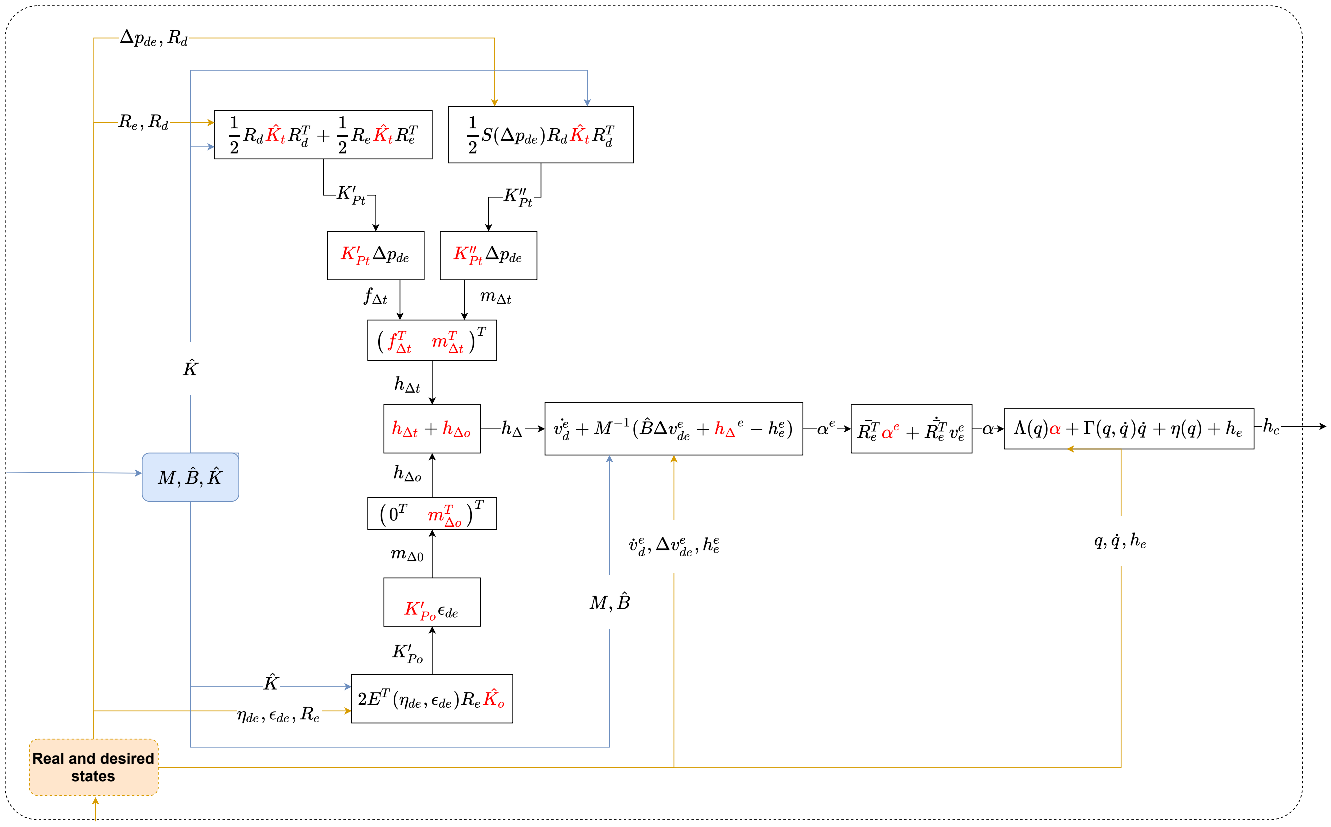
When only doing force control in z, the adaptive laws in (60) only applies to the z-dimensional properties of $\hat{\boldsymbol{M}}$, $\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{K}} \in \mathbb{R}^{6\times6}$. However, since an adaptive inertia-matrix $\hat{\boldsymbol{M}}$ easily can lead to instability, the inertia is chosen to be static. This results in a system with adaptive damping and stiffness in z. Now choosing $\boldsymbol{K}_v$ and $\boldsymbol{P}$ equal to $\boldsymbol{I} \in \mathbb{R}^{6\times6}$ in the adaptive law (60), the law is reduced to

$$\dot{\boldsymbol{\lambda}}(\dot{\tilde{\boldsymbol{B}}}, \dot{\tilde{\boldsymbol{K}}}) = \begin{pmatrix} 0... & ... & \gamma_B^{-1}\dot{E}_z E_{fz} & 0... & \gamma_K^{-1} E_z E_{fz} & 0... \end{pmatrix}^T \in \mathbb{R}^{18\times1}, \tag{86}$$

where E_z is the error in z-position, E_{fz} is the error in z-force, and γ_B^{-1} and γ_K^{-1} are the rate of adaptability for the damping and stiffness in z respectively. The damping and stiffness matrices are thus denoted

$$\hat{\boldsymbol{B}} = \boldsymbol{B} + \tilde{\boldsymbol{B}}, \qquad \hat{\boldsymbol{K}} = \boldsymbol{K} + \tilde{\boldsymbol{K}} \tag{87}$$

where $\boldsymbol{B}$ and $\boldsymbol{K}$ are the initial damping and stiffness, and $\tilde{\boldsymbol{B}}$ and $\tilde{\boldsymbol{K}}$ are the results of the



$\Delta p_{de}, R_d$
R_e, R_d
$\frac{1}{2}R_d\hat{K}_t R_d^T + \frac{1}{2}R_e\hat{K}_t R_e^T$
$\frac{1}{2}S(\Delta p_{de})R_d\hat{K}_t R_d^T$
K'_{Pt}
K''_{Pt}
$K'_{Pt}\Delta p_{de}$
$K''_{Pt}\Delta p_{de}$
$f_{\Delta t}$
$m_{\Delta t}$
$(f_{\Delta t}^T \quad m_{\Delta t}^T)^T$
$\hat{K}$
$h_{\Delta t}$
$h_{\Delta t} + h_{\Delta o}$
h_Δ
$\dot{v}_d^e + M^{-1}(\hat{B}\Delta v_{de}^e + h_\Delta{}^e - h_e^e)$
α^e
$\bar{R}_e^T\alpha^e + \dot{\bar{R}}_e^T v_e^e$
α
$\Lambda(q)\alpha + \Gamma(q,\dot{q})\dot{q} + \eta(q) + h_e$
h_c
$M, \hat{B}, \hat{K}$
$h_{\Delta o}$
$(0^T \quad m_{\Delta o}^T)^T$
$m_{\Delta 0}$
$\dot{v}_d^e, \Delta v_{de}^e, h_e^e$
$q, \dot{q}, h_e$
$M, \hat{B}$
$K'_{Po}\epsilon_{de}$
K'_{Po}
$\hat{K}$
$\eta_{de}, \epsilon_{de}, R_e$
$2E^T(\eta_{de}, \epsilon_{de})R_e\hat{K}_o$
Real and desired states

Figure 16: The implementation details of the impedance control block in Fig. 15. The controller is receiving the inertia $\boldsymbol{M}$, damping $\hat{\boldsymbol{B}}$ and stiffness $\hat{\boldsymbol{K}}$ from the adaptation law in (60)

adaptive law. $\tilde{\boldsymbol{B}}$ and $\tilde{\boldsymbol{K}}$ are initialised at $\mathbf{0}$ and tuned by

$$\dot{\tilde{\boldsymbol{B}}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \gamma_B^{-1}\dot{E}_z E_{fz} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{88}$$

$$\dot{\tilde{\boldsymbol{K}}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \gamma_K^{-1}E_z E_{fz} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{89}$$

as detailed in (86). Thus, the impedance controller is fed with a constant inertia matrix $\boldsymbol{M}$

and damping- and stiffness matrices on the form

$$\hat{\boldsymbol{B}}, \hat{\boldsymbol{K}} = \begin{pmatrix} fixed(x) & 0 & 0 & 0 & 0 & 0 \\ 0 & fixed(y) & 0 & 0 & 0 & 0 \\ 0 & 0 & adaptive(z) & 0 & 0 & 0 \\ 0 & 0 & 0 & fixed(x_{ori}) & 0 & 0 \\ 0 & 0 & 0 & 0 & fixed(y_{ori}) & 0 \\ 0 & 0 & 0 & 0 & 0 & fixed(z_{ori}) \end{pmatrix}. \tag{90}$$

$\boldsymbol{M}$, $\boldsymbol{B}$ and $\boldsymbol{K}$ are chosen to be diagonal with values tuned to provide stable results in tests. Also γ_B^{-1} and γ_K^{-1} are found from testing and are designed to have $\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{K}}$ adjust at an appropriate rate. Furthermore, upper and lower bounds are set for $\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{K}}$, preventing instability.

4.2 Making the Robot Environment PILCO Compatible

The PILCO algorithm, as other RL methods, needs their *agent* to have a certain access to the states of the environment it is interacting with. In this case, the environment consists of the respective force controller, the robot itself, and the object it is interacting with. The agent is to change certain stiffness- and damping related parameters in the force controllers, with the end goal of improving performance. To achieve this, the agent requires knowledge of the states, and an ability to influence the environment, both in real time. OpenAI's toolkit *Gym* provides a convenient platform for such an interface. By setting up the robot environment as a *Gym environment*, which essentially is a python class with all the necessary functions, it is providing the agent with the required information.

As the PILCO-implementation from Section 3.5 is highly dependent on different machine learning libraries in Python 3, there was a need to provide compatibility with the Python 2 dependent robot environment. This problem was solved using the Python package *execnet*. *execnet* allows one to initiate a subprocess gateway, execute code in it and send bidirectional messages. By initiating Python 2 subprocess gateways in Python 3 scripts, all code related to running the robot-environment could be executed separately, continuously conveying information relevant to the PILCO-algorithm. This way the PILCO-algorithm is enabled to receive data from the robot-process, now represented as a *Gym environment*, and respond with appropriate actions.

4.3 Combining PILCO and Force Control

To be able to calculate appropriate actions, PILCO must first learn a model of the transition dynamics. As the motivation for introducing learning is to achieve better force tracking, the entity that we really want to predict is the contact force. For this purpose, the state space was chosen as

$$\boldsymbol{x} = \begin{pmatrix} F & p_z & v_z \end{pmatrix}, \tag{91}$$

where F is the contact force, p_z is the z-position and v_z is the z-velocity. Due to the controllers' different approaches to force-control, their action-spaces are not chosen equal. However, in all three cases, one action is related to damping in z, and one to stiffness in z. Denoting the action space as $\boldsymbol{u} \in \mathbb{R}^2$, the training inputs of the model are identified as $(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}) \in \mathbb{R}^5$. Now using the state transitions $\boldsymbol{\Delta}_t = \boldsymbol{x}_t - \boldsymbol{x}_{t-1} + \boldsymbol{\epsilon} \in \mathbb{R}^3$ as targets, the dynamics model can be learned according to the method depicted in Section 2.8.

Regarding policy models, the PILCO-implementation offers both linear and RBF, both described in Section 2.7. The RBF policy models are much more flexible, enabling more complex control strategies. However, it is computationally expensive. In fact, tests proved that the RBF action-calculation of the PILCO-implementation is too slow to be used in the experimental setup. Having the ability to slow down time in the simulator, it is however possible to test it in there. The linear policy limits the maximal frequency of the controller as well, but to a lesser extent. Especially as we were able to optimize the function responsible for its action computation in [46], cutting the computation time to a tenth. Subsampling is also an important aspect when applying RL to a system. By for example only sampling every fifth state-vector, the amount of data collected from each run can be severely reduced. If the underlying controller is running on 100 Hz, this would lead to a PILCO-frequency of 20 Hz. Thus, implying that the policy will have a 20 Hz action-frequency, and that the learned model will have a resolution of 50 ms. It was found from testing that 20 Hz is an appropriate frequency for the PILCO-framework. The testing is documented in Section 5.

In order for the PILCO-algorithm to evaluate and improve the policy, a reward function is required, specifying the desired behaviour of the system. To this purpose, the exponential reward function provided in the PILCO-package is used. The desired behaviour is specified through a constant target- and a weight-vector linked to the state $\boldsymbol{x}$. By setting $\boldsymbol{t} = \begin{pmatrix} F_d & 0 & 0 \end{pmatrix}$ and $\boldsymbol{W} = \begin{pmatrix} 5 & 0 & 0 \end{pmatrix}$, the agent can at each time step receive a reward of up to 5 for tracking the desired force F_d.

Admittance Control in the PILCO framework

As it was covered in Section 4.1, the compliance of the Admittance Controller is defined by the three scalars M, B and K, representing the inertia, damping and stiffness in the z-plane of the manipulator. Due to unfortunate consequences of having a dynamic inertia, M was chosen to be constant. B and K on the other hand, are decided to be part of the agent's action space. The PILCO-algorithm sees this action space as two continuous intervals, spanning from -1 to 1. However, when the action signals arrive in the robot environment, they are scaled to some appropriate intervals. For the admittance controller, this meant scaling the damping B so that it could vary in the range from 150 and 400, and K between 200 and 500.

Hybrid Control in the PILCO framework

In Section 4.1, $\boldsymbol{K}_{Dr}, \boldsymbol{K}_{Pr}, K_{D\lambda}, K_{P\lambda}$ and $\boldsymbol{C}$ were identified as tunable parameters, responsible for the behaviour of the Hybrid Control. As the main focus of the thesis is directed at force tracking, $\boldsymbol{K}_{Dr}$ and $\boldsymbol{K}_{Pr}$, responsible for the motion tracking, is left constant. As we in tests will have a constant desired contact force F_d (λ_d), it can be seen from Fig. 4 that it would be redundant to include $\boldsymbol{C}$ in the adaptive strategy along with $K_{D\lambda}$ and $K_{P\lambda}$. Thus, only $K_{D\lambda}$ and $K_{P\lambda}$ are decided to constitute the action space of the PILCO-agent.

Force-based VIC in the PILCO framework

The Force-based VIC differ from the other two controllers in terms of already, before introducing learning, having variable damping and stiffness parameters in z (90). As it is detailed in (88) and (89), their adaptive laws are given by $\dot{\tilde{B}}_z = \gamma_B^{-1}\dot{E}_z E_{fz}$ and $\dot{\tilde{K}}_z = \gamma_K^{-1} E_z E_{fz}$ respectively. The tunable rates of adaptability, γ_B^{-1} and γ_K^{-1}, clearly have a great influence on the dynamic behavior of the system. Thus, for the Force-based VIC, PILCO is used to learn adaptive strategies for γ_B^{-1} and γ_K^{-1}. As for the action spaces of the other controllers, their boundaries are found through trial and error.

4.4 Setting Up the Environment

In order to run the implemented controllers, available at `https://github.com/martihmy/Compliant_control`, either in the simulator or in the experimental setup, the necessary software environment needs to be set up. This is achieved by following the procedure described in this section.

Setup of the extended FCI

The extended FCI can be set up in an Ubuntu 18.04 system in the following steps:

1. Install ROS Melodic Morenia
2. Build *libfranka* and *franka_ros*
3. Set up the real-time kernel
4. Build *panda_simulator* and *panda_robot*

As the extended interface from Section 3.4 has support for ROS Melodic Morenia, this is the ROS distribution we are using. Accordingly, the workstation is running Ubuntu 18.04.5 LTS (Bionic Beaver) which the ROS distribution is targeting. Both the building of *libfranka* and *franka_ros*, and the setting up of the real-time kernel, needed to control the robot, are described in detail in [48]. The process of building *panda_simulator* and *panda_robot* is done according to the respective README files on Github. *franka_ros_interface* and *franka_panda_description* are both installed when performing the fourth step.

Setup of the PILCO compatibility

All compatibility related to PILCO can be achieved through the following steps:

1. Install the PILCO implementation
2. Install OpenAI *Gym*
3. Install *execnet*

The PILCO implementation in `https://github.com/nrontsis/PILCO` should preferable be installed in a fresh Conda environment with Python ≥ 3.7. The installation itself is performed by running the following two commands in the terminal:

```
git clone https://github.com/nrontsis/PILCO && cd PILCO
python setup.py develop
```

Step 2 and 3 are performed by similarly running

```
pip install gym
```

in the python 2 environment, and

```
pip install execnet
```

in the Conda environment.

Setup of the implemented controllers

In order to run the implemented controllers, the package `https://github.com/martihmy/Compliant_control`, is to replace *panda_simulator*. Additionally, the four python scripts in the folder *Modifications*, is to replace the files with the same names in the *PILCO*-, *franka_ros_interface*- and *panda_robot* packages respectively.

5 Results

In this section, the results from simulation and experimental studies are presented, for scenarios with and without learning. This section is divided into three subsections. In the first part, the respective force controllers are evaluated, both in simulation and experiment. The second part is evaluating the force controllers in the learning framework, also in simulation and experiment. The last subsection is assessing the introduction of supplemental transition models, following the same structure.

Regarding tuning of the controllers' static parameters, equivalent efforts have been made in all three controllers. The focus being on their respective performances in the learning framework, they have not been perfectly tuned. Where there has been a trade-off between motion- and force control, the aim has been to achieve an overall balanced performance. Nevertheless, as the results will show, the strengths and weaknesses of the controllers still materialize. The results presented are representative outcomes, aimed at demonstrating the average performance of the respective implementations.

5.1 Evaluation of Force Controllers

Before introducing learning, the three force controllers are to be assessed with static control laws. As method of evaluation, the controllers are to ensure motion- and force tracking for the robots' end-effector when interacting with a flat surface. All three controllers are first tested in simulation, then in the experimental setup.

5.1.1 Simulation

The simulations are performed in the Gazebo-based simulator described in Section 3.4 and are expected to give a fair indication of the real system's behaviour. However, a significant advantage of the simulation is the ability to slow down real-time, enabling control frequencies of 100 Hz. Fist, the respective controllers' performance in a homogeneous environment is tested. The environment is a flat surface with a stiffness coefficient of $kp = 5.000$, and a damping coefficient of $kd = 3$. The setup is illustrated in Fig. 17. With the automatic ultrasound use case in mind, the objective is for the respective controllers to establish a vertical contact force of 3 N with the surface, before performing a 5 cm movement forward, maintaining a desired contact force of 3 N. The orientation shall remain neutral (as in Fig. 17) throughout the duration of the task.

To increase the legitimacy of the simulated tests, noise is added to the force estimates before arriving at the controllers. The noise is drawn from a Gaussian distribution with a 0 mean and a standard deviation of 1.5 % of the absolute value of the original force estimate. The magnitude of the noise was found through comparison with the real system. To be on the safe side, the noise is chosen just larger than that of the real system. A comparison of the Hybrid Controller's performance with and without added sensor noise is shown in Fig. 18. Throughout this chapter, noise is added to the force estimates of all simulated tests unless otherwise stated.

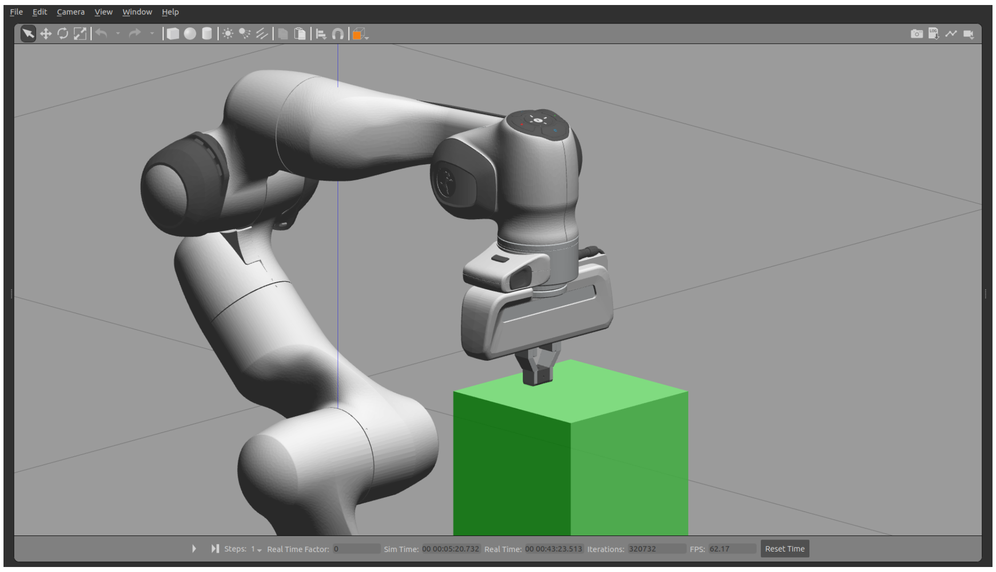


Figure 17: The setup used when testing the force controllers without learning. The object (green box) has a stiffness coefficient $kp = 5.000$, and a damping coefficient $kd = 3$

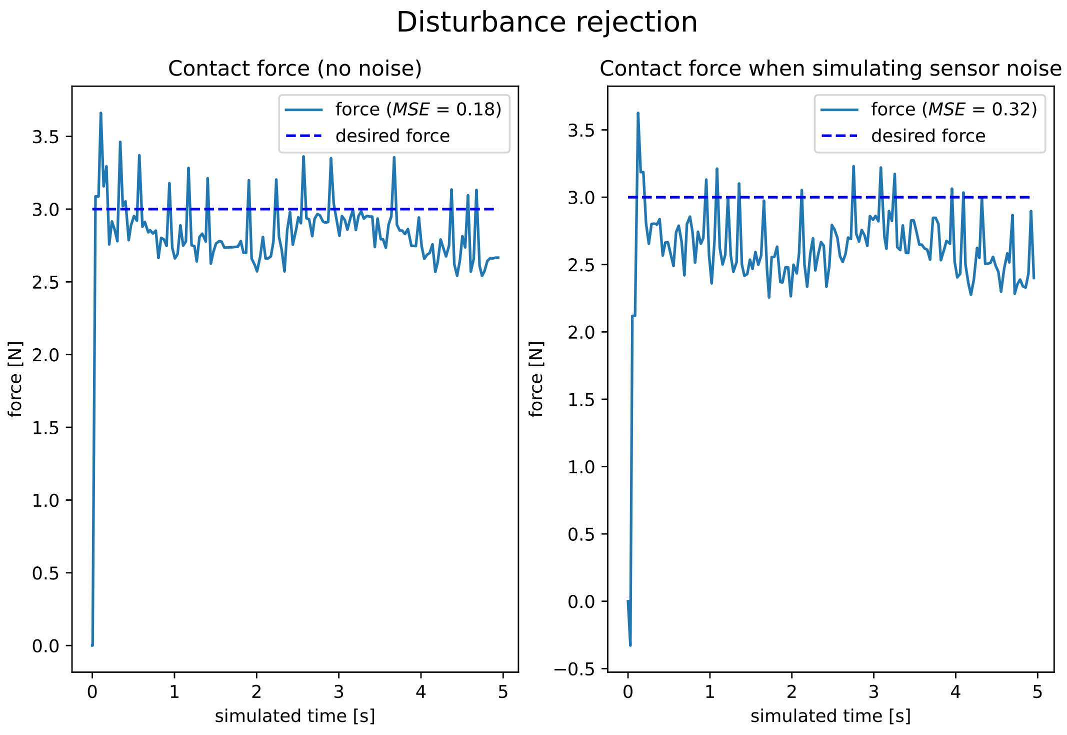


Figure 18: Force tracking performance of the Hybrid Control with and without adding noise. The noise is represented as random samples from a Gaussian distribution with $\mu = 0$ and $\sigma = 0.015|F|$. The simulated noise slightly increases the force tracking error

Admittance Control

The performance of the Admittance Controller in the above-mentioned test scenario is shown in Fig. 19. Furthermore, the figure illustrates how the compliant position is adjusted in the attempt of tracking the desired force of 3 N. When tuned to avoid overshooting at impact, there is a static deviation to the desired force at rest. Moreover, the contact force oscillates when performing the motion trajectory. Regarding motion tracking however, Fig. 19 shows minimal deviation.

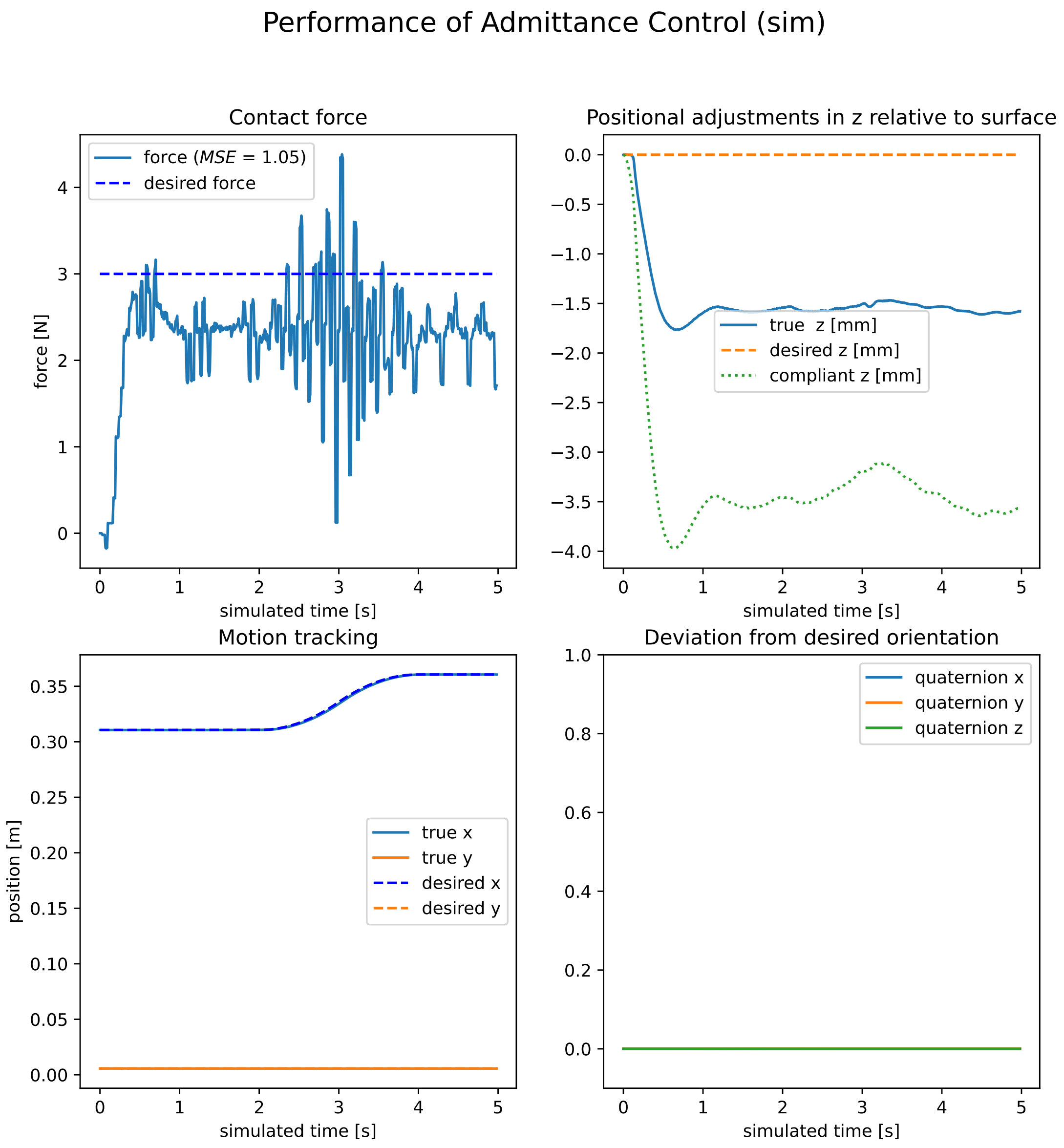


Figure 19: Result from running the Admittance Controller in the homogeneous environment. The recorded data shows that it performs well at motion tracking, but poor at force tracking, especially when in motion. The poor tracking is reflected by the *mean square error* (MSE) of 1.05

Hybrid Control

The performance of the Hybrid Controller in the same environment is shown in Fig. 20. Most notable, it does not suffer from the same oscillations in contact force as the Admittance Controller when moving along the surface. In general, it is doing better at tracking the desired force. However, there still is a noticeable deviation throughout most of the task execution. Moreover, the contact force is generally noisy. In terms of motion tracking, the performance is decent.

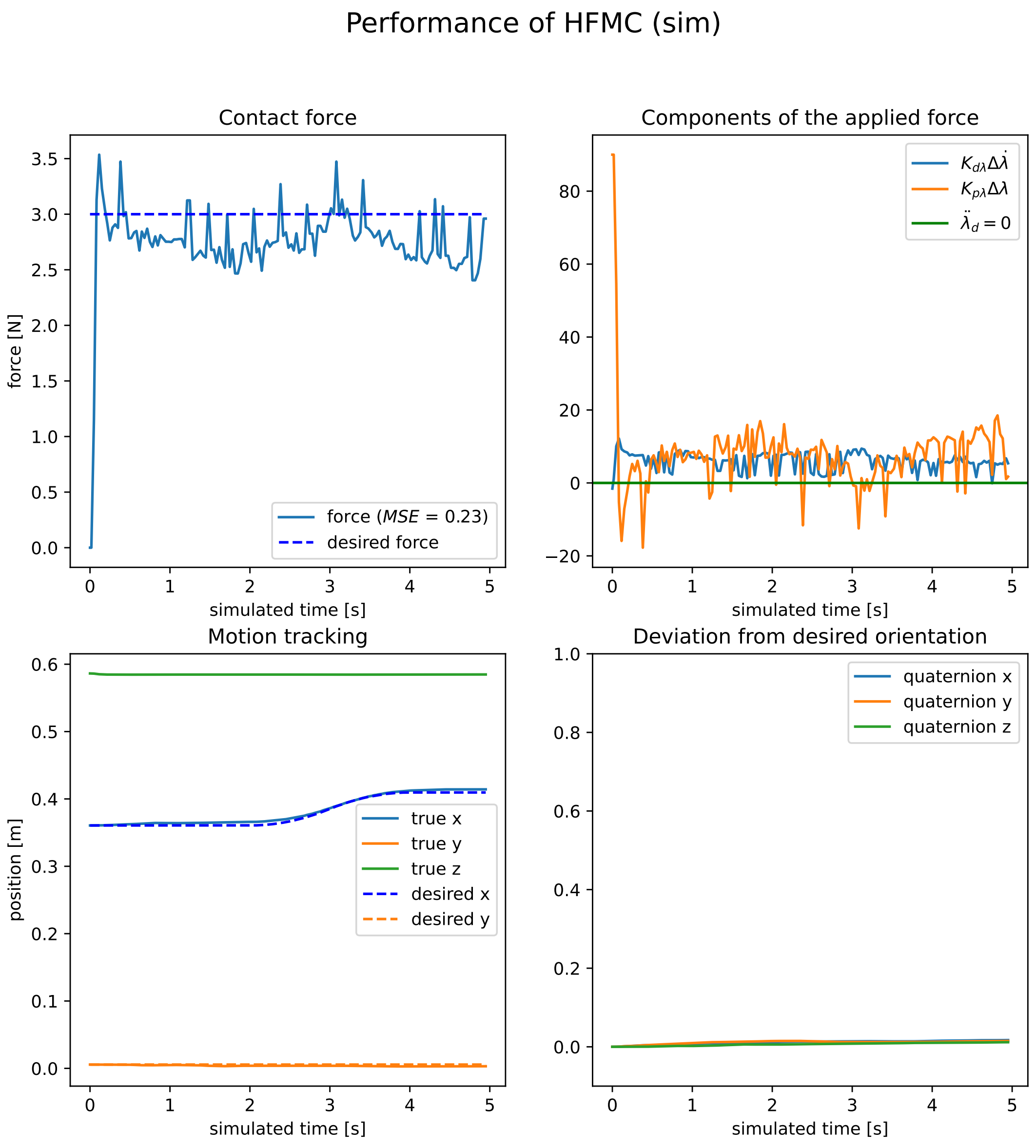


Figure 20: Result from running the Hybrid Control in the homogeneous environment. The recorded data indicate decent performance in terms of both force- and motion tracking. Furthermore, the upper right plot illustrates the subtotals of the force controller's output

Force-based VIC

By repeating the test for the Force-based VIC, the result in Fig. 21 was generated. The figure illustrates the force- and motion tracking along with a plot of the varying damping and stiffness in z. This plot shows how the stiffness is increasing gradually to approach the desired contact force of 3 N. This adaptive trait ensures that it achieves a better force tracking than the two other controllers. Although the magnitude of force is noisy, it manages to keep a contact force in the region of 3 N, both when still and moving in x. However, Fig. 21 suggests poor motion tracking ability. There is a static deviation both in x-position, and in end-effector orientation.

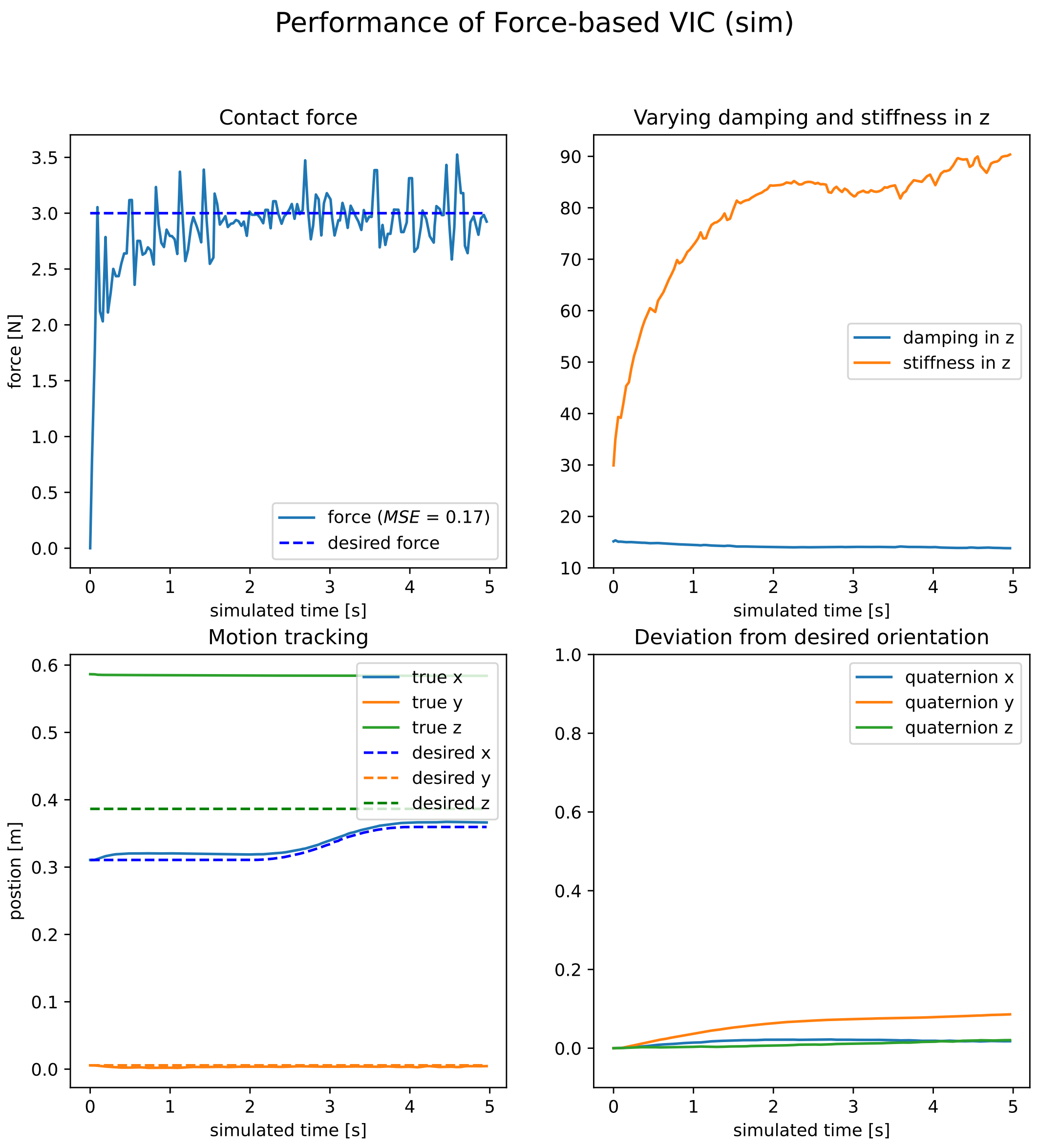


Figure 21: Result of the Force-based VIC in the homogeneous environment. The plots show that it is performing well at force tracking, but poor at motion tracking

5.1.2 Experiments

The experiments are performed using the hardware and software described in Sections 3.1-3.4. The Panda robot itself, and the rest of the experimental setup with the Franka Emika Panda robotic manipulator is shown in Fig. 22. The object subject to interaction is a stiff plastic surface, lubricated to reduce friction. To imitate environmental compliance, the gripper of the manipulator is equipped with a soft foam rubber ball. Similar to the simulated test, the objective of the manipulator is to track a desired vertical contact force of 3 N while performing a motion trajectory. Tests proved that the friction of the experimental interaction is significantly higher than in simulation. When the end-effector first attempts to move along the surface, the static friction gradually increases until it eventually starts to slide, inducing dynamical friction that is lower in magnitude. The complex influence of friction causes irregular end-effector motion, consequently making force tracking very challenging.

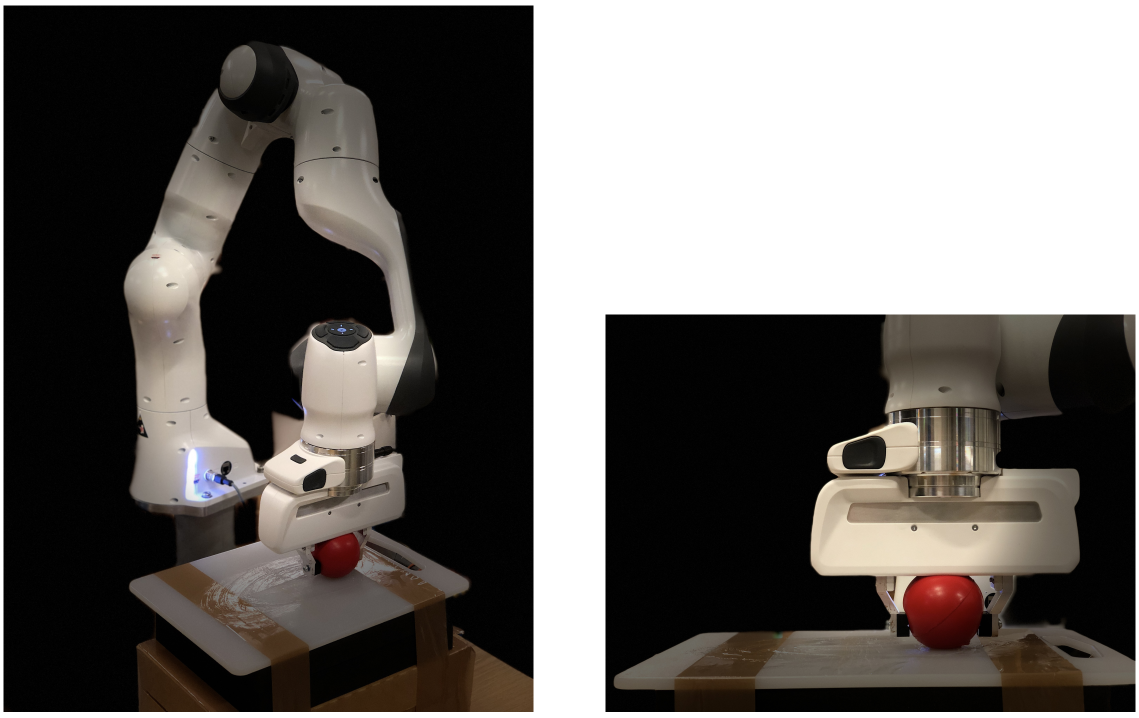

(a) test environment (b) close-up of the test environment

Figure 22: The experimental setup used to test the force controllers. The end-effector is equipped with a foam rubber ball to stimulate environmental compliance. The object subject to interaction is a plastic cutting board, lubricated to reduce friction

When executing the controllers in the experimental setup, the extended FCI proved not to be very real time friendly. Due to slow fetching of states, the controllers cannot maintain control frequencies higher than 30 Hz. This is a significant reduction from the 100 Hz allowed in simulation, on account of the ability to slow down time. The reduction in control frequency proved to be critical for the Admittance Controller, breaking the safety limitations of the experimental setup. Thus, it has only been assessed in simulation. Although not optimal, the

Hybrid Control and the Force-based VIC, both torque controlled, perform better on low control frequencies. Being considered the two most promising candidates, the Admittance Controller's absence in the experimental assessment is not considered decisive. An illustration of the setup's capability to barely maintain a control frequency of 30 Hz is shown in Fig. 23.

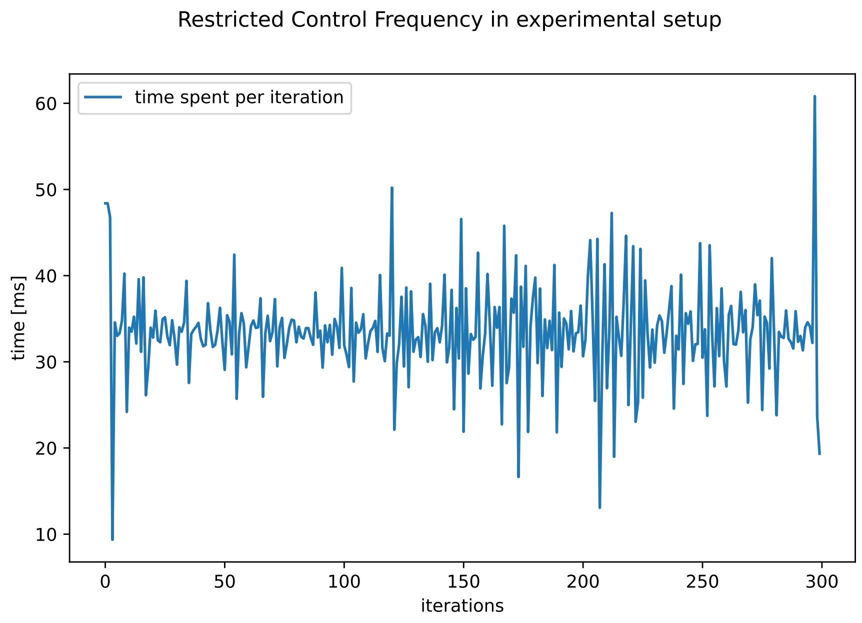


Figure 23: Restricted control frequency in the experimental setup. The controller, in this case the Force-based VIC, is barely maintaining a control frequency of 30 Hz. Moreover, the time-step variance itself can inhibit performance

Hybrid Control

The Hybrid Controller's performance in the above-mentioned test is depicted in Fig. 24. During force tracking, there is initially a substantial overshoot in contact force. The immediate overshoot is followed by a steady state deviation, growing larger when starting its motion trajectory. Regarding motion tracking, the manipulator is unable to smoothly follow the desired trajectory in x-direction. Both the illustration in Fig. 24 and visual impressions suggest that the lagging motion tracking is caused by friction. The friction affects the orientation as well, causing a slight tilt of the end-effector. Additionally, it is fair to assume that the friction and tilt is causing the decrease in force tracking ability, observable halfway out the task-execution. Compared to in simulation, the measured force is a lot less noisy.

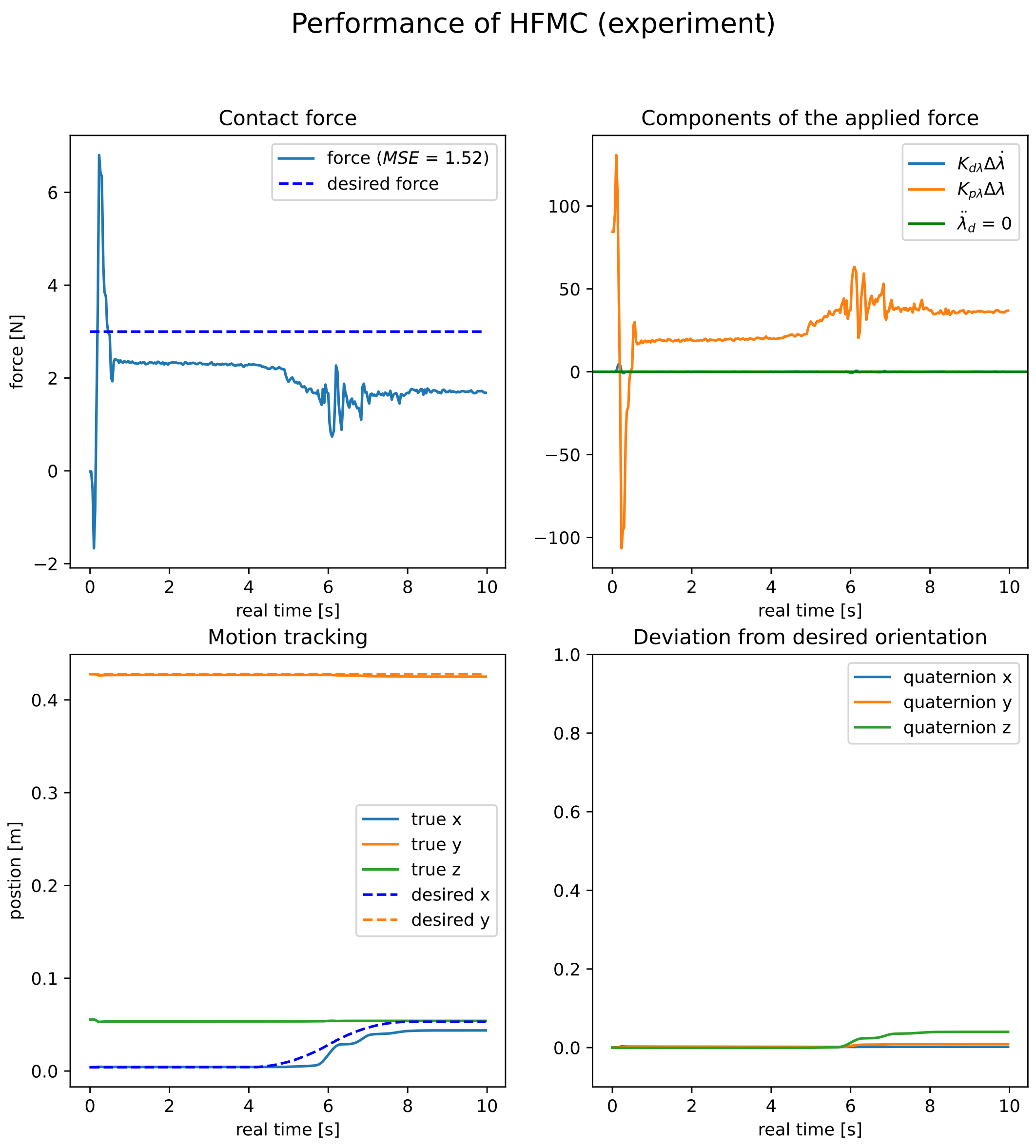


Figure 24: Result of the Hybrid Control in the experimental setup. Most notable, friction appears to restrict the manipulator's ability to track the desired force and motion. Nevertheless, prior the influence of friction, there is an initial force overshoot, followed by a large steady state force tracking error

Force-based VIC

The performance of the Force-based VIC is illustrated in Fig. 25. As in the case of the simulated test, the figure shows how the damping and stiffness are adapted to achieve force regulation. This ability gives the Force-based VIC a superior ability to track the desired force compared to Hybrid Control. However, the friction of the experimental setup leads to poorer force tracking while moving. In terms of motion tracking, it performs poorly, specifically in x direction along the desired trajectory.

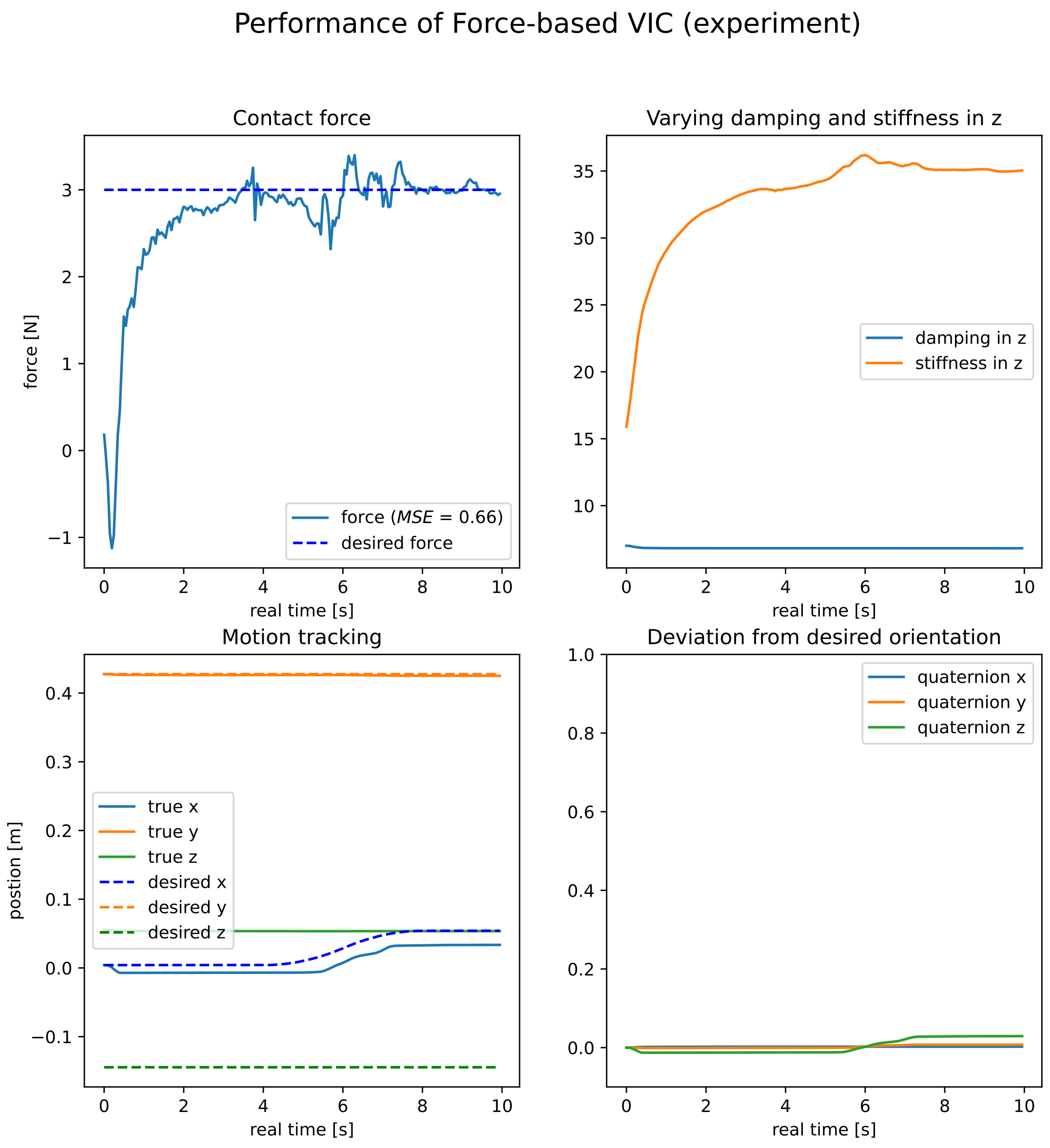


Figure 25: Result of the Force-based VIC in the experimental setup. The controller's force tracking ability is good when standing still but weakens when the end-effector is affected by friction. Furthermore, the motion tracking is poor

5.2 Evaluation of Force Controllers in the Learning Framework

Before initiating the evaluation of the learning-based force controllers, some implementation details shall be reviewed. Especially, the effect of different subsampling intervals and policy models, is to be investigated. Starting with subsampling, a good sampling frequency is one that gives a sufficient model-resolution and action frequency, while still being able to observe the effect of actions between each sample. To find an appropriate sample frequency, tests were carried out, comparing the model variances and accumulated rewards. A compact representation of the findings is presented in Fig. 26.

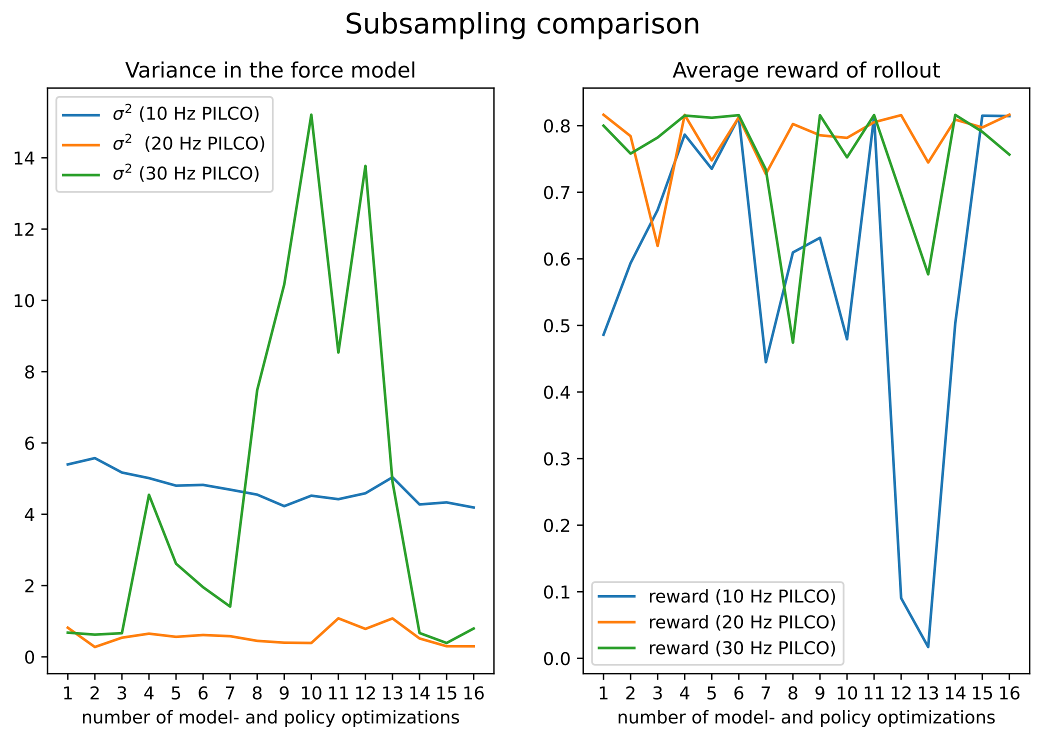


Figure 26: Model- and policy optimization using different model-resolutions. The left plot is showing the evolution of variance in the GP models of contact force, using different sampling frequencies. The right plot is showing the rewards that the resulting linear policy achieves. Using a sampling- and action frequency of 20 Hz appears to be the better choice, granting more precise and stable models and policies

The data was collected from training in simulation, using the Hybrid Control with a 100 Hz control frequency. By testing different subsampling intervals, the sampling- and action frequency was assessed at 10-, 20-, and 30 Hz. Comparing the 10- and 20 Hz implementation, the 20 Hz variant can obtain a much lower variance in the force model, resulting in a policy that grants higher rewards. The result in Fig. 26 suggests that 30 Hz is too frequent. The variance of the force model increases halfway into the optimization process, and the rewards are not as high as for the 20 Hz variant. On the basis of these findings, PILCO is consistently set to run on 20 Hz. For the experimental setup, not able to exceed a 20 Hz control frequency in the learning framework, this corresponds to sample all recorded data. In simulation however, with control frequencies of 100 Hz, only every fifth state vector is sampled.

Regarding the choice of policy model, the performance of the RBF- and linear policy were both assessed for the Hybrid Control in simulation. The outcome of this test is shown in Fig. 27. Although the force profiles of both policy models are noisy, the RBF policy model performs better, obtaining a slightly lower *mean square error* (MSE). Consequently, the RBF policy model is preferred. However, as it was mentioned in Section 4.1, it is not applicable to the experimental setup, due to slow action computations. Thus, only the linear policy model is used on the real robot.

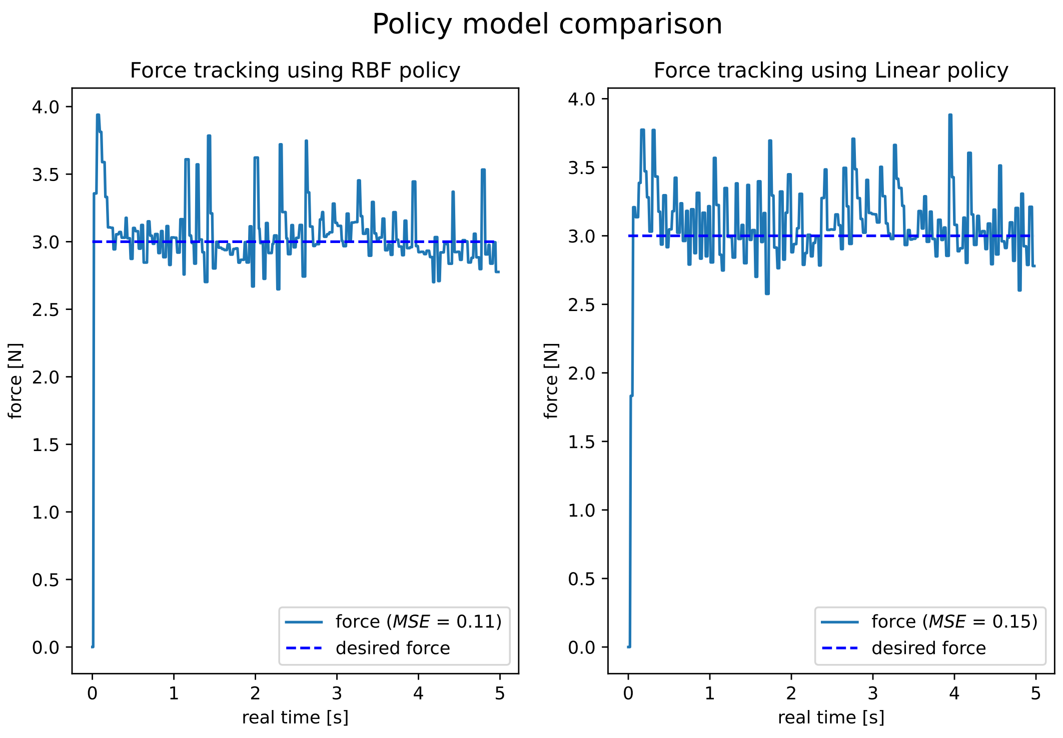


Figure 27: The force tracking performance of the Hybrid Control using both a RBF- and a linear policy model. The result suggest that the RBF policy model enables better force tracking, obtaining a slightly lower MSE

Emphasizing learning efficiency, no controller has exceeded a total of 20 rollouts of training in the results presented. As each rollout has a duration of 5 seconds, the total runtime of the robot is no more than 100 seconds.

5.2.1 Simulation

Now, testing the controllers in the learning framework, they are assessed in the same simulated test scenario as in Section 5.1.1, performing a motion and force trajectory in the environment depicted in Fig. 17.

Learning-based Admittance Control

The result of the Admittance Controller with learning in the homogeneous environment is shown in Fig. 28. The figure compares the force tracking ability of the controller with- and without learning. Furthermore, it shows how the damping and stiffness parameters are changing throughout the execution of the task, according to the learned policy. As it might be expected, despite introducing learning, the force tracking is still poor. Moreover, while in motion, the contact force fluctuates. By inspecting the calculated MSE in Fig. 28, the learning suggest an insignificant improvement in force tracking performance.

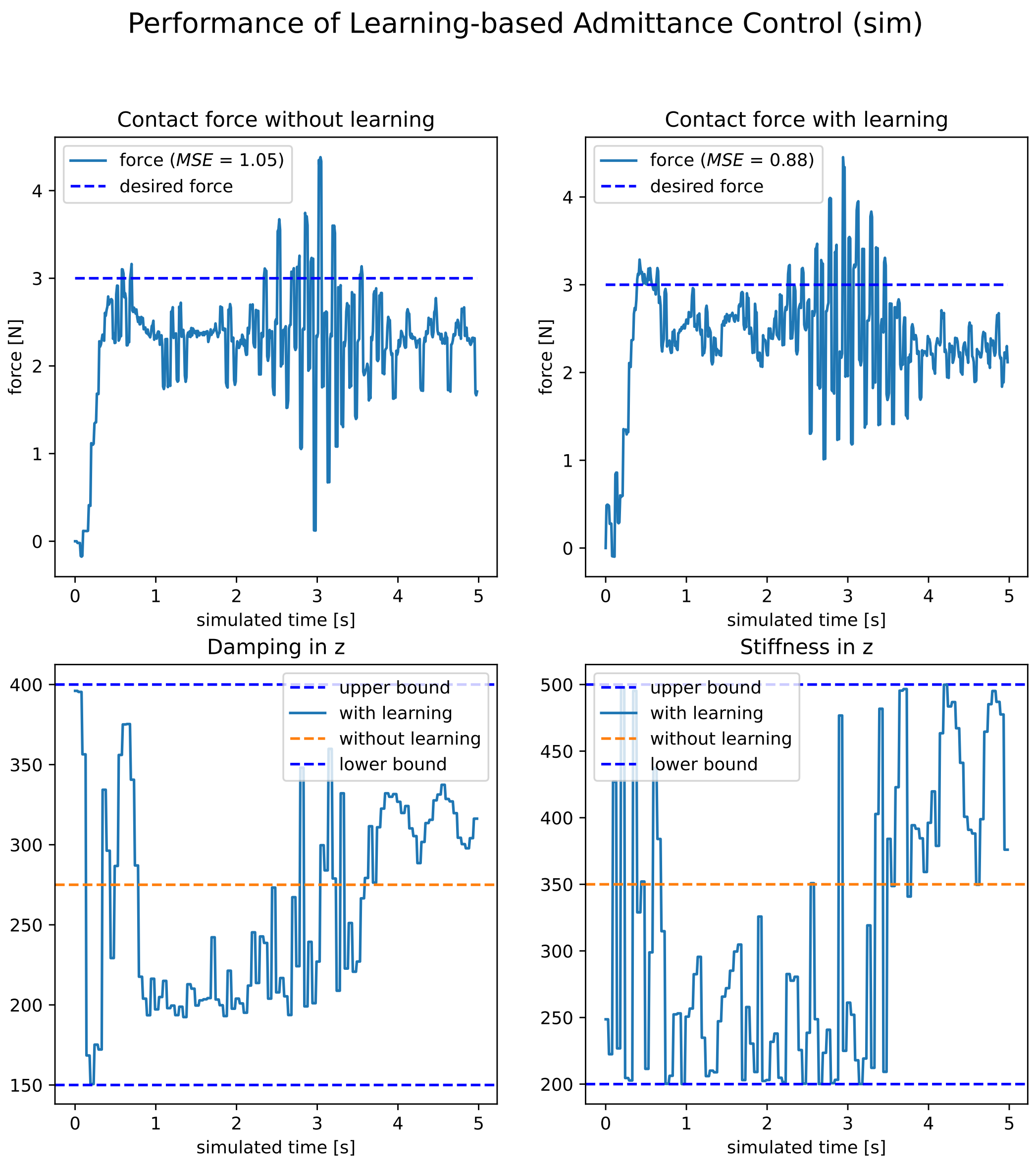


Figure 28: Result of the learning-based Admittance Controller in the homogeneous environment of Fig. 17. The learned impedance strategy, illustrated in the lower plots, does not significantly improve the force tracking performance

Learning-based Hybrid Control

Repeating the test for the learning-based Hybrid Control resulted in the plots in Fig. 29. The depicted impedance strategy improves force tracking, reducing the MSE from 0.23 to 0.11. Although the measured force is noisier with learning, the values are now centered around the desired value of 3 N.

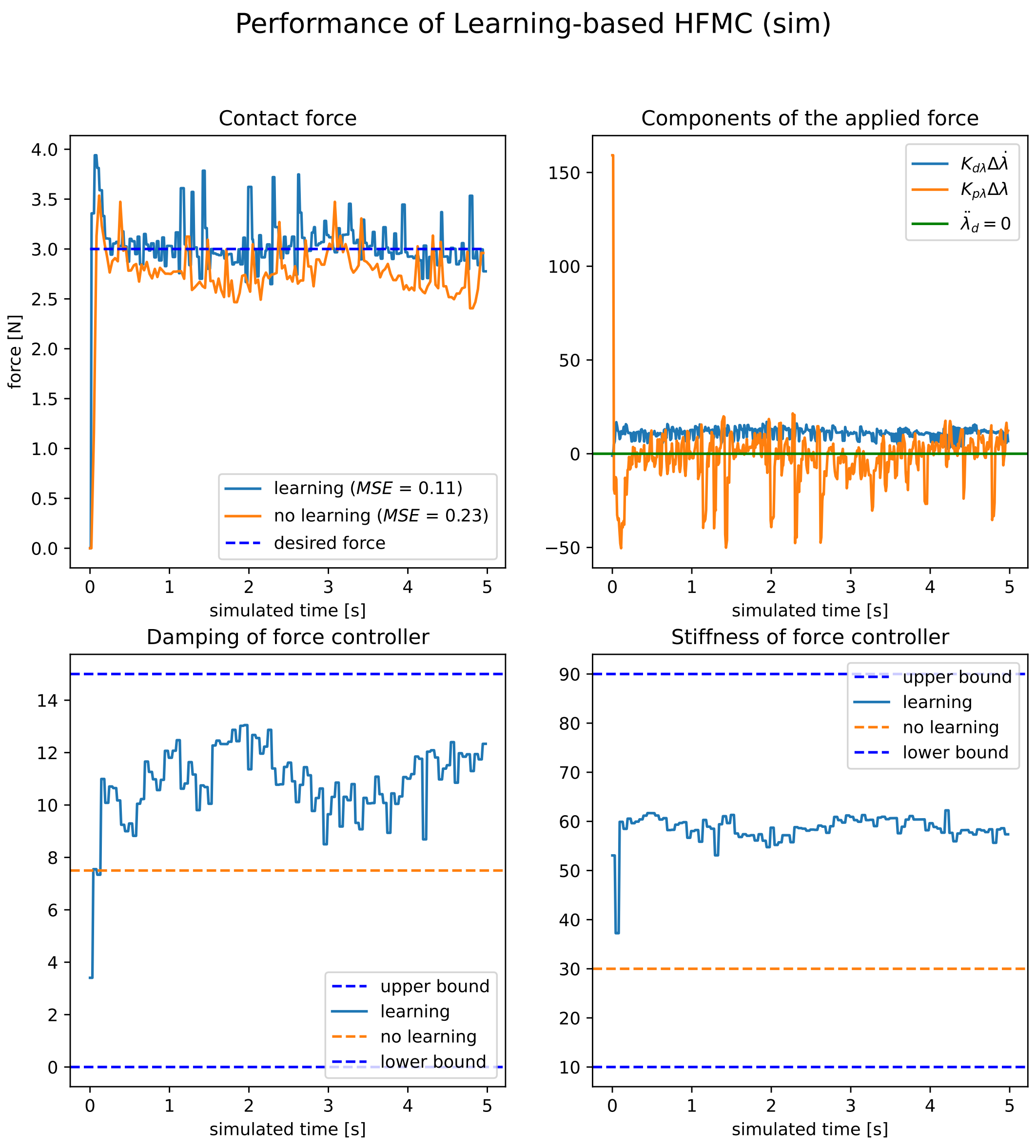


Figure 29: Result of the learning-based Hybrid Control in the homogeneous environment of Fig. 17. The contact force comparison, upper left plot, suggests that the learned impedance strategy improves the force tracking ability, cutting the MSE in half

Force-based VIC with learning

The outcome of introducing learning in the Force-based VIC is presented in Fig. 30. Most notable, the learned strategy maximizes the rate of stiffness-adaptability in the first phase of the task-execution. This results in a faster convergence to the desired contact force of 3 N. For the rest of the trial, from approximately 1.5 seconds, learning does not improve force tracking. On the contrary, the MSE in this interval is marginally increased from 0.035 to 0.038. However, due to faster convergence, the total MSE is reduced from 0.23 to 0.16.

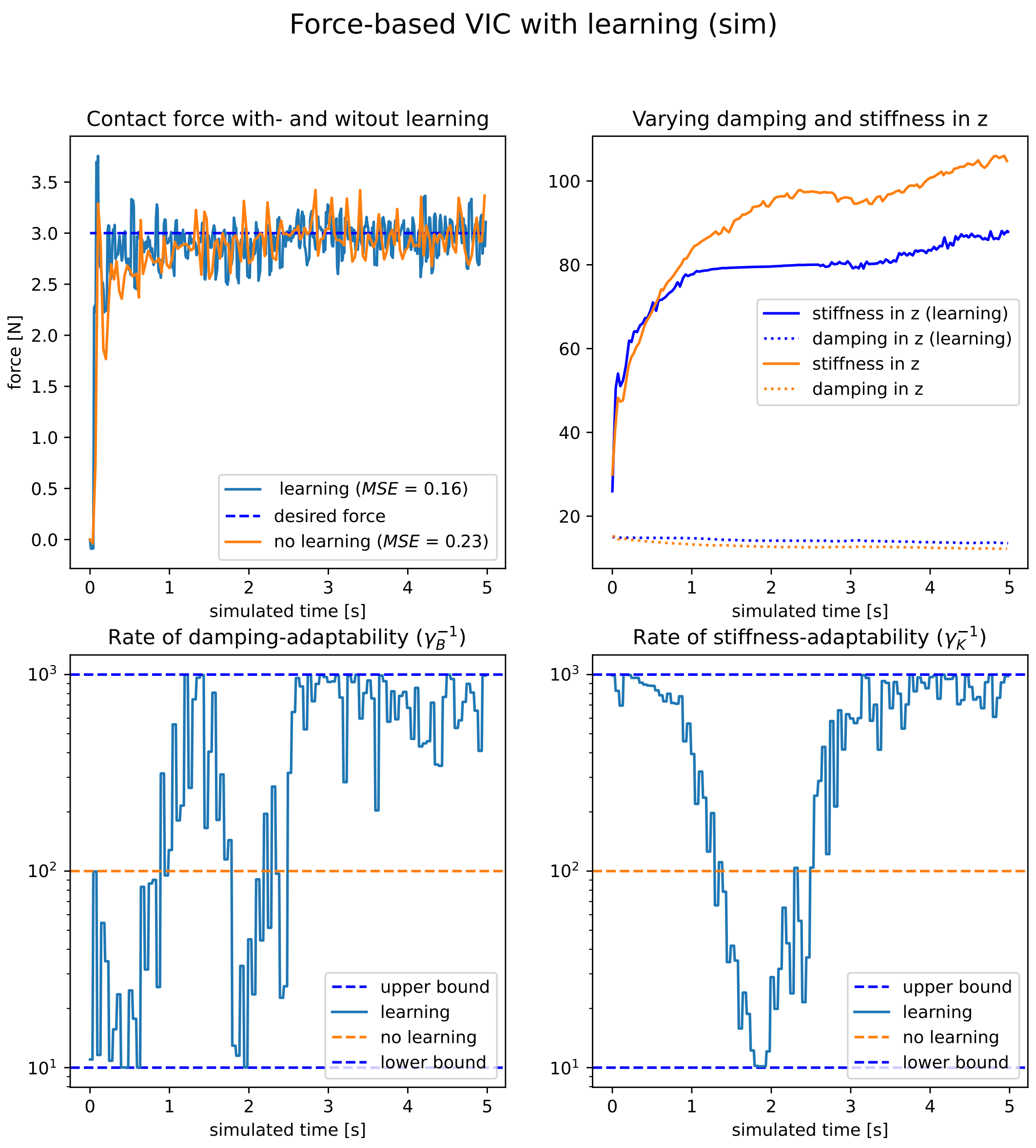


Figure 30: Result of introducing learning in the Force-based VIC, and testing it in the homogeneous environment of Fig. 17. Faster convergence to the desired contact force reduces the MSE from 0.23 to 0.16

5.2.2 Experiments

Next, the learning-based controllers will be assessed in the same experimental setup as their learning-free counterparts in Section 5.1.2, performing a motion- and force trajectory on the surface shown in Fig. 22.

Learning-based Hybrid Control

The performance of the learning-based Hybrid Control in the experimental setup is shown in Fig. 31. Like its static counterpart, it experiences an initial force overshoot. Then, in the stationary phase, it is showing an improved ability to track the desired force, reducing the stationary deviation. However, when starting the motion trajectory, encountering friction and end-effector tilt, the force tracking ability is reduced. The learning-based controller ends up with the same MSE as the one with static stiffness and damping parameters.

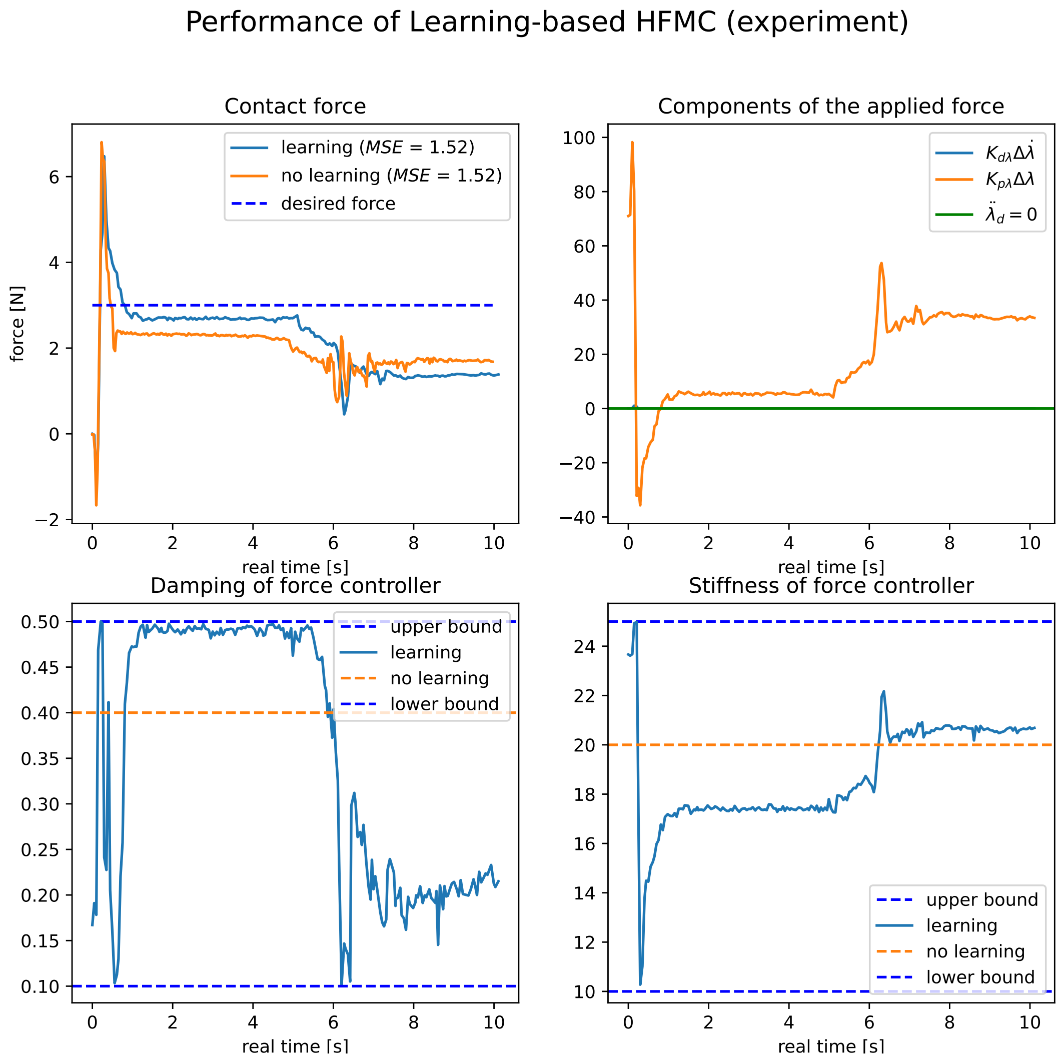


Figure 31: Result of the Hybrid Control with learning in the experimental setup. Despite starting off with a better force tracking ability than its static-impedance counterpart, it has the same total MSE, appearing more affected by friction and tilt

Force-based VIC with learning

The learning Force-based VIC's performance in the same test is depicted in Fig. 32. The initial force tracking is worse than that of the learning-free variant, evidently increasing the deviation from the desired force. As a result, it has a higher total MSE than the learning-free VIC. However, disregarding the first second, the learning-based controller achieves a MSE of only 0.0046, substantially lower than the 0.059 MSE of the learning-free controller in the same interval. The learning-based force tracking leads to faster convergence, and appears more robust against friction.

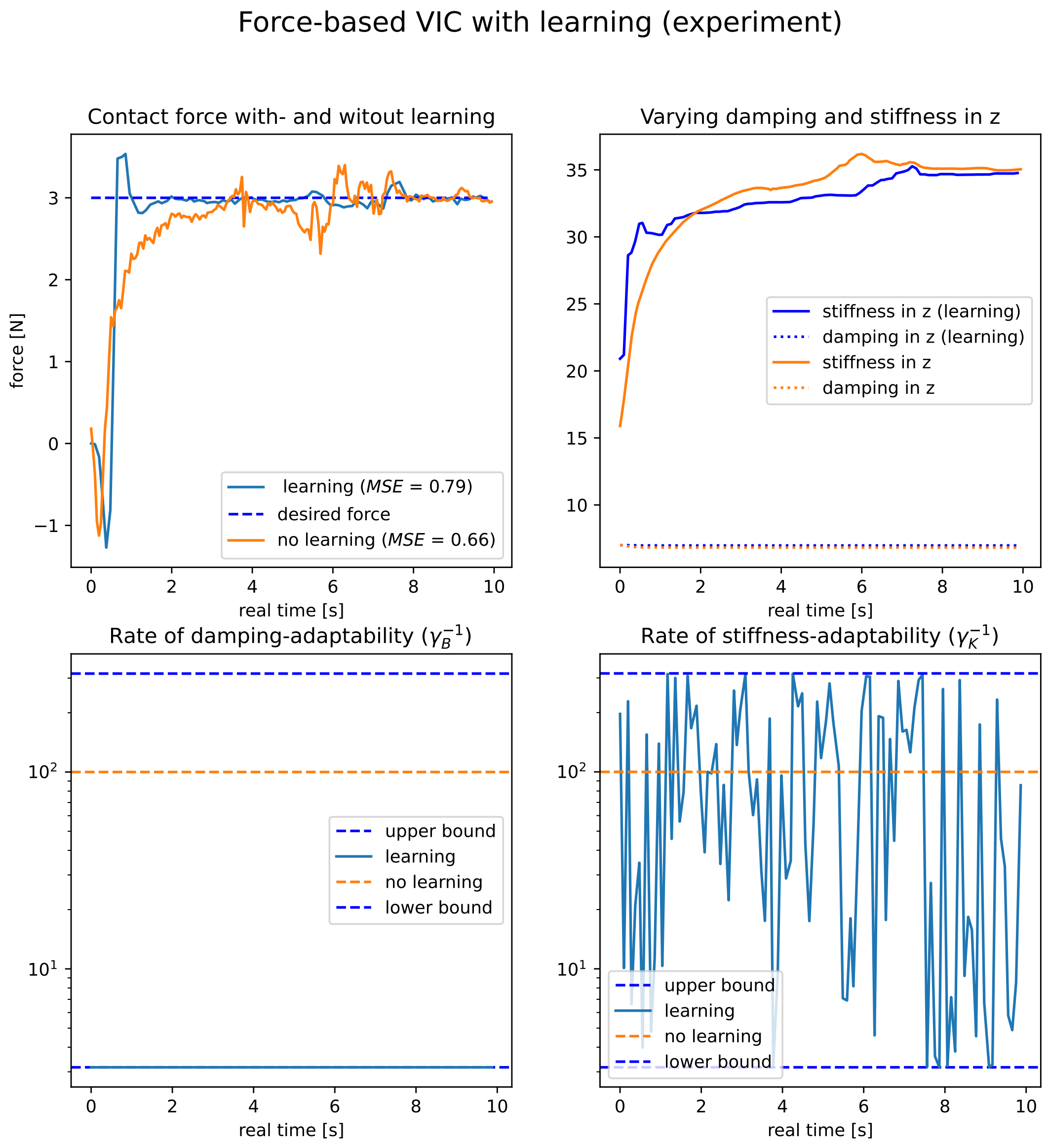


Figure 32: Result of the Force-based VIC with learning in the experimental setup. Due to bad initial tracking, it ends up with a higher MSE than its learning-free counterpart. However, disregarding the initial second, the learning-based controller achieves a MSE of only 0.0046, substantially lower than the 0.059 MSE of the learning-free implementation

5.3 Introducing Supplemental Transition Models

To be able to perform force tracking on non-homogeneous surfaces, encountering different environmental impedance-characteristics, it is not sufficient to learn only one transition model. In this section, supplemental models are introduced. In theory, by dedicating a model to each distinct part of the environment, the force tracking ability will improve. This hypothesis is evaluated both in a simulated test scenario and in an experiment using the Hybrid Control.

5.3.1 Simulation

In the simulated setup, the end-effector is to interact with a more complex surface than before. The object, illustrated in Fig. 33, consists of parts with different stiffness and damping. The green part of the surface still has a stiffness coefficient of $kp = 5.000$, and damping coefficient $kd = 3$, while the new red region has stiffness and damping coefficients of 10.000 and 1 respectively. As before, the object of the controller is to first establish a vertical contact force of 3 N, before moving 5 cm forward, maintaining both a contact force of 3 N and a neutral orientation. The forward movement in x will now have the controller experiencing an abrupt change in environmental stiffness and damping, constructing a challenging task.

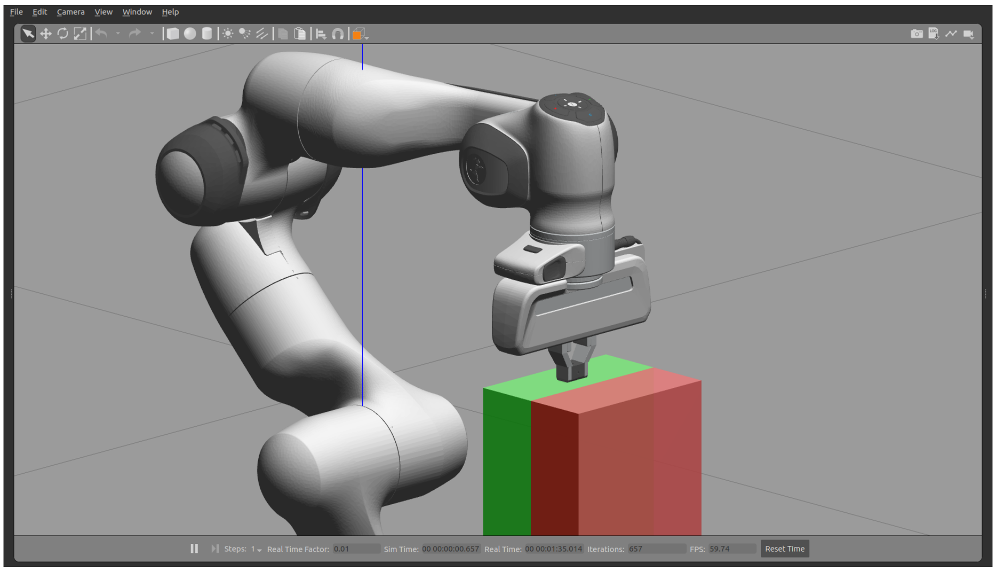


Figure 33: The setup used to test the introduction of supplemental transition models in simulation. The green part of the object has a stiffness coefficient $kp = 5.000$, and a damping coefficient $kd = 3$. The red area has stiffness and damping coefficients of 10.000 and 1 respectively

In such a non-homogeneous environment, it is unfeasible to fit the transition dynamics into one single model. This is illustrated in Fig. 34. The variance in the force model increases substantially, and is thus unable to make reasonable predictions, resulting in a fluctuating policy.

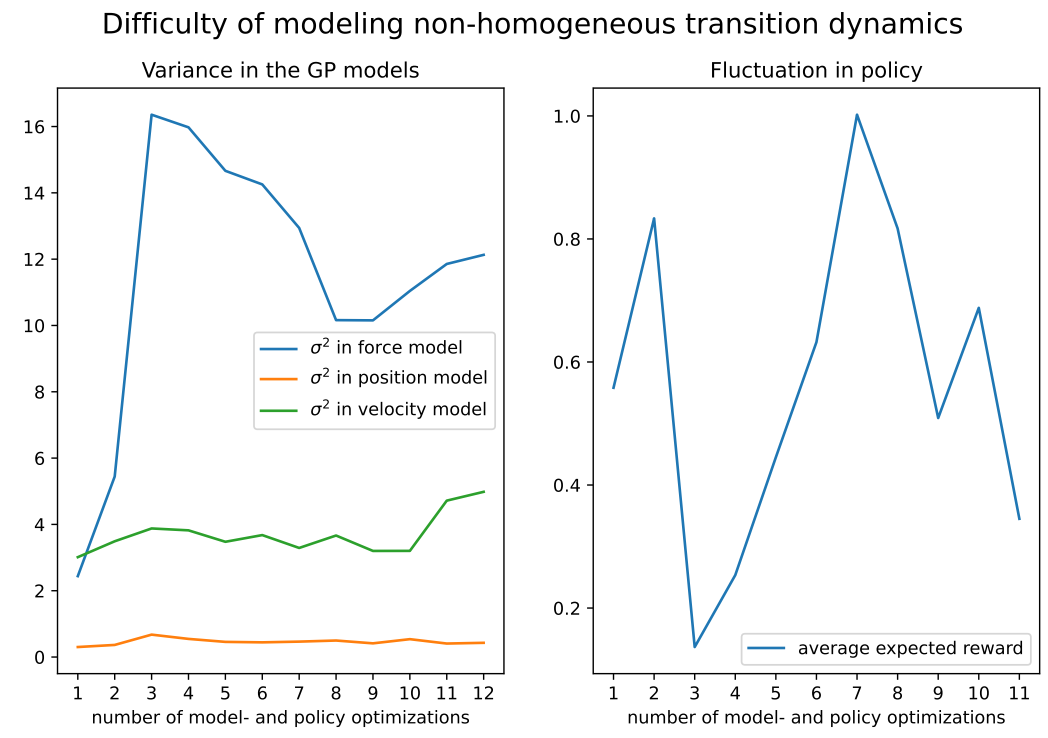


Figure 34: The outcome when GPs' are assigned to non-homogeneous transition dynamics. The left plot shows how the variance of the force model increases significantly. The right plot does not indicate any policy-convergence

Now, dedicating models to distinct parts of the environment, the controller is to alternate between separate policies. In this specific case, the switch is initiated on the basis of the end-effector's x-coordinate, knowing the location of the dividing line shown in Fig. 33.

To see the effect additional models and policies have on performance, three distinct modelling-strategies are tested in the dual environment: 1) learning and using one policy, 2) learning and using two policies, 3) learning and using three policies. The outcome is presented in Fig. 35. For comparative reasons, also the performance of static-impedance is included.

Starting with the static controller, it is behaving similar in the soft part of the environment as it did in Section 5.1.1, showing a decent force tracking ability. However, when entering the part of the environment with higher stiffness, there is a force overshoot of 1.5 N.

The learning-based variant using only one policy is not able to improve force tracking. On the contrary, the MSE is doubled. There are substantial force overshoots when making contact with both parts of the environment, and the contact force is generally noisy.

The controller using a dedicated model and policy for each part of the environment on the other hand, improves force tracking, although only marginally. At steady state it achieves better force tracking than the previous candidates. Nevertheless, the measured forces are still noisy, and there is still a significant overshoot when encountering the stiffer surface.

For the last variant, an additional third model and policy is assigned to the border between the soft and stiff area. The result in Fig. 35 shows that the force tracking is significantly improved in part 1 and 3, where the respective models and policies are designated to homogeneous parts of the environment. However, in the transition between the soft and stiff environment, the force tracking is very poor. When the manipulator makes full contact with the stiffer surface, there is a 180 % overshoot in contact force, contributing to a substantial MSE of 2.2. In fact, inspection of the applied impedance strategy tells that the stiffness is increased in this middle part.

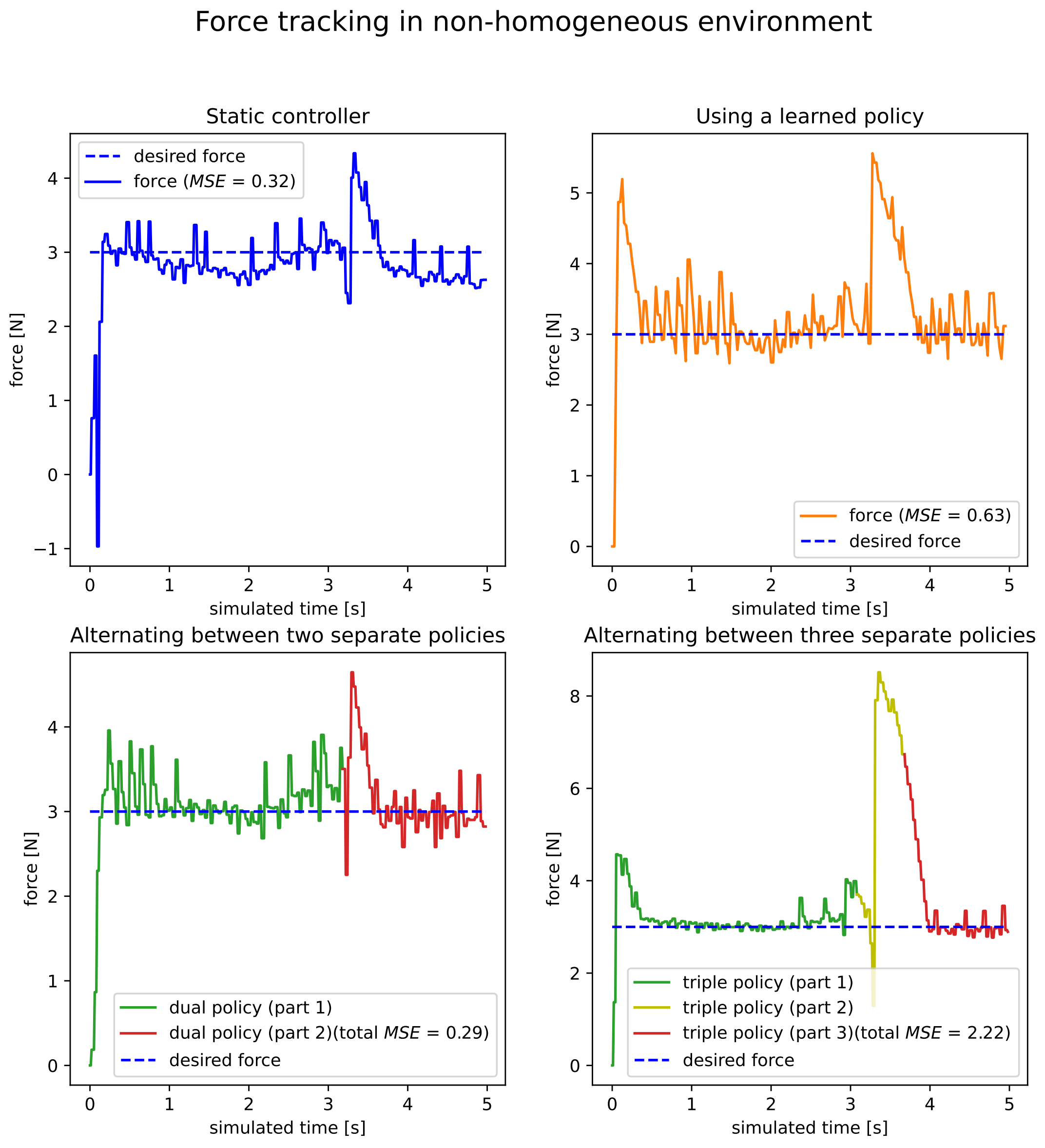


Figure 35: Results of the Hybrid Control in the dual environment depicted in Fig. 33, utilizing 0 to 3 policies. Only the two-policy variant is able to improve force tracking performance, and it does so only marginally. In sum, the results show that the GP models are unable to accurately describe the dynamics of the transition between the two distinct environmental impedances

5.3.2 Experiments

As one model only can represent a single transition dynamic, it is expected to do poorly if the dynamics changes during the execution of a task. It is a reasonable assumption that friction and end-effector tilt had such an impact on the learning-based experiments in Section 5.2.2. Especially the Hybrid Control looks affected by this in Fig. 31. As the transition dynamics changes when the manipulator starts performing its motion trajectory, it is presumed that separate models will generate better force-predictions, consequently leading to a better force tracking ability. The theory was put to the test, resulting in the plots in Fig. 36. Starting with the initial contact force, there is almost a 2 N force overshoot. Compared to the previous results however, this is a noticeable improvement. In the first stationary phase, before moving along the surface, it is performing exactly as the one-model variant, having a stationary deviation of approximately 0.3 N. However, when starting the movement in x, where friction and tilt changes the transition dynamics, they respond differently. The implementation utilizing a dual policy is much less affected by the external influence, enabling a better force tracking ability. By dedicating a model to each distinct transition dynamic, the MSE is reduced from 1.5 to 0.60.

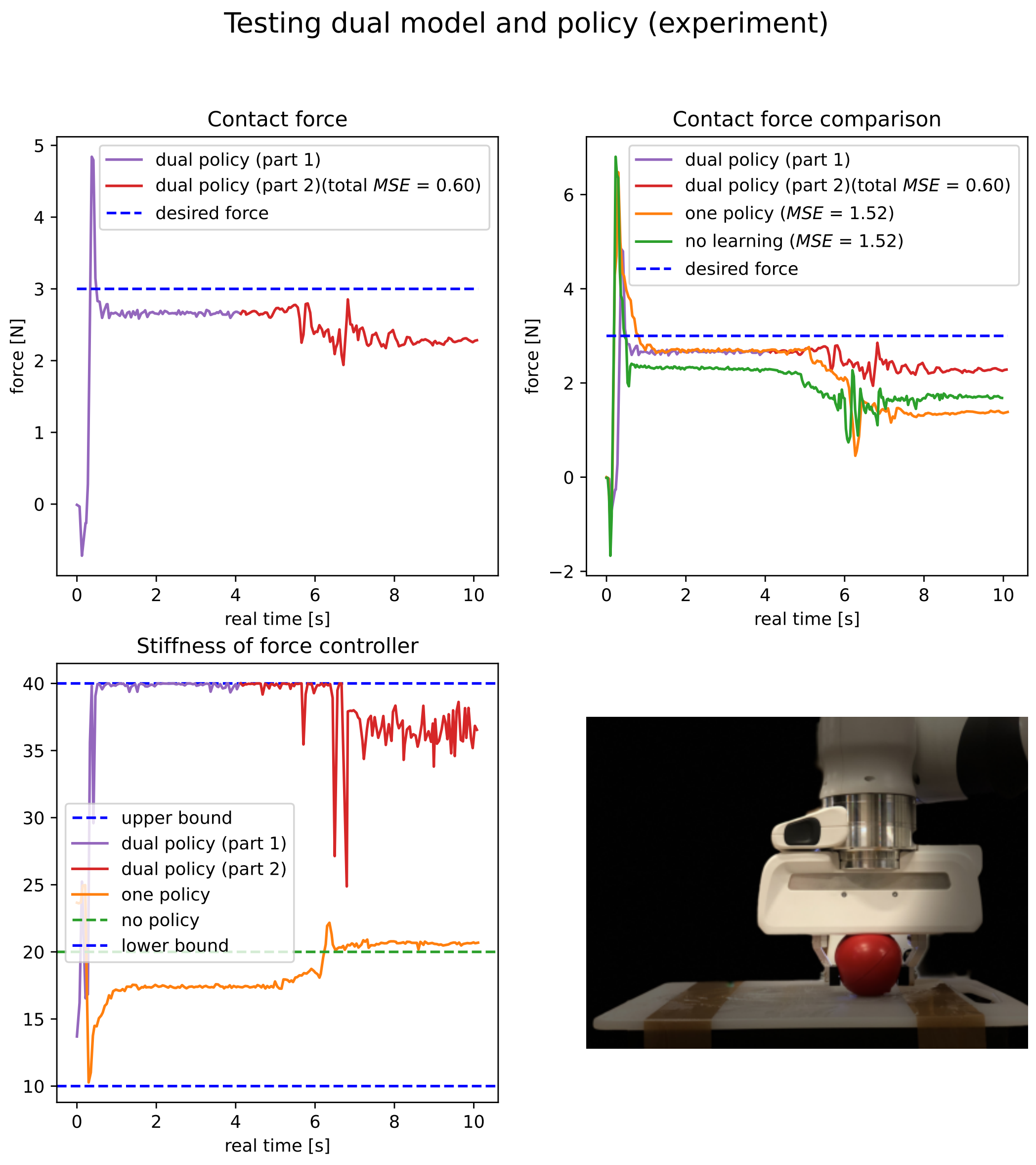


Figure 36: Result of introducing an additional model, dedicated to represent the transition dynamics in the presence of increasing friction and tilt. The extension improves force tracking, reducing the MSE from 1.5 to 0.60

6 Discussion

In this section, the results and findings are up for discussion. First, the low-level force controllers are reviewed, both with respect to the task at hand and the learning-based framework. Then, the software facilitating the implemented framework is discussed. Subsequently, experiences from using PILCO for learning compliant control are reviewed. Following, the findings are viewed in the context of robotic ultrasound examination, evaluating the learning-based controllers application potential. Finally, future directions are up for discussion.

6.1 Overall Force Controller Performance

Admittance Control

The overall results of the Admittance Controller suggest that it achieves good motion tracking at the expense of force tracking. This is a fair assumption as the controller only offer compliance in the vertical z-plane. With no compliance related to its orientation, it slips in the direction of movement when performing the motion trajectory, causing oscillations in the contact force. This issue remains a problem when introducing learning. The force oscillations cause great variance in the models' predictions, and thus leads to bad policies. The results suggests that the PILCO-algorithm has been assigned to an unfeasible task. In summary, due to the limited degrees of compliance, this particular admittance controller does not appear to be a good solution for contact trajectories.

Hybrid Control

Both the results from simulation and experiment, shown in Fig. 20 and Fig. 24, suggest that the Hybrid Control is better suited for the task at hand. Nevertheless, its performance is far from perfect. In terms of force tracking, especially the experimental result shows a lacking ability to avoid static deviation. Furthermore, there is a significant initial force overshoot, but this is not currently of particular interest, being more related to tuning. More interesting, the static deviation in force tracking is increasing when the end-effector starts moving along the surface. The fluctuation in contact force in Fig. 24, observable from the sixth to eight second, appears to correlate with the lagging motion trajectory and the sudden end-effector tilt. Overall, the results suggests that friction, experienced when performing the motion trajectory, complicate the tracking of both force and motion.

In the simulated, low friction environment, the introduction of learning improves the force tracking ability of the Hybrid Control. As Fig. 29 shows, the MSE is reduced from 0.23 to 0.11. Nevertheless, the learned impedance strategy brought with it a slight increase in initial force overshoot, and the measured force is somewhat noisier. Inspecting the varying stiffness, there is a noticeable decrease in the initial impact-phase, looking like an attempt to reduce impact forces. However, with a perfect model and policy, the drop would probably be even bigger and occur slightly earlier, avoiding an overshoot altogether. The consistent spikes in experienced contact force may very well be a consequence of sensor noise, leading to the consistent small adjustments of damping and stiffness observable in Fig. 29.

In the experimental setup, the introduction of learning did not have the same impact. Although the force profiles in Fig. 31 look different, they have the same considerable MSE of 1.5. In the first phase of the task-execution, the learning-based controller is performing better than its learning-free counterpart. However, when friction and tilt start affecting the system, the force tracking ability drops significantly. This happens as the learned model is inaccurate for this phase of the task-execution, suggesting that the transition dynamics of the system changes when friction and tilt are occurring. This is a problem, as one model is unable to describe two distinct transition dynamics. In Section 5.3.2 however, it was demonstrated that the performance could be improved by introducing an additional model, designated to represent the transition dynamics in the presence of friction and tilt. The extension improved force tracking, reducing the MSE from 1.5 to 0.60.

Force-based VIC

The simulated and experimental results of the Force-based VIC, depicted in Fig. 21 and 25, showcases the strength of variable impedance. Unlike the Hybrid Control, there is no static deviation to the desired contact force. Both above figures clearly show how the stiffness is increased to do so. Furthermore, the adaptive ability allows it to start off with a high compliance in z, avoiding any initial force overshoot. As it was the case for the Hybrid Control, the measured contact forces in simulation are quite noisy. Since the contact force in the experiment is much smoother, the noise is likely related to the force sensor of the simulator. Nevertheless, when performing the motion trajectory, the experimental results show fluctuation in force as well. Like in the case of the Hybrid Control, this appears to be caused by friction between the end-effector and the surface of interaction. Additionally, this disturbance inhibits motion tracking, observable in Fig. 25 as slightly lagging x-motion and end-effector tilt. Regarding motion tracking, unlike the Hybrid Control, the Force-based VIC is experiencing static deviations throughout the task. When tuning the controllers, it appeared more challenging to achieve good motion tracking of the Force-based VIC.

When introducing learning in the Force-based VIC, the most notable difference in the simulated test is how fast it converges to the desired contact force. The plotted rate of stiffness-adaptability in Fig. 30 is showing that the fast convergence is a result of a clear strategy. The high initial adaptive rate is decisive in reducing the MSE from 0.23 to 0.16. In fact, disregarding the initial phase, the MSE is increased when introducing learning. In regards to the noisy force profiles in Fig. 30, they appear to be a result of noisy measurements from the simulated force sensor. As the noise propagates in the dynamics model, the ability to improve performance is inhibited.

Interestingly, when testing the Force-based VIC with learning in experiment, the total MSE is increased. However, closer inspection proved that the rise was solely caused by the initial drop in contact-force, shown in Fig. 32. As the end-effector is starting in contact with the surface, the drop in contact-force is likely caused by the high initial z-compliance, causing it to lose the initial contact. Nevertheless, disregarding the initial second, the introduction of learning is improving the force tracking significantly, reducing the MSE from 0.059 to 0.0046. Compared to the implementation with a fixed impedance law, the contact force converges faster to the desired value of 3 N, causing a minor overshoot doing so. After converging to the reference contact-force, the learning-based controller achieves better force tracking. Most interestingly, the learning-based controller is able to compensate for the effect of friction. Fig. 32 shows how the rate of stiffness-adaptability (γ_K^{-1}) is lowered for that purpose. By lowering the adaptive rate in the interval where friction is applied, from approximately 5.5- to 6 seconds, the force tracing is significantly improved, reducing the MSE from 0.11 to 0.0032. This property makes it more robust to environmental changes. Furthermore, the underlying adaptive law makes the learning Force-based VIC less prone to modelling errors than the Hybrid Control. In fact, the underlying adaptive mechanism will provide regulation regardless of the learned strategy. However, the strategy will determine the extent of the regulation, being responsible for optimizing the force tracking ability.

Method of evaluation

Due to the way all reward- and policy related functions are implemented in [46], the framework does not support varying target states. Thus, the desired contact force was set to be constant for the learning-based controllers. For comparative reasons however, the learning-free controllers where subject to a static desired contact force of 3 N as well. Consequently, there is initially a substantial error in force tracking. In a more elegant solution, the desired force would start off at 0 N before gradually rising to 3 N. From experience with the static controllers, such tests significantly reduce the risk of force overshoots. Especially the Hybrid Control would benefit from a smoother increase of desired contact force. Furthermore, the forced initial tracking error of 3 N is causing the MSE values to be less conclusive. For example, the Force-based VIC with learning ended up with a MSE of 0.79, despite having a MSE of only 0.0046 for 90 % of the test. Thus, it can be argued that the performance after achieving a stable contact is more conclusive in terms of evaluating the benefit of learning. Nevertheless, the initial force error poses as an interesting challenge for the learning-based controllers, putting their adaptive abilities to the test.

In terms of the respective force controller's performance, both in terms of motion- and force tracking, one should not underestimate the impact of parameter-tuning. Even for the learning-based controllers, the general performance is very dependent on the parameters that are fixed. However, it is believed that the results are representative for its purpose, showcasing characteristic properties, pros and cons.

Regarding the results from simulation, they are considered valuable for evaluating the performance of the controllers. Nevertheless, it must be specified that the environmental stiffness, damping and friction are not constituting a true representation of any real system. For example, the surface friction of the object in simulation is significantly lower than the one experienced in the experimental setup. However, the simulated tests are thus giving an advantageous indication of what behaviour to expect in a low-friction environment. After all, the high static- and dynamic friction proved to be a complicating factor in the experimental setup. In fact, if the experiments were to be repeated, an attempt would be made to reduce the friction of the interaction, providing a smoother motion trajectory. Regarding the dual environment used in simulation (Fig. 33), it is believed to give a good indication of what behaviour to expect when interacting with a more challenging surface. Consequently it has provided some valuable insight, showcasing the limitations of the PILCO-algorithm.

6.2 Software

Extended FCI framework

The extended FCI framework, consisting of FCI and the extensions described in Section 3.4, has been crucial for the implementation of the controllers. By offering predefined functions to fetch states and send commands, the Python interface is making this process easier and more intuitive. Furthermore, the level of abstraction with respect to robot communication is making it easier for those not familiar with ROS. Additionally, the Gazebo-based simulator is a very helpful tool when developing code for the Panda robot. Particularly, the direct *sim-to-real* transfer of code is very convenient as it allows straightforward testing in simulation prior to robot-deployment.

However, despite its indisputable convenience, the extended FCI has its limitations. The lengthy state-retrieval is a significant one, limiting the maximum control frequency to modest 30 Hz in the experimental setup. This proved to be critical for the Admittance Controller as the inner loop position controller was unfit to provide stable behaviour running such a low control frequency. The Hybrid Control and the Force-based VIC handled the low control frequency better. Nevertheless, their performance is restrained. As Fig. 23 shows, there is a significant variance in the controllers' time-steps when running 30 Hz, which can have unfortunate consequences on performance. The varying size of the time-steps is critical when introducing learning. As the model of the transition dynamics expect deterministic time-steps, these variations are causing the model to perceive inconsistent system-dynamics. Thus, the model variance increases, consequently leading to a less accurate system-representation. In future work, speeding up the state-retrievals should be prioritized. As the issue appears to be a consequence of overloading the python wrapper, it should preferable be circumvented. The speed can potentially be increased by implementing the python controllers as *rospy* nodes, communicating with the robot through *roscpp*, a C++ implementation of ROS providing functionality without delay and with minimal stochasticity.

PILCO algorithm

The PILCO-algorithm package provided by the Github repository `https://github.com/nrontsis/PILCO` [46] is utilized in the implementation of learning-based force controllers. After having made the robot-environment compatible with the PILCO package by setting it up as a Gym environment, reachable through Python 2 gateways, the process of extracting sampled data and using it for learning was fairly intuitive. The respective functions, responsible for each step in Alg. 1, are easy to use. Especially the provided examples are very helpful. Furthermore, an extensive note is included, providing tips for hyperparameter setting and troubleshooting.

When learning impedance strategies, some of the provided functions proved to be prone to numerical errors. This was especially the case for the function responsible for computing actions. Several attempts were made to fix the issue, but no adequate solution has been found. Another reoccurring problem was the large memory requirement in the computer during policy optimization. For example, doing policy optimization with a dataset of 700 state-samples requires 2.0 GB of memory. To circumvent this limitation, the data from the oldest rollout is removed from the dataset when the memory usage reaches the upper limit.

Long computation time is another important drawback of the PILCO-algorithm package. For real time systems such as the Panda robot, the implementation is not optimal. For example, it would be beneficial to use the RBF policy in the experimental setup as well, but this was simply not possible due to slow action-calculations. The action-computation of the linear policy model is a limiting factor in terms of control frequency as well. In fact, if the function responsible for action-calculation had not been adjusted to be more computationally efficient, it would not be applicable to the experimental setup either. Even after significantly reducing the computation time from 80- to 8 ms, the control frequency had to be lowered from 30- to 20 Hz when introducing learning. The additional, on average, 8 ms made the time-step way too stochastic when running on 30 Hz.

6.3 PILCO for Learning Compliant Control

Randomized trials

When applying the PILCO algorithm to learn force control, several adjustments proved to be necessary. For example, it was identified early on that the initial trials, where the controller is applying random control inputs, should only be partially random. When randomly changing the inputs at every iteration, the system can become unstable. Moreover, it is unfeasible to construct a representative model of the transition dynamics. Even when remaining stable, it is impossible to separate the effects of each control input. The solution was to only change one control input at the time, and letting it remain unchanged for some iterations. This way, the models of the transition dynamics were improved, significantly increasing learning efficiency.

It is important for the initial data-collecting rollouts to represent a variety of force profiles, to provide a rich dataset for learning a locally accurate model. If the randomized policy is causing an overshoot in impact force at every trial, there is a great chance that the dynamics model will be highly uncertain about other more desirable states. This often resulted in poor policies, causing big impact forces. Thus, to ensure that different behaviours are represented in the initial trials, each trial is assigned to randomly sample from a specific part of the action-space. The first trial of the Hybrid Control is for example sampling random stiffness values in the interval 10-25, even though the full action-space ranges all the way up to 90.

Optimization

Given a representative selection of randomized trials, and appropriate hyperparameters, the learning proved to be very sample efficient. This was for example illustrated in Fig. 26, when using the 20 Hz sampling frequency. In this test, a standard procedure of doing 4 randomized trials and 16 on-policy trials was used. As each trial is for a duration of 5 seconds, the total interaction time never exceeded 100 seconds. An extended illustration of the Hybrid Controller's learning efficiency with a 20 Hz sampling frequency, including the variance of the position- and velocity model, is shown in Fig. 37. Already after the first model- and policy optimization, the learned impedance strategy is capable of collecting a top-tier reward. Furthermore, the GP variances have saturated. An interesting effect is however shown from the seventh optimization of the velocity model. When the size of the dataset has reached its upper limit, the variance in the velocity model decreases for five straight optimizations. This is an effect of sequentially removing the data from the first five trials and replacing them with new ones. Out of these five trials, four of them were performing random actions. This suggests that the model is more precise when only being fed with data collected from similar impedance strategies. The reduction in model variance result in more stable rewards.

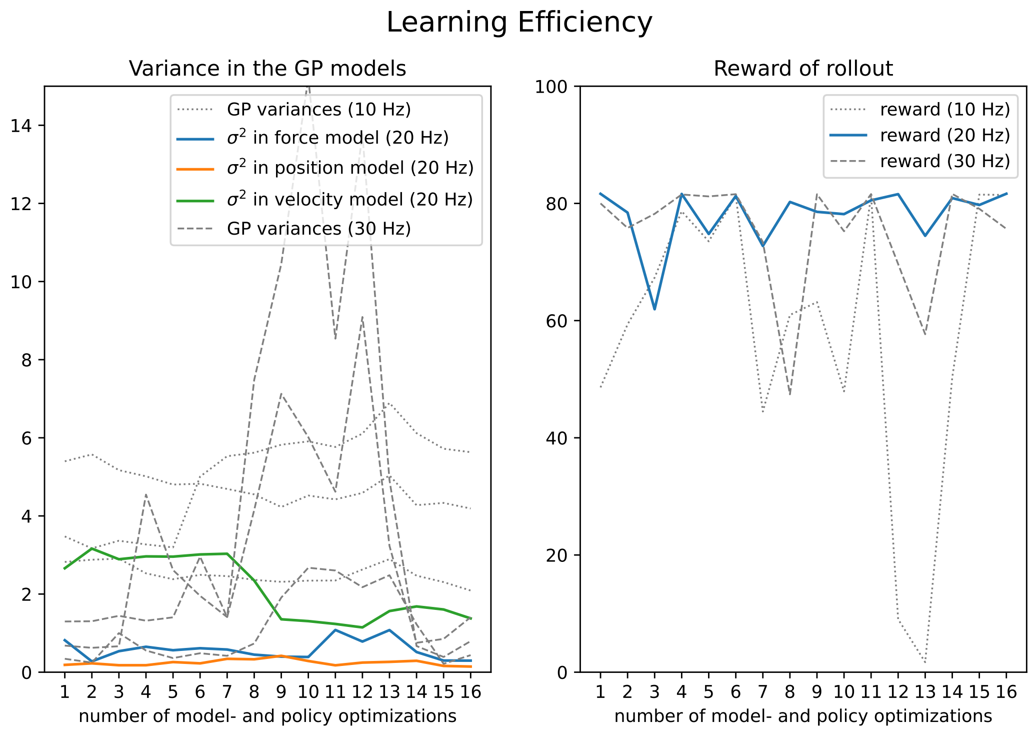


Figure 37: Illustration showing the convergence in the model- and policy optimization of the Hybrid Control using the 20 Hz sampling frequency. The left plot is showing the evolution of variance in the GP models. The right plot is showing the rewards that the resulting linear policy is able to achieve.

In Fig. 38, the evolution in the model variance of the Force-based VIC is compared with one of the Hybrid Control. As their control-inputs have fundamentally different meanings, it is not given that their transition dynamics are equally identifiable. The figure shows that the model variance of the Force-based VIC also converges rapidly. However, the force-model variance of the Force-based VIC is consistently higher than that of the Hybrid Control, consequently making it harder to predict the outcome of actions. Due to the underlying regulating effect of the Force-based VIC however, it is much more robust against modelling errors. Even in its randomized trials, force tracking is achieved at some point during the task.

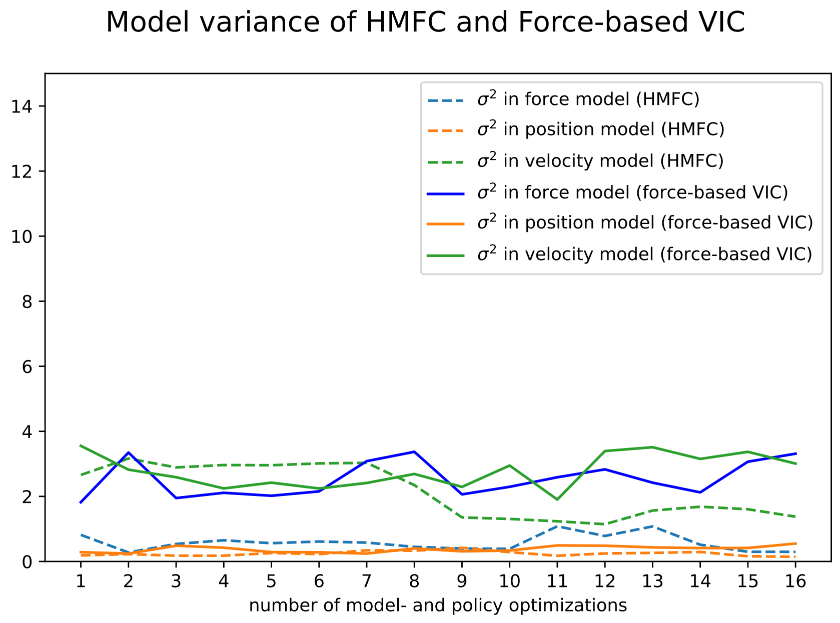


Figure 38: Illustration comparing the convergence in the model variance of the Hybrid Control and the Force-based VIC. The model variance of both force controllers converge rapidly. However, the data suggest that the model of the Force-based VIC is less accurate

In terms of facilitating learning efficiency, except for the representativeness of the initial trials, and the sampling frequency, the reward hyperparameters were the most decisive factors. When setting a high force tracking weight in the exponential reward function, the reward is decaying fast, making exploration harder. A low value weight makes the reward gradient uninformative, or very small in magnitude, slowing down learning. A force tracking weight of 5 proved to accelerate learning. However, it did not offer much exploration, hence the need for a representative set of initial trials.

Limitation of GP modelling

With the current state space, $\boldsymbol{x} = \begin{pmatrix} F & p_z & v_z \end{pmatrix}$, a set of GPs is only able to learn the transition dynamics of one specific object-impedance. Thus, in a dual environment, the two distinct transition dynamics will by tried fitted into one dynamics model, resulting in poor policies. This is illustrated in Fig. 34 and 35. Furthermore, any unforeseen external impact during model optimization, will be tried explained through the model. Consequently, PILCO is not robust to sudden, non-consistent disturbances. Concerning robustness, Cheng and Zhao et. al. proposed the approach $\mathcal{L}_1$ Adaptive Control in [49], aimed at making non-robust RL policies-, among them PILCO-policies, more robust.

To make PILCO applicable to an environment with changing transition dynamics, more than one model is required to represent it, subsequently leading to multiple impedance strategies. Furthermore, the change in transition dynamics needs to be predicable, allowing well timed policy shifts. This is done successfully in Section 5.3.2 when friction and tilt start affecting the Hybrid Control in the experimental setup. In this case, the shift of policy is triggered when the motion trajectory is initiated. The dual policy improves the force tracking ability in both phases, being a logical consequence of not trying to fit distinct transition dynamics into one common model. However, the force tracking of the dual policy is not flawless. In the second phase of the task-execution, shown in red in Fig. 36, there is still a gradual increase in the force tracking error. The timing of it suggest that it is caused by the increasing magnitudes of friction and tilt. Hence, there is a gradual change in transition dynamics, meaning that it is unfeasible to achieve a perfect representation using GPs. Nevertheless, the introduction of supplemental GPs improves performance.

The introduction of supplemental GPs for the dual environment in Fig. 33 however, was not as successful. Starting with the one-policy variant, it is not able to improve force tracking. This is expected given the difficulty of modelling non-homogeneous transition dynamics, illustrated in Fig. 34. The two-policy variant however, is able to marginally improve force tracking. Nevertheless, the contact force is quite bumpy. Furthermore, the tracking when entering the stiffer part of the environment is worse than that of the static controller. For the three-policy variant, the transition dynamics are represented by three sets of GP models. Two of them are representing homogeneous areas of the environment while the latter is representing the border between them. Now, with dedicated models, the force tracking in phase 1 and 3 is significantly improved. However, in phase two, representing the transition between the two distinct environmental impedances, there is a massive overshoot in contact force. The overshoot results in a substantial MSE of 2.2, higher than any other policy composition. The fact that the stiffness of the controller is increased in this critical phase, suggests that the learned model is highly inaccurate.

In sum, the GP models are only giving accurate representations when dealing with homogeneous environments, where the transition dynamics remain constant. When dealing with a complex environment, this is usually not the case, e.g., due to increasing friction, tilt or varying environmental impedance. The result in Fig. 36 shows that the introduction of a supplemental model can improve the total performance. However, as the results of both Fig. 36 and 35 suggests, the task can not necessarily be separated into phases of perfectly homogeneous transition dynamics. If this is the case, and the change in transition dynamics is substantial, as it is in the dual environment of Fig. 33, PILCO is not able to learn optimal policies.

6.4 Application Potential for Robotic Ultrasound Examinations

Starting with the low-level force controllers, both the Hybrid Control and the Force-based VIC have potential in the learning-based framework. In both controllers, learning has the ability to improve their performance. Nevertheless, the improvement is more significant for the Hybrid Control as it does not have the same underlying regulating mechanisms. The most interesting attribute of the Force-based VIC with learning is its apparent ability to mitigate the effect of disturbances by reducing the rate of stiffness-adaptability. However, as the Force-based VIC is adaptive by default, the changing learning rates has a less of an impact than the raw impedance strategy in the Hybrid Control. As its actions are less influential, and there is some force-regulation regardless, the force tracking is more robust, both against modelling errors and general disturbances.

However, the underlying adaptive law limits the potential flexibility of the controller. Regardless the control-input, changes in damping and stiffness parameters are only triggered by force tracking errors. The policy is only able to decide how *much* the damping and stiffness changes. Whether it is an increase or decrease is decided by the sign of the force tracking error. Another drawback of the Force-based VIC is the apparent difficulty of achieving good motion tracking. When tuned, even substantial position- and orientation gains resulted in static deviations. The poor motion tracking ability is illustrated in Fig. 21, where stationary deviations occur despite marginal environmental friction.

To improve force tracking with PILCO, it has been shown that the transition dynamics must be accurately represented by the GP models. Thus, when dealing with environments where the transition dynamics changes during the task-execution, separate models are required. Accordingly, the moment where the dynamics changes must be identifiable. When the change is gradual, there is no perfect way to split it, eventually leading to a suboptimal policy. If this is the case, and the change in transition dynamics is substantial, PILCO is not able to learn optimal policies.

With respect to the robotic ultrasound examination use case, GPs are posing a significant limitation on modelling of the system dynamics. For example, as every transition in the learning process is to conform with a model, also state-changes caused by sudden disturbances are incorporated into the model, leading to inaccurate, fluctuating models of the interaction dynamics. In the ultrasound use case, disturbances such as respiratory movement in a patient will confuse the model as it gradually changes the transition dynamics. Furthermore, the probe cannot move along paths where the stiffness of the body changes significantly.

6.5 Future Directions

As the performance of the implemented force controllers are restrained by software-limitations, it would be interesting to repeat the tests under more optimal conditions. In a software environment allowing higher control frequencies, it is likely that the learning-based controllers will perform considerably better. Specifically, results and reason suggest that the ability to run the controllers at a higher frequency, having more consistent time-step, and using the RBF policy model, will improve performance in experiments.

Nevertheless, the results indicates that GP-modelling is not optimal for this use case. When performing complex interaction tasks in challenging, dynamic circumstances, flexibility is key. It must provide safe and adaptable behaviour, even in the occurrence of unforeseen external influences. This is not GP-modelling's strong suit. Once the transition dynamics unexpectedly changes, the model breaks down. Consequently, for the task at hand, a more flexible learning-framework is needed. This might be achieved by replacing the GPs with BNNs or Neural Networks, e.g., using PETS [33] or Deep PILCO [34], or adopting Roveda et. al's VIC approach [35]. In these approaches, some sampling efficiency would be sacrificed for the benefit of more robust force tracking behaviour in complex environments as they are better suited for modelling discrete dynamics. Being less computationally extensive, and scaling better with bigger action-spaces, more control parameters can be included in the learned strategy. Thus, increasing the controllers' flexibility and making performance less dependent on manual tuning.

7 Conclusion

Constituting the thesis' main contribution, three different force controllers have been implemented and assessed in the state of the art sampling efficient PILCO framework. Furthermore, literature relevant to the task have been reviewed, and a framework has been implemented, providing compatibility between the Franka Emika Panda robot and modern machine learning libraries.

The three fundamental approaches in robotic interaction control, respectively Admittance Control, Hybrid Control and a Force-based VIC, were implemented and evaluated. It was shown that the learning-based framework has the ability to improve performance in both the Hybrid Control and the Force-based VIC. Furthermore, the learning proved to be extremely sample efficient, improving their force tracking ability in just a few trails. For the Hybrid Control, the framework was used to learn direct strategies for its vertical damping- and stiffness parameters. In the Force-based VIC, strategies were learned for the adaptive-rates of the already existing adaptation laws. As the Hybrid Control do not have the same underlying force-regulating mechanisms, its improvement by learning was more significant. Besides, by controlling the impedance characteristics directly, it has a higher maximum achievable flexibility. However, the underlying force-regulation of the Force-based VIC makes the force tracking more robust, both against modelling errors and general disturbances. The Admittance Controller however, did not achieve force tracking behaviour. Only providing compliance vertically, the end-effector was not able to maintain contact with the surface when performing motion trajectories. Moreover, it caused oscillations in the contact force, entailing difficult learning conditions.

The results showed that the transition dynamics must be accurately represented by the GP models for PILCO to improve force tracking. Consequently, if the transition dynamics changes during the interaction task, an accurate model must be learned for each phase. Furthermore, the phases must be separable and recognizable. Thus, irregular external influences will be incorporated into the model, leading to inaccurate, fluctuating models of the interaction dynamics. Consequently, GP modelling does not offer the robustness and flexibility required to model complex interaction dynamics. Hence, in future work, it would be interesting to replace the GPs with Neural Networks or BNNs, sacrificing sampling efficiency for achieving robust force tracking behaviour in complex environments.

References

[1] M. H. Myrestrand, "Compliant robotic manipulation," 2020.

[2] M. Vagia, "Romo," 2019. [Online]. Available: https://www.sintef.no/en/projects/romo/

[3] M. P. Deisenroth, *Efficient reinforcement learning using Gaussian processes.* KIT Scientific Publishing, 2010, vol. 9.

[4] GPflow, "Readme.md," 2021. [Online]. Available: https://github.com/GPflow/GPflow

[5] I. OpenAI, "About openai," 2021. [Online]. Available: https://openai.com/about/

[6] SINTEF, "Applied research,technology and innovation," 2021. [Online]. Available: https://www.sintef.no/en/sintef-group/this-is-sintef/

[7] TensorFlow.org, "Why tensorflow," 2021. [Online]. Available: https://www.tensorflow.org/

[8] I. The MathWorks, "Urdf primer," 2020. [Online]. Available: https://se.mathworks.com/help/physmod/sm/ug/urdf-model-import.html#:~:text=URDF%2C%20or%20Unified%20Robotics%20Description,animatronic%20robots%20for%20amusement%20parks.

[9] L. Villani and J. De Schutter, "Force control," in *Springer handbook of robotics.* Springer, 2016, pp. 195–220.

[10] F. J. Abu-Dakka and M. Saveriano, "Variable impedance control and learning—a review," *Frontiers in Robotics and AI*, vol. 7, 2020.

[11] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[12] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11).* Citeseer, 2011, pp. 465–472.

[13] B. G. et al Revision, "Franka arm," 2018. [Online]. Available: https://de3-rob1-chess.readthedocs.io/en/latest/franka.html

[14] F. E. GmbH, "Franka control interface (fci) - overview," 2017. [Online]. Available: https://frankaemika.github.io/docs/overview.html#:~:text=The%20Franka%20Control%20Interface%20(FCI,workstation%20PC%20connected%20via%20Ethernet.

[15] N. Hogan and S. P. Buerger, "Impedance and interaction control," in *Robotics and automation handbook.* CRC press, 2018, pp. 375–398.

[16] N. Hogan, "Impedance control: An approach to manipulation," in *1984 American control conference.* IEEE, 1984, pp. 304–313.

[17] H.-P. Huang and S.-S. Chen, "Compliant motion control of robots by using variable impedance," *The International Journal of Advanced Manufacturing Technology*, vol. 7, no. 6, pp. 322–332, 1992.

[18] T. Tsuji and Y. Tanaka, "On-line learning of robot arm impedance using neural networks," *Robotics and Autonomous Systems*, vol. 52, no. 4, pp. 257–271, 2005.

[19] C. Li, Z. Zhang, G. Xia, X. Xie, and Q. Zhu, "Efficient force control learning system for industrial robots based on variable impedance control," *Sensors*, vol. 18, no. 8, p. 2539, 2018.

[20] F. Winter, M. Saveriano, and D. Lee, "The role of coupling terms in variable impedance policies learning," in *International Workshop on Human-Friendly Robotics*, 2016.

[21] J. Buchli, E. Theodorou, F. Stulp, and S. Schaal, "Variable impedance control a reinforcement learning approach," *Robotics: Science and Systems VI*, pp. 153–160, 2011.

[22] K. J. Waldron and J. Schmiedeler, "Kinematics," in *Springer Handbook of Robotics.* Springer, 2016, pp. 11–36.

[23] G. J. Lahr, J. V. Soares, H. B. Garcia, A. A. Siqueira, and G. A. Caurin, "Understanding the implementation of impedance control in industrial robots," in *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR).* IEEE, 2016, pp. 269–274.

[24] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 297–330, 2020.

[25] E. Cha, K. Kronander, and A. Billard, "Combined kinesthetic and simulated interface for teaching robot motion models," in *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN).* IEEE, 2015, pp. 83–88.

[26] M. Saveriano, S.-i. An, and D. Lee, "Incremental kinesthetic teaching of end-effector and null-space motion primitives," in *2015 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2015, pp. 3570–3575.

[27] F. J. Abu-Dakka, L. Rozo, and D. G. Caldwell, "Force-based variable impedance learning for robotic manipulation," *Robotics and Autonomous Systems*, vol. 109, pp. 156–167, 2018.

[28] L. Rozo, S. Calinon, D. Caldwell, P. Jiménez, and C. Torras, "Learning collaborative impedance-based robot behaviors," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 27, no. 1, 2013.

[29] D. A. Bristow, K. L. Barton, and A. G. Alleyne, "Iterative learning control," in *The Control Systems Handbook: Control System Advanced Methods, Second Edition.* CRC Press, 2010, pp. 857–876.

[30] A. Kramberger, E. Shahriari, A. Gams, B. Nemec, A. Ude, and S. Haddadin, "Passivity based iterative learning of admittance-coupled dynamic movement primitives for interaction with changing environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6023–6028.

[31] V. van Spaandonk, "Learning variable impedance control: A model-based approach using gaussian processes," 2016.

[32] C. Li, Z. Zhang, G. Xia, X. Xie, and Q. Zhu, "Efficient learning variable impedance control for industrial robots," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, pp. 201–212, 2019.

[33] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *arXiv preprint arXiv:1805.12114*, 2018.

[34] Y. Gal, R. McAllister, and C. E. Rasmussen, "Improving pilco with bayesian neural network dynamics models," in *Data-Efficient Machine Learning workshop, ICML*, vol. 4, no. 34, 2016, p. 25.

[35] E. Roveda, A. Mulè, L. Galasso, L. Castelli, R. Scurati, G. Michielon, F. Esposito, A. Caumo, and A. Montaruli, "Effect of chronotype on motor skills specific to soccer in adolescent players," *Chronobiology international*, vol. 37, no. 4, pp. 552–563, 2020.

[36] WiredWorkers, "Meet the franka emika panda," 2020. [Online]. Available: https://wiredworkers.io/cobot-franka-emika-panda/

[37] ROS, "About ros," 2020. [Online]. Available: https://www.ros.org/about-ros/

[38] A. Dattalo, "Ros / introduction," 2018. [Online]. Available: http://wiki.ros.org/ROS/Introduction#:~:text=ROS%20is%20an%20open%2Dsource,between%20processes%2C%20and%20package%20management.

[39] F. E. GmbH. (2017) libfranka. [Online]. Available: https://frankaemika.github.io/docs/libfranka.html

[40] FlorianWalch. (2017) franka_ros. [Online]. Available: https://wiki.ros.org/franka_ros

[41] F. E. GmbH. (2017) Franka control interface documentation. [Online]. Available: https://frankaemika.github.io/docs/

[42] S. Sidhik, "justagist," 2020. [Online]. Available: https://github.com/justagist

[43] ——, "justagist/franka_ros_interface: Controller and Interface API for Franka Emika Panda Robot Manipulator," Nov. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4270092

[44] ——, “panda_simulator: Gazebo simulator for Franka Emika Panda robot supporting sim-to-real code transfer,” Apr. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3747459

[45] ——, “Panda robot,” 2020. [Online]. Available: https://www.saifsidhik.page/panda_robot/

[46] K. P. Nikitas Rontsis, “Probabilistic inference for learning control (pilco),” 2021. [Online]. Available: https://github.com/nrontsis/PILCO

[47] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.

[48] F. E. GmbH., “Installation on linux,” 2017. [Online]. Available: https://frankaemika.github.io/docs/installation_linux.html

[49] Y. Cheng, P. Zhao, M. Gandhi, B. Li, E. Theodorou, and N. Hovakimyan, “Robustifying reinforcement learning policies with $\mathcal{L}_1$ adaptive control,” *arXiv preprint arXiv:2106.02249*, 2021.