Context-Dependent Variable Impedance Control with Stability Guarantees

Kontextabhängige variable Impedanzregelung mit Stabilitätsgarantien
Master thesis by Leon Keller
Date of submission: May 31, 2023

1. Review: João Carvalho
2. Review: Dorothea Koert
3. Review: Jan Peters
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB der TU Darmstadt

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

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

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

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

Darmstadt, 31. Mai 2023

L. Keller

Abstract

Assembly tasks pose significant challenges for robots as safety must be ensured during all physical interactions. A prominent approach for achieving this safety is impedance control, which allows the robot to respond to external forces by mimicking the behavior of a physical spring-damper system. Typically, the stiffness and damping of this spring-damper system are fixed during the execution of the controller. However, recent research has highlighted the advantages of varying robot compliance during task execution. As manually designing such variable impedance controllers is time-consuming and costly, there has been a growing interest in learning variable impedance profiles autonomously. This work addresses two challenges in current research on variable impedance control: Stability and generalization across contexts. Firstly, we propose a novel parameter transformation to guarantee stability during the entire training process. Secondly, we employ contextualized policy search to acquire context-dependent variable impedance profiles that not only depend on the current state or time but also specific task properties. Through experiments, we demonstrate that our approach is able to learn a context-dependent variable impedance controller capable of adapting its stiffness to a wide range of different friction scenarios.

Contents

Figures and Tables

List of Figures

1 Introduction

Assembly tasks, such as assembling parts in manufacturing, have long been a challenging problem for robots [1, 2]. In these tasks, robots face the intricate challenge of manipulating objects with accuracy while simultaneously guaranteeing the safety of all physical interactions among themselves, the objects being assembled, and the surrounding environment. Accomplishing this goal necessitates the consideration of various factors, including uncertainties that arise from potential inaccuracies in the sensors used for detecting and locating the parts.

Manually programming a robot to handle these challenges is time-consuming and costly [3], and the resulting solutions are inflexible: If properties of the task change, for example, the shape or friction coefficient of a part that needs to be inserted, the robot needs to be reprogrammed. Learning-based approaches offer a promising alternative to manual programming by enabling robots to learn from experience, thus, allowing them to adapt autonomously to such new scenarios [4].

Safety is a critical aspect when applying learning methods to contact-rich tasks like assembly. It is essential to ensure that the robot's behavior remains safe throughout the learning process. In practice, impedance control methods [5] are often used to guarantee safety. These methods allow the robot to respond to external forces by mimicking the behavior of a physical spring-damper system. The compliance of the robot can be controlled by adjusting the stiffness and damping of this system. Traditionally, in impedance control, these stiffness and damping matrices are fixed during execution. However, recent research has shown that humans vary their compliance during tasks [6], and learning agents can improve their performance by adopting a similar approach [7, 8, 9]. Consequently, in recent years there has been growing interest in autonomously learning variable impedance profiles [10] using techniques such as reinforcement learning or imitation learning.

In this work, our objective is to learn stable variable impedance controllers for assembly tasks. In particular, we focus on peg-in-a-hole tasks, which involve inserting a peg-like object into a corresponding hole. To achieve our objective, we utilize a controller that

combines motion generation and state-dependent variable impedance into a single control law [11]. We introduce a novel parameter transformation for this controller that ensures the stability of the dynamical system for every training rollout. Additionally, we employ contextualized policy search to acquire impedance profiles that not only depend on the current state but also on specific task properties, such as the friction coefficient of the object being inserted. This contextual information allows the robot to adapt its compliance to different assembly scenarios, improving its performance and overall flexibility.

2 Background

In this chapter, we present an overview of the methods and concepts that are essential to understanding the contributions of this work. We begin by introducing key concepts of Reinforcement Learning (RL) and relevant algorithms in Section 2.1 Following, in Section 2.2, we delve into the foundations of robotics and discuss impedance control. Finally, in Section 2.3, we summarize a recently proposed approach for stable variable impedance control.

2.1 Reinforcement Learning

Reinforcement Learning [12] is a sub-field of machine learning where an agent has to learn to solve a given task by interacting with an environment. In contrast to supervised machine learning, no labeled data which explicitly indicates optimal behaviour is given to the agent in advance. Instead, RL algorithms are designed to learn by trial and error: The agent receives feedback in the form of rewards or penalties based on its actions. This feedback is then used to learn an decision making function, often called policy or controller, that maximizes the cumulative reward received over time. Learning an optimal policy in RL is particularly challenging due to the complexity of the interaction between the agent and its environment: The actions taken by the agent can have far-reaching consequences, not only influencing the immediate reward but also the state of the environment and subsequently all future rewards.

RL has gained immense popularity in recent years and has been successfully applied in various domains such as robotics [13], gaming [14], finance [15], and healthcare [16]. In this section, we provide a comprehensive background on RL, introducing its basic concepts and the algorithms used throughout this work.

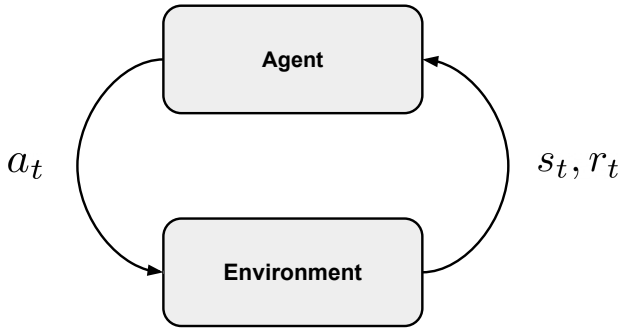


Figure 2.1: Illustration of the agent-environment interaction loop. The agent observes the current state s_t along with the reward r_t and subsequently chooses an action a_t.

2.1.1 Foundations

Reinforcement Learning (RL) problems or tasks are typically formalized as Markov Decision Processes (MDPs) [17]. MDPs are a mathematical framework that provides a formal way of modeling an agent's interaction with an environment, where the agent takes actions based on the current state of the environment and consequently receives a reward or penalty.

A MDP is defined as a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, p_0)$ where $\mathcal{S}$ is the set of all states, $\mathcal{A}$ is the set of all actions, $\mathcal{P}$ is the transition probability distribution, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and p_0 is the starting state distribution. For example, in a robotic task a state $\boldsymbol{s} \in \mathcal{S}$ could encode the current joint positions of the robot, while an action $\boldsymbol{a} \in \mathcal{A}$ could encode motor commands. The transition probability distribution would be given by the dynamics of the physical environment.

MDPs are characterized by the Markov property, which states that the probability distribution over the next state of the environment $\mathcal{P}(\boldsymbol{s_{t+1}}|\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{s_{t-1}}, \boldsymbol{a_{t-1}}, ..., \boldsymbol{s_0}, \boldsymbol{a_0})$ only depends on the current state and action, e.g. $\mathcal{P}(\boldsymbol{s_{t+1}}|\boldsymbol{s_t}, \boldsymbol{a_t})$, and consequently is independent of any previous states or actions.

Figure 2.1 shows the interaction loop between an agent and the environment: The first state $\boldsymbol{s_0}$ of the environment is randomly sampled from the starting state distribution p_0.

After that, the agent observes the current state $\boldsymbol{s_t}$ and chooses an action $\boldsymbol{a_t}$ to execute. Following the environment transitions according to the transition probability function $\mathcal{P}(\boldsymbol{s_{t+1}}|\boldsymbol{s_t}, \boldsymbol{a_t})$ and returns the new state $\boldsymbol{s_{t+1}}$, as well as the corresponding reward r_{t+1} to the agent. This interaction between the agent and the environment produces a trajectory $\boldsymbol{\tau} = (\boldsymbol{s_0}, \boldsymbol{a_0}, r_0, \boldsymbol{s_1}, \boldsymbol{a_1}, r_1, ...)$

The agent decides what action to take in a given state based on a decision making function, usually called policy. This policy can be either deterministic $\boldsymbol{a_t} = \pi(\boldsymbol{s_t})$ or stochastic $\boldsymbol{a_t} \sim \pi(\boldsymbol{a}|\boldsymbol{s_t})$. In order to solve the MDP the agent has to find a policy π which maximizes the expected cumulative reward [12]

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(\boldsymbol{s_t}, \boldsymbol{a_t}) \middle| \boldsymbol{s_0} \sim p_0, \boldsymbol{a_t} \sim \pi(\boldsymbol{a}|\boldsymbol{s_t}), \boldsymbol{s_{t+1}} \sim \mathcal{P}(\boldsymbol{s}|\boldsymbol{s_t}, \boldsymbol{a_t})\right],$$

where γ is a value between 0 and 1 which is typically called the discount factor. This discount factor determines how much the agent should discount future rewards relative to immediate rewards.

A common practice in Reinforcement Learning is to describe the policy as a parameterized function $\pi_{\boldsymbol{\theta}}$, where the output of the policy depends on a set of parameters $\boldsymbol{\theta}$ that can be adjusted to change its behavior. RL algorithms aim at finding a set of parameters such that $\pi_{\boldsymbol{\theta}}$ maximizes the expected cumulative reward. The central optimization problem of RL [12] can thus be expressed as

$$\boldsymbol{\theta^*} = \arg\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}),$$

where $J(\boldsymbol{\theta}) = J(\pi_{\boldsymbol{\theta}})$ is the expected cumulative reward of the policy parameterized by $\boldsymbol{\theta}$. A variety of approaches to find an optimal policy have been developed, with notable examples being value-based methods [18, 19] that estimate the value of state-action pairs and policy gradient methods [20, 21] that directly optimize the policy by adjusting its parameters.

2.1.2 Episodic Relative Entropy Policy Search

Episodic Relative Entropy Policy Search (EREPS) [22] is a Reinforcement Learning algorithm that employs an information-theoretic approach to mitigate information loss during policy updates, preventing premature convergence. EREPS utilizes two distinct policies: A lower-level policy that deterministically maps states to actions $\boldsymbol{a_t} = \pi_{\boldsymbol{\theta}}(\boldsymbol{s_t})$ (corresponding to the policy in the previous sections) and an upper-level policy $\boldsymbol{\theta} \sim \pi_{\boldsymbol{w}}$ that determines the parameters of the lower-level policy.

Rather than directly optimizing the lower-level policy parameters, EREPS optimizes a parametric probability distribution defined by the upper-level policy. Specifically, EREPS maximizes the average return by optimizing the upper-level policy while simultaneously ensuring a smooth and stable learning process by constraining the Kullback-Leibler (KL) divergence [23] between the new and old upper-level policy. This update process can be formulated as the constrained optimization problem

$$\begin{aligned} \max_{\pi} & \int \pi(\boldsymbol{\theta}) \mathcal{R}(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ \text{s.t.} \quad \epsilon & \geq \int \pi(\boldsymbol{\theta}) \log \frac{\pi(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta} \\ 1 &= \int \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}, \end{aligned}$$

where $\mathcal{R}(\boldsymbol{\theta})$ is the cumulative reward of the lower-level policy with parameters $\boldsymbol{\theta}$ and $q(\boldsymbol{\theta})$ is the old upper-level policy. The first constraint ensures that the KL divergence between the old and new upper-level policy is smaller that ϵ while the second constraint ensures that the new upper-level policy is still a valid probability density.

Using the method of Lagrange multipliers [24], a closed-form solution for this constrained optimization problem can be obtained. The new upper-level policy π is given by the soft-max distribution

$$\pi(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta}) \exp\left(\frac{\mathcal{R}(\boldsymbol{\theta})}{\eta}\right), \tag{2.1}$$

where the temperature η is given by the Lagrangian multiplier corresponding to the KL-divergence constraint. The optimal value for η can be obtained by minimizing the dual function

$$g(\eta) = \eta\epsilon + \eta \log \int q(\boldsymbol{\theta}) \exp\left(\frac{\mathcal{R}(\boldsymbol{\theta})}{\eta}\right) d\boldsymbol{\theta}.$$

The dual-function is smoothly differentiable and convex in η and can therefore be efficiently minimized with any standard optimization algorithm such as, for example, the Broyden–Fletcher–Goldfarb–Shannon (BFGS) method [25]. In practice, it is infeasible to compute the integral in the dual function and, hence, it is approximated using Monte Carlo integration

$$g(\eta) = \eta\epsilon + \eta \log \sum_i \frac{1}{N} \exp\left(\frac{\mathcal{R}(\boldsymbol{\theta}^{[i]})}{\eta}\right). \tag{2.2}$$

The resulting algorithm, thus, first generates a set of parameter vectors $\boldsymbol{\theta}^{[i]}$ using the current policy and evaluates them to obtain the corresponding returns $\mathcal{R}(\boldsymbol{\theta}^{[i]})$. Subsequently, the dual function given by Equation 2.2 is minimized to obtain the Lagrange multiplier η. Utilizing this multiplier η, a weight $d^{[i]}$ is computed for each parameter vector $\boldsymbol{\theta}^{[i]}$ using the soft-max distribution given by Equation 2.1. Finally, the new policy is obtained by fitting a new parametric distribution to the weighted samples using weighted maximum-likelihood estimation. This process is then repeated until convergence. The pseudo-code of the full algorithm is outlined in Algorithm 1.

Algorithm 1 Relative Entropy Policy Search

```
j = 0
while not converged do
    for i = 0, ..., N do
        Sample parameter vector θ^[i] ~ π_w(θ)
        Evaluate θ^[i] and obtain R(θ^[i])
    end for
    Compute η by minimizing (2.2)
    for i = 0, ..., N do
        Compute d^[i] using (2.1)
    end for
    Compute w_{j+1} by weighted MLE
    j = j + 1
end while
```

2.1.3 Contextual Relative Entropy Policy Search

Contextual Relative Entropy Policy Search (CREPS) [26] is an extension of EREPS designed for environments where the optimal policy depends on a context variable s. It introduces an extension to the upper-level policy $\pi(\boldsymbol{\theta}|s)$ that incorporates the context variable s to select the lower-level policy parameters $\boldsymbol{\theta}$. Like the standard episodic formulation, CREPS aims to maximize the expected cumulative reward while simultaneously constraining the expected KL-divergence between the new and old policy. The update process of CREPS can be formulated as the constrained optimization problem

$$\begin{aligned}\max_{\pi} & \int \mu(s) \int \pi(\boldsymbol{\theta}|s)\mathcal{R}(\boldsymbol{\theta}, s)d\boldsymbol{\theta}ds \\ \text{s.t.} \quad & \epsilon \geq \int \mu(s)\mathrm{KL}(\pi(\boldsymbol{\theta}|s)||q(\boldsymbol{\theta}|s))ds \\ & \forall s : 1 = \int \pi(\boldsymbol{\theta}|s)d\boldsymbol{\theta},\end{aligned}$$

where $\mathcal{R}(\boldsymbol{\theta}, s)$ is the cumulative reward of the lower-level policy with parameters $\boldsymbol{\theta}$ in context s, $q(\boldsymbol{\theta}|s)$ is the old upper-level policy and $\mu(s)$ is the context distribution. However, solving this optimization problem is infeasible as this formulation would require access to multiple parameter vector samples $\boldsymbol{\theta}^{[i]}{}_{i=1,\dots,N}$ for a single context variable $s^{[i]}$. Moreover, the number of constraints would be infinite for environments with a continuous context variable.

In order to avoid theses challenges, CREPS optimizes for the joint probability distribution $p(\boldsymbol{\theta}, s)$ instead and resorts to matching feature expectations rather than single probabilities. The resulting optimization problem is given by

$$\begin{aligned}\max_{p} & \iint p(\boldsymbol{\theta}, s)\mathcal{R}(\boldsymbol{\theta}, s)d\boldsymbol{\theta}ds \\ \text{s.t.} \quad & \epsilon \geq \iint p(\boldsymbol{\theta}, s) \log\frac{p(\boldsymbol{\theta}, s)}{\mu(s)q(\boldsymbol{\theta}|s)}d\boldsymbol{\theta}ds \\ & \hat{\boldsymbol{\phi}} = \iint p(\boldsymbol{\theta}, s)\boldsymbol{\phi}(s)d\boldsymbol{\theta}ds \\ & 1 = \iint p(\boldsymbol{\theta}, s)d\boldsymbol{\theta}ds,\end{aligned}$$

where $\boldsymbol{\phi}(s)$ is a feature transformation of the context variable and $\hat{\boldsymbol{\phi}}$ is the observed context feature expectation. Again, using the method of Lagrange multipliers, a closed-

form solution for the constrained optimization problem can be obtained

$$p(\boldsymbol{\theta}, s) \propto q(\boldsymbol{\theta}|s)\mu(s) \exp\left(\frac{\mathcal{R}(\boldsymbol{\theta}, s) - \boldsymbol{\phi}(s)^T v}{\eta}\right), \tag{2.3}$$

where η and v are Lagrange multipliers. The context dependent baseline $V(s) = \boldsymbol{\phi}(s)^T v$ which is subtracted from the cumulative rewards can be interpreted as a value function [26]. Similar as in the standard episodic formulation the Lagrange multipliers can be obtained by minimizing the dual function

$$g(\eta, v) = \eta\epsilon + \hat{\boldsymbol{\phi}}^T v + \eta \log\left(\sum_i \frac{1}{N} \exp\left(\frac{\mathcal{R}(\boldsymbol{\theta}^{[i]}, s^{[i]}) - \boldsymbol{\phi}(s^{[i]})^T v}{\eta}\right)\right), \tag{2.4}$$

where the integrals are approximated using Monte Carlo methods.

The resulting algorithm, first generates parameter vectors $\boldsymbol{\theta}^{[i]}$ using the current policy $\pi(\boldsymbol{\theta}|s)$ and the observed context variables $s^{[i]}$ and evaluates them to obtain their cumulative return $\mathcal{R}(\boldsymbol{\theta}, s)$. After that, the dual function given by Equation 2.4 is minimized to obtain the Lagrange multiplier η and v. Using these multiplies a weight $d^{[i]}$ is computed for each parameter vector $\boldsymbol{\theta}^{[i]}$ using the soft-max distribution given by Equation 2.3. Finally, the new policy is obtained by fitting a new parametric distribution to the weighted samples using weighted maximum-likelihood estimation. This process is then repeated until convergence. The pseudo-code of the full algorithm is given in Algorithm 2.

2.2 Robotics

Robotics is a rapidly evolving sub-field of computer science and engineering that focuses on the development of machines that automate tasks and assist humans in a variety of ways [27]. With the advancements in technology, robots are increasingly being utilized in various domains like for example manufacturing [28], transportation [27] and assembly [2].

In this section we first introduce key concepts in robotics software and following describe impedance control, a control method often used in contact-rich tasks.

Algorithm 2 Contextualized Relative Entropy Policy Search

$j = 0$
while not converged **do**
  for $i = 0, ..., N$ **do**
    Observe context variable $s^{[i]}$
    Sample parameter vector $\boldsymbol{\theta}^{[\boldsymbol{i}]} \sim \pi_{\boldsymbol{w}}(\boldsymbol{\theta}|s^{[i]})$
    Evaluate $\boldsymbol{\theta}^{[\boldsymbol{i}]}$ and obtain $\mathcal{R}(\boldsymbol{\theta}^{[\boldsymbol{i}]}, s^{[i]})$
  end for
  Compute η and v by minimizing (2.4)
  for $i = 0, ..., N$ **do**
    Compute $d^{[i]}$ using (2.3)
  end for
  Compute $\boldsymbol{w_{j+1}}$ by weighted MLE
  $j = j + 1$
end while

2.2.1 Foundation

In this work, we consider robots that consists of a series of rigid bodies or links which are connected by spherical or prismatic joints. Moreover, we assume that the robot is an open kinematic chain meaning that the first link of the robot is fixed and the last link is free to move in space.

In order to control a robot its joint can be actuated to achieve a desired rotation or translation. The current rotation or translation of the joints is called the joint configuration and is typically denoted by a vector $\boldsymbol{q}$. The corresponding joint velocities and accelerations are typically denoted as $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ respectively. For practical applications it is often more desirable to control the cartesian position and rotation of the robots last link, instead of the angles of the joints. This last link is called the end-effector and its cartesian position and rotation is typically denoted as $\boldsymbol{x}$, the corresponding velocities and accelerations as $\dot{\boldsymbol{x}}$ and $\ddot{\boldsymbol{x}}$ respectively.

The relationship between the joint configuration of a robot and the position and orientation of the end-effector is described by the kinematics equations: Using the geometric properties and arrangements of the robots links and joints one can derive a mapping $\boldsymbol{x} = f(\boldsymbol{q})$ from joint positions to the cartesian position and orientation of the end-effector, which is called the forward kinematics. The reverse mapping $\boldsymbol{q} = f^{-1}(\boldsymbol{x})$, which computes joint positions for a given position and orientation of the end-effector, is called inverse kinematics. The

process of inverse kinematics is generally more complex compared to forward kinematics due to the non-uniqueness of the inverse mapping.

The relationship between joint velocities $\dot{\boldsymbol{q}}$ and end-effector velocities $\dot{\boldsymbol{x}}$ can be described using the time-derivative of the kinematics equations

$$\dot{\boldsymbol{x}} = \frac{d}{dt} f(\boldsymbol{q}) = \frac{df(\boldsymbol{q})}{d\boldsymbol{q}} \frac{d\boldsymbol{q}}{dt} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}},$$

where $J(\boldsymbol{q})$ is a matrix which is usually called the Jacobian of the robot. For robots that are open kinematic chains, this Jacobian matrix can be computed using the transformation matrices provided by the forward kinematics.

In general, kinematics is only concerned with geometric aspects of a motion and does not consider the forces and torques which caused the motion. The analysis of the forces and torques acting on the robot and its components is called robot dynamics. A dynamics model of a robot can be used to compute the forces and torques required to achieve a desired motion of the robot and, thus, to design control algorithms. A general formulation of the dynamics model of an uncontrolled robot is given by

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau_{ext}}, \tag{2.5}$$

where $\boldsymbol{M}(\boldsymbol{q})$ represents the Mass matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ represents the Coriolis and Centripetal forces, $\boldsymbol{g}(\boldsymbol{q})$ represents the gravity and $\boldsymbol{\tau_{ext}}$ represents the external forces from the environment.

2.2.2 Impedance Control

Impedance control [5] is a technique often used to control robots in contact-rich tasks. It allows the robot to prevent damage and ensure safety by controlling how the robot behaves during the presence of external forces. The key idea of impedance control is to model the robot as an impedance to its environment using a spring-damper system. By adapting the stiffness and damping of this spring-damper system the compliance of the robot can be specified: Lower stiffness and damping allows for safe interaction, but also results in large tracking error and slow movements. Vice-versa high stiffness and damping allow low tracking errors and fast movements, but lead to unsafe interaction.

Consider the control law

$$\boldsymbol{\tau} = \boldsymbol{S}(\boldsymbol{q_d} - \boldsymbol{q}) + \boldsymbol{D}(\dot{\boldsymbol{q}}_{\boldsymbol{d}} - \boldsymbol{q}) + \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\boldsymbol{d}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}), \tag{2.6}$$

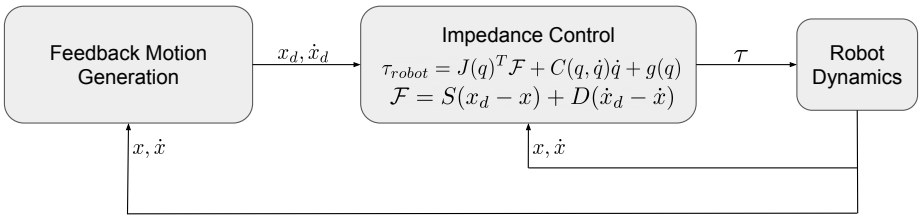


Figure 2.2: Block diagram illustrating an often used impedance control framework. It consists of two distinct control loops: An inner control loop for impedance control and an outer control loop for motion planning.

where $\boldsymbol{S}$ is the stiffness matrix, $\boldsymbol{D}$ is the damping matrix and $\boldsymbol{q_d}, \boldsymbol{\dot{q}_d}, \boldsymbol{\ddot{q}_d}$ are desired values for the joint angles, velocities and accelerations. Inserting 2.6 into the robot dynamics 2.5 yields an equation for the closed-loop system

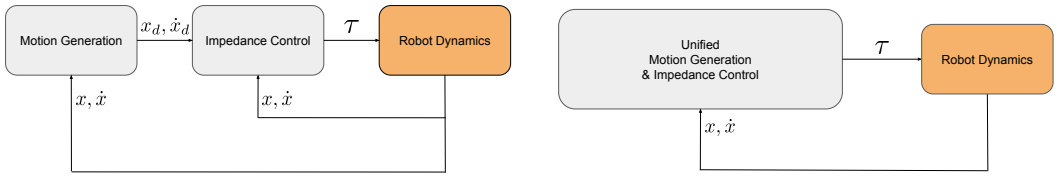
$$\boldsymbol{\tau_{ext}} = \boldsymbol{S}(\boldsymbol{q_d} - \boldsymbol{q}) + \boldsymbol{D}(\boldsymbol{\dot{q}_d} - \boldsymbol{q}) + \boldsymbol{M}(\boldsymbol{q})(\boldsymbol{\ddot{q}_d} - \boldsymbol{\ddot{q}}).$$

Clearly, the robot behaves as a spring-damper system to its environment and the stiffness and damping of the system are defined by the matrices $\boldsymbol{S}$ and $\boldsymbol{D}$. Thus, the compliance of the robot can be controlled by adapting these matrices. Alternatively, by leveraging the robot's Jacobian, it is possible to define the spring-damper system in Cartesian space [29]

$$\begin{aligned}\boldsymbol{\tau} &= \boldsymbol{J}(\boldsymbol{q})^T \mathcal{F} + \boldsymbol{M}(\boldsymbol{q})\boldsymbol{\ddot{q}_d} + \boldsymbol{C}(\boldsymbol{q}, \boldsymbol{\dot{q}}) + \boldsymbol{g}(\boldsymbol{q}) \\ \mathcal{F} &= \boldsymbol{S}(\boldsymbol{x_d} - \boldsymbol{x}) + \boldsymbol{D}(\boldsymbol{\dot{x}_d} - \boldsymbol{\dot{x}}).\end{aligned}$$

This is often preferable as it provides a more intuitive way to determine suitable stiffness and damping values because the compliant behavior can be defined in terms of forces in cartesian space, instead of in joint space.

In practice, impedance control is typically combined with a second control-loop, for example a feedback controller, which handles motion generation by setting the desired values for the joint angles, velocities and accelerations of the impedance controller. This allows the robot to follow a predefined reference trajectory while controlling its compliance using the impedance law. A block diagram of an impedance controller with feedback motion planning is shown in Figure 2.2.



(a) Classical impedance control

(b) IMOGIC

Figure 2.3: Comparison between a classical impedance control architecture and the IMOGIC control architecture. In contrast to the classical architecture, IMOGIC unifies motion generation and impedance control into a single control loop. This simplifies the global stability analysis of the system significantly.

2.3 Integrated Motion Generation And Impedance Control

Integrated Motion Generation And Impedance Control (IMOGIC) [11] is a recently proposed framework for stable variable impedance control. In contrast to previous approaches which rely on two distinct control loops for motion generation and interaction control, IMOGIC unifies motion generation and impedance control into a single control law. Figure 4.1 compares the unified control loop used by IMOGIC with a classical two-loop architecture. Robot motions are modeled as a time-invariant dynamical system which consists of a nonlinear weighted combination of several linear spring-damper systems. Consequently, the nominal trajectory of the robot motion and the impedance profile are described by a single set of parameters, which simplifies stability analysis and allows the derivation of sufficient conditions for global asymptotic stability of the overall system by Lyapunov analysis.

2.3.1 Definition

The control law consists of several spring-damper terms which are combined using a state-dependent weighting function. The first spring-damper term is always active (implying a weight of 1) and has the origin as attractor point. The other spring-damper terms can have arbitrary attractors and are only active in certain sub-spaces of the state-space. The

overall control law is given by

$$\begin{aligned}\boldsymbol{\tau} &= -\boldsymbol{S^0}\boldsymbol{x} - \boldsymbol{D^0}\dot{\boldsymbol{x}} - \sum_{k=1}^{K} \omega^k(\boldsymbol{x}) \left[\boldsymbol{S^k}(\boldsymbol{x} - \boldsymbol{x^k}) + \boldsymbol{D^k}\dot{\boldsymbol{x}}\right] \\ &= -\bar{\boldsymbol{S}}(\boldsymbol{x})(\boldsymbol{x} - \bar{\boldsymbol{x}}(\boldsymbol{x})) - \bar{\boldsymbol{D}}\dot{\boldsymbol{x}} \\ &= f(\boldsymbol{x}, \dot{\boldsymbol{x}}),\end{aligned}$$

where $\{\boldsymbol{S^k}\}_{k=0,\dots,K}$ are the Stiffness matrices, $\{\boldsymbol{D^k}\}_{k=0,\dots,K}$ are the Damping matrices, $\{\boldsymbol{x^k}\}_{k=1,\dots,K}$ are the attractor points and $\omega^k(\boldsymbol{x})$ is the state-dependent weighting function. The state $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ can be either defined in joint or operational space. The control law can be seen as a state-dependent variable impedance controller which varies the stiffness of the spring and the viscosity of the damper across the state-space. In addition, the equilibrium position of the spring, also, varies across the state-space. The weighting function $\omega^k(\boldsymbol{x})$ is always greater or equals to zero and is defined as the product of two positive scalars

$$\omega^k(\boldsymbol{x}) = \alpha^k(\boldsymbol{x})\beta^k(\boldsymbol{x}).$$

The scalars $\alpha^k(\boldsymbol{x})$ and $\beta^k(\boldsymbol{x})$ are parameterized by $\boldsymbol{S^k}, \boldsymbol{x^k}, l^k$ and are designed such that the accumulation of the several spring-damper systems does not create any unwanted attractors and does not cause instability

$$\alpha^k(\boldsymbol{x}) = \begin{cases} \boldsymbol{x}^T\boldsymbol{S^k}(\boldsymbol{x} - 2\boldsymbol{x^k}) & \text{if } \boldsymbol{x}^T\boldsymbol{S^k}(\boldsymbol{x} - 2\boldsymbol{x^k}) \geq 0 \\ 0 & \text{if } \boldsymbol{x}^T\boldsymbol{S^k}(\boldsymbol{x} - 2\boldsymbol{x^k}) < 0 \end{cases},$$

$$\beta^k(\boldsymbol{x}) = e^{-\frac{l^k}{4}(\alpha^k(\boldsymbol{x}))^2}.$$

When certain constraints on the parameters $\boldsymbol{\theta} = \left\{\boldsymbol{S^0}, \boldsymbol{D^0}, \boldsymbol{S^k}, \boldsymbol{D^k}, \boldsymbol{x^k}, l^k\right\}_{k=1,\dots,K}$ are ensured, the system is globally asymptotically stable at the origin, as outlined in the following sub-section.

2.3.2 Stability

The conditions for global asymptotic stability can be derived using Lyapunov's direct method [30]: First a non-negative Lyapunov function needs to be defined. Second, it needs to be verified that this function always decreases throughout the motion and vanishes at the target location. In [11] the authors propose the Lyapunov function

$$V(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{S^0}\boldsymbol{x} + \sum_{k=1}^{K} \frac{1}{l^k}(1 - \beta^k(\boldsymbol{x})) + \frac{1}{2}\dot{\boldsymbol{x}}^T\boldsymbol{M}(\boldsymbol{x})\dot{\boldsymbol{x}},$$

where $\boldsymbol{M}(\boldsymbol{x})$ is the mass matrix of the robot and $\boldsymbol{S^0}, l^k, \beta^k(\boldsymbol{x})$ are defined as in $f(\boldsymbol{x}, \dot{\boldsymbol{x}})$. This Lyapunov function has a unique global minimum at the origin, shares the same set of parameters as $f(\boldsymbol{x}, \dot{\boldsymbol{x}})$ and can be evaluated in closed-form. By following Lyapunov's direct method it can be shown that the gravity-compensated robot motion driven by the control law $\boldsymbol{\tau} = f(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \boldsymbol{g}(\boldsymbol{x})$ is globally asymptotically stable at the origin if

$$\begin{aligned} \boldsymbol{S^0} = (\boldsymbol{S^0})^T &\succ 0 \\ \boldsymbol{D^0} &\succ 0 \\ \boldsymbol{S^k} = (\boldsymbol{S^k})^T &\succeq 0 \\ \boldsymbol{D^k} &\succeq 0 \\ l^k &> 0. \end{aligned}$$

Thus, the control law is globally asymptotically stable at the origin if the state-varying stiffness $\bar{\boldsymbol{S}}$ and damping $\bar{\boldsymbol{D}}$ remains positive definite. Note that globally asymptotically stability is only guaranteed for free-space movement. For example, when an object that is present in the environment is preventing the robot to move towards the target position, asymptotic stability is lost. However, when interacting with an passive, non-actuated environment $f(\boldsymbol{x}, \dot{\boldsymbol{x}})$ remains stable (but not asymptotically stable).

2.4 Related Work

Methods for learning variable impedance controllers were extensively studied in prior research [10]. The learned impedance profiles are typically either state- or time-dependent and are obtained by trial-and-error using reinforcement learning or from human demonstrations using imitation learning.

The application of reinforcement learning to learn variable stiffness and damping profiles has been explored using various policy representations and training algorithms. For instance, in [31] the natural actor-critic algorithm is employed to determine the optimal time-dependent impedance parameters of a 2-link manipulator along a given reference trajectory. In contrast, in [32] the authors simultaneously optimize the gain schedule of a impedance controller and the reference trajectory, represented by a dynamic movement primitive, using the PI^2 algorithm. These methods, however, rely on small parameter spaces in order to reduce the search space.

Notably, recent advancements have integrated impedance control with deep neural networks: In [7, 8, 9], different action spaces for mapping outputs of deep neural network

policies to robot commands are compared, demonstrating that reinforcement learning agents can benefit from allowing the policy to control the stiffness and damping parameters of an impedance controller. Another approach proposed in [33] introduces a neural network architecture that represents a hybrid position-force controller capable of achieving adaptive impedance implicitly using sensed forces. This network is then optimized through a model-based reinforcement learning approach. In [34], an ensemble of neural networks is employed to learn an interaction dynamic model, which is then utilized to optimize the variable stiffness and damping parameters using model-predictive control and the cross-entropy method

Similar to the control framework used throughout this work, in [35], a variable impedance control law is realized through a mixture of proportional-derivative systems. The parameters are optimized using expectation-maximization based reinforcement learning. Closest to our work, in [36] stable state-dependent variable impedance policies are learned using a controller which unifies motion generation and impedance control into a single control law, and episodic reinforcement learning. In contrast to previous work, which largely disregarded the issue of stability, in [36] stability throughout the whole training process is guaranteed by restricting the exploration distribution to assign zero probability mass to unstable parameter vectors.

Another research direction focuses on learning variable impedance profiles from human demonstrations. For example, in [37], a method is proposed to learn force-based variable impedance skills by probabilistically encoding sensed forces and estimated stiffnesses using a Gaussian mixture model. In [38], the stiffness matrices of a variable impedance controller are estimated from kinesthetic demonstrations using Gaussian mixture regression. In the work presented in [39], dynamic movement primitives (DMPs) are employed to learn position and force trajectories from human demonstrations. Subsequently, a control policy which maps position and force to the desired change in robot impedance is learned through deep reinforcement learning. By utilizing inverse reinforcement learning, the method proposed in [40] learns both the variable impedance profile and the reward function from expert demonstrations.

While the existing literature has primarily focused on state- or time-dependent impedance profiles, we address the limitations of such approaches and explore the advantages of incorporating context variables into the impedance profile. By adapting the compliance of the controller based on the current context, we aim to enhance the performance and safety of robotic insertion tasks across diverse scenarios.

3 Context-Dependent Variable Impedance Control With Stability Guarantees

In this work, we propose a novel framework to learn context-dependent variable impedance controllers for robot insertion tasks with stability guarantees. To accomplish this, we utilize a controller that unifies motion generation and state-dependent variable impedance into a single control law. We propose a novel parameter transformation for this controller which allows us to guarantee the stability of the dynamical system for every training rollout. Furthermore, we leverage contextualized policy search to acquire impedance profiles that are not solely dependent on the current state but also on specific task properties, such as the friction coefficient of the object being inserted.

In this chapter we provide a detailed explanation of our approach. First, in section 3.1 we introduce the aforementioned controller parameter transformation. Following, in section 3.2 we explain how contextualized policy search can be used to learn context-dependent variable impedance profiles.

3.1 Stable Training

While the issue of stability remains unaddressed in most research on variable impedance controllers, stability guarantees are crucial for widespread adoption to real world robotic tasks: Unstable controllers can lead to unsafe behaviour of the robot which, in the worst case, can damage the system or its environment. In contrast, stability guarantees make the movement of the robot more predictable, prevent erratic behaviour and, thus, increase safety. Moreover, asymptotic stability provides a strong bias during training. Insertion tasks are goal-orientated, meaning that a successful controller must move the robots end-effector to a goal position which is predefined by the position of the hole in which the object needs to be inserted. While unstable controllers are able to reach almost

any final position, asymptotic stability guarantees that every rollout converges to this predefined goal position, assuming no object in the environment blocks its movement. Thus, asymptotic stability restricts the training process to a smaller, more promising area of the full workspace and thereby increases sample-efficiency.

Consider the IMOGIC controller, as discussed in Section 2.3. The controllers tune-able parameters are given by

$$\boldsymbol{\theta} = \left\{ \boldsymbol{S^0}, \boldsymbol{D^0}, \boldsymbol{S^k}, \boldsymbol{D^k}, \boldsymbol{x^k}, l^k \right\}_{k=1,\dots,K},$$

and the controller is guaranteed to be globally asymptotically stable when all matrices $\{\boldsymbol{S^k}\}_{k=0,\dots,K}, \{\boldsymbol{D^k}\}_{k=0,\dots,K}$ are positive definite and $l^k > 0, \forall k = 1, \dots, K$. Directly optimizing these parameters using an off-the-shelf episodic reinforcement learning algorithm is not viable as unrestricted updates of the parameters may lead to the loss of the positive definiteness of the matrices, thereby compromising the system's stability properties. In contrast to previous work [36] which ensures that the constraints on $\boldsymbol{\theta}$ are met by restricting the search distribution, our approach guarantees stability for every rollout by transforming the parameter space of the policy. For that, we introduce a set of auxiliary parameters

$$\boldsymbol{\phi} = \left\{ \widetilde{\boldsymbol{s^0}}, \widetilde{\boldsymbol{d^0}}, \widetilde{\boldsymbol{s^k}}, \widetilde{\boldsymbol{d^k}}, \widetilde{\boldsymbol{x^k}}, \widetilde{l^k} \right\}_{k=1,\dots,K},$$

along with a bijective mapping $f : \Phi \to \Theta$. The bijectiveness of this mapping is crucial as it guarantees that every possible set of parameters $\boldsymbol{\theta} \in \Theta$ can be represented by a corresponding set of auxiliary parameters $\boldsymbol{\phi} \in \Phi$, and vice versa. Importantly, the mapping is designed to ensure that the aforementioned parameter restrictions for $\boldsymbol{\theta}$ are always fulfilled, while no specific restrictions need to be imposed on the auxiliary parameters $\boldsymbol{\phi}$. This approach enables us to leverage off-the-shelf episodic reinforcement learning algorithms, such as EREPS, to learn the controller parameters while preserving global asymptotic stability throughout each rollout.

Defining the mapping function for the parameters $\{l^k\}_{k=1,\dots,K} \subset \boldsymbol{\theta}$ is straightforward: We adopt a log-space representation and the transformation is, thus, defined using the exponential function, i.e.

$$l^1 = e^{\widetilde{l^1}}.$$

For the positive definite matrices $\{\boldsymbol{S^k}, \boldsymbol{D^k}\}_{k=0,\dots,K} \subset \boldsymbol{\theta}$ the mapping can be defined using the Cholesky decomposition: Any real positive definite matrix can be expressed as the product of a lower triangular matrix with real and positive diagonal entries, and its transpose. This lower triangular matrix is unique for each positive definite matrix and,

hence, the cholesky decomposition can be used to define a bijective mapping. Exemplary for $\boldsymbol{S^0} \in \boldsymbol{\theta}$, the mapping can be defined as

$$S^1 = \begin{pmatrix} e^{\widetilde{s_1^1}} & 0 & 0 \\ \widetilde{s_2^1} & e^{\widetilde{s_4^1}} & 0 \\ \widetilde{s_3^1} & \widetilde{s_5^1} & e^{\widetilde{s_6^1}} \end{pmatrix} \begin{pmatrix} e^{\widetilde{s_1^1}} & 0 & 0 \\ \widetilde{s_2^1} & e^{\widetilde{s_4}} & 0 \\ \widetilde{s_3^1} & \widetilde{s_5^1} & e^{\widetilde{s_6^1}} \end{pmatrix}^T . \tag{3.1}$$

By leveraging the proposed parameter transformation, we are now able to utilize EREPS to learn the controller parameters while maintaining global asymptotic stability during every rollout. First, we define the upper-level policy as a probability distribution over the auxiliary parameters $\boldsymbol{\phi}$. While any probability distribution could be used, in this work we stick to a simple Gaussian distribution, i.e. $\mathcal{N}(\boldsymbol{\phi}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Next, we define the lower-level policy using the IMOGIC controller alongside the proposed bijective mapping function: For a given sample $\boldsymbol{\phi}^{[i]} \sim \mathcal{N}(\boldsymbol{\phi}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ the mapping is used to compute the controllers parameters $\boldsymbol{\theta}^{[i]} = f(\boldsymbol{\phi}^{[i]})$ and the cumulative reward $\mathcal{R}(\boldsymbol{\phi}^{[i]}) = \mathcal{R}(\boldsymbol{\theta}^{[i]})$ is obtained using the stable parameter set $\boldsymbol{\theta}^{[i]}$. Finally, using the described policies, we employ the standard EREPS algorithm as described in Section 2.1.2 to optimize the upper-level policy.

3.2 Context-Dependent Variable Impedance

As discussed in Section 2.4, previous research has extensively explored the learning of impedance profiles that vary depending on the state or time. However, it is important to recognize that an optimal impedance profile may not exclusively rely on the environments state or time variables. In fact, there are several scenarios where it can be highly advantageous to learn a variable impedance controller that adjusts its compliance based on the current context.

In many real-world applications, the desired behavior of a controller is influenced by contextual factors or environmental properties. For example, consider an assembly robot tasked with inserting various objects that have significantly different properties. In practice, it is desirable to find a suitable balance by setting the stiffness as low as possible to ensure safety while maintaining a sufficiently high stiffness to effectively solve the insertion task. However, determining this optimal balance may depend on the specific properties of the object at hand: For instance, when dealing with an object characterized by a very low friction coefficient, utilizing a low-stiffness impedance profile would be suitable. On the

other hand, an object with high friction would require a higher stiffness to overcome the insertion-related friction. Consequently, employing a fixed impedance profile without considering the context could lead to suboptimal performance for different insertion tasks. By incorporating a context variable into the impedance profile, the controller can adapt its behavior to accommodate such diverse situations. As a result, the controller can dynamically adjust its impedance to ensure effective and safe interactions for a wide range of insertion objects.

In this work, we consider context variables that are fixed during the execution of the controller, e.g. during a single insertion. Our objective is to learn a function that predicts the suitable variable impedance profile for a given context prior to task execution. To accomplish this, we leverage CREPS and IMOGIC along with the parameter transformation proposed in the previous section. This enables us to learn context-dependent variable impedance profiles while, again, maintaining global asymptotic stability during every rollout. Similar as in Section 3.1 we define the upper-level policy of CREPS as a probability distribution over the auxiliary parameters $\boldsymbol{\phi}$. However, we now utilize a Gaussian distribution whose mean is given by a linear function of the context variable, i.e. $\mathcal{N}(\boldsymbol{\phi}|\boldsymbol{w}^{\boldsymbol{T}}\psi(s), \boldsymbol{\Sigma})$ where $\psi(s)$ represents a user-defined feature transformation of the context variable s. While any feature transformation could be used, in this work we utilize radial basis functions. The lower-level policy is defined using the IMOGIC controller alongside the bijective mapping function, following the procedure discussed in the previous section. Using these policies, we employ the standard CREPS algorithm as described in Section 2.1.3 to optimize the upper-level policy. After convergence, the sought-after context-mapping is given by the mean of the upper-level policy, $\boldsymbol{w}^{\boldsymbol{T}}\psi(s)$, which can be utilized to predict appropriate variable impedance profiles for a given context.

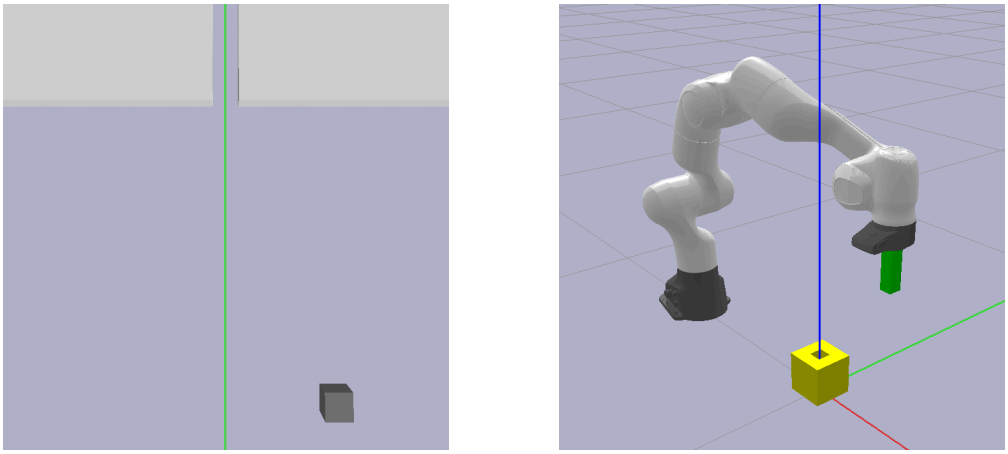
4 Experiments

In this chapter, we showcase the experiments conducted to evaluate the proposed method. In Section 4.1 we explain the process of setting up the environments and provide implementation details. After that, in Section 4.2, we present and discuss the results for both tasks.

4.1 Environments

The primary objective of this research is to develop context-dependent variable impedance controllers specifically designed for insertion tasks. In order to evaluate the effectiveness of the proposed method, it is crucial to create benchmark environments that incorporate contextual factors which influence desired behavior of the controller and, thereby, the optimal impedance profile.

To accomplish this, we implemented two robotic tasks utilizing the PyBullet [41] simulation engine: `Insertion2D` and `InsertionPanda`. Figure 4.1a depicts the `Insertion2D` task, a simple toy task chosen to reduce complexity and provide fast results. It features a 3-dimensional box which is controlled by two prismatic joints along the x- and y-axis. The box's z-axis position and orientation remain fixed. Figure 4.1a provides a visual representation of the `Insertion2D` task, which was intentionally designed to reduce complexity and provide fast results. This task involves a 3-dimensional box that is manipulated by two prismatic joints along the x- and y-axis. The policy moves the box in the x-y plane by controlling these two joints. The position and orientation of the box along the z-axis remain fixed throughout the task. For a more realistic evaluation, we introduced the `InsertionPanda` task, as shown in Figure 4.1b. This task involves a Panda robot equipped with 7 spherical joints, with a box attached to its end-effector. Although the policy has the capability to control all joints of the Panda robot, we specifically utilize an operational space impedance controller with fixed gains for orientation. Our primary



(a) Insertion2D (b) InsertionPanda

Figure 4.1: Screenshots of the implemented environments. (a) shows the `Insertion2D` task, which features a 3-dimensional box which is controlled by two prismatic joints along the x- and y-axis. The box's z-axis position and orientation remain fixed. (b) shows the `Insertion2D` task, which features the 7-dof panda robot with a box attached to its end-effector. In both tasks, the goal is to insert the box into the hole as safely as possible.

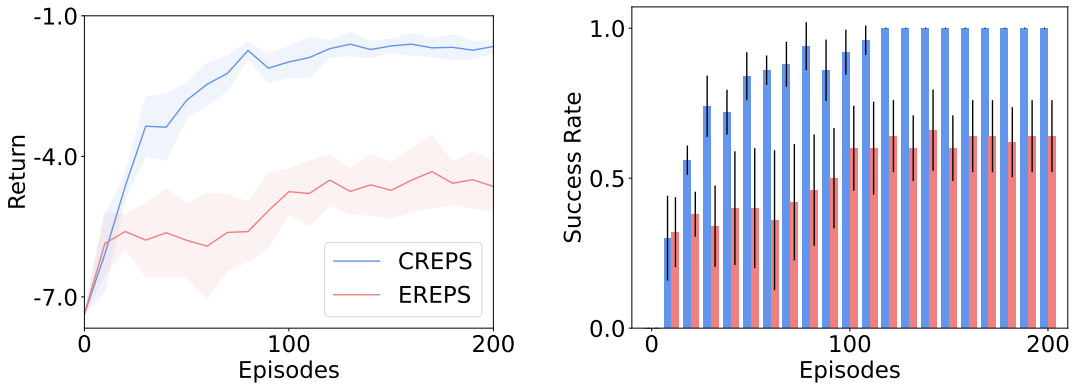
focus is on learning a variable impedance profile for Cartesian translation, leveraging the proposed method.

In both tasks, an action $\boldsymbol{a}$ consists of desired torques or forces for each joint. The state $\boldsymbol{s}$ includes the current joint positions and velocities. To predict joint torques based on their positions and velocities, the policy is designed as explained in Section 3. The initial position of the box or the end-effector is deliberately set laterally away from the hole. An episode terminates either when the insertion is successfully completed or when a predefined maximum number of environment steps is reached. The objective in both tasks is to learn a policy that achieves fast insertion while ensuring safety by using minimal torques or forces in regions of the state-space where collisions might occur. The reward function is carefully crafted to incentivize this behavior and is represented by the equation,

$$R(\boldsymbol{s}, \boldsymbol{a}) = \alpha \mathrm{dist}(\boldsymbol{s}, \boldsymbol{s_g}) + \beta \begin{cases} 0 & \text{if } \mathrm{dist}(\boldsymbol{s}, \boldsymbol{s_g}) > \epsilon \\ \|\boldsymbol{a}\| & \text{else} \end{cases}$$

where dist is a function that measures the distance between its two inputs and ϵ is set such that the torque penalty is only active in regions of the state-space where a potential collision may occur, such as near the goal position. Consequently, an optimal policy will exhibit high stiffness during free-space movement of the task and low stiffness when the box approaches the goal position.

In both tasks, the friction coefficient of the insertion is not fixed, but rather determined by a context value which gets sampled from a context distribution which spans over the range $[0.3, 1.5]$ at the beginning of each episode. This necessitates that the optimal policy must be able to adapt it behaviour based on this context variable: The controller's goal is to achieve an optimal balance by setting the stiffness as low as possible for safety while maintaining sufficient stiffness to effectively perform the insertion task. However, this balance depends on the current context value. For example, when dealing with an object characterized by a very low friction coefficient, a low-stiffness impedance profile would be suitable. Conversely, an object with high friction would require a higher stiffness to overcome the insertion-related friction. Therefore, the controller needs to adapt its stiffness to the observed friction coefficient to achieve an optimal balance across all possible context values.



(a) (b)

Figure 4.2: Illustration of the results achieved on the `InsertionPanda` task. The results are averaged over 10 random seeds, and the plots depict the mean as well as the standard deviation. (a) shows the average episode return and (b) shows the average success rate. While the REPS algorithm was unable to solve the task, the CREPS algorithm reaches a 100 percent success rate across all seeds.

4.2 Experiments

To empirically evaluate our approach, we employ CREPS, as described in Section 3.2, to train the context-dependent variable impedance controller in both environments. We compare this approach to training without context-dependency using EREPS. In each iteration of both algorithms, we utilize 10 rollouts. The algorithms are executed until convergence or a maximum of 25 iterations. The KL-divergence bound is set to 1.0. After each iteration we evaluate the current policy in terms of returns and success rate by executing it on 10 randomly sampled context values.

The results obtained from the `Insertion2D` and `InsertionPanda` tasks are presented in Figure 4.2 and Figure 4.3 respectively. The results are averaged over 10 random seeds, and the plots depict the mean as well as the standard deviation.

First and foremost, it is evident that the REPS algorithm was unable to successfully solve the tasks. This failure can be attributed to the algorithm's inherent incapability to handle the significant stochasticity introduced by the context variable. The contextual nature of

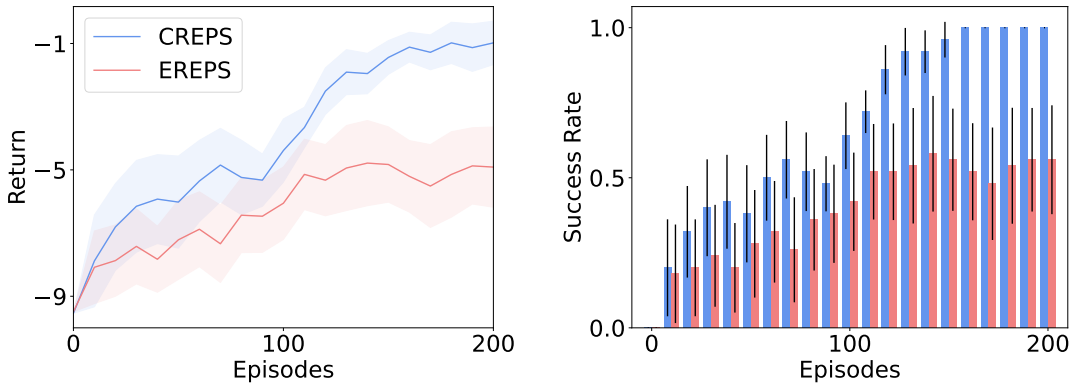


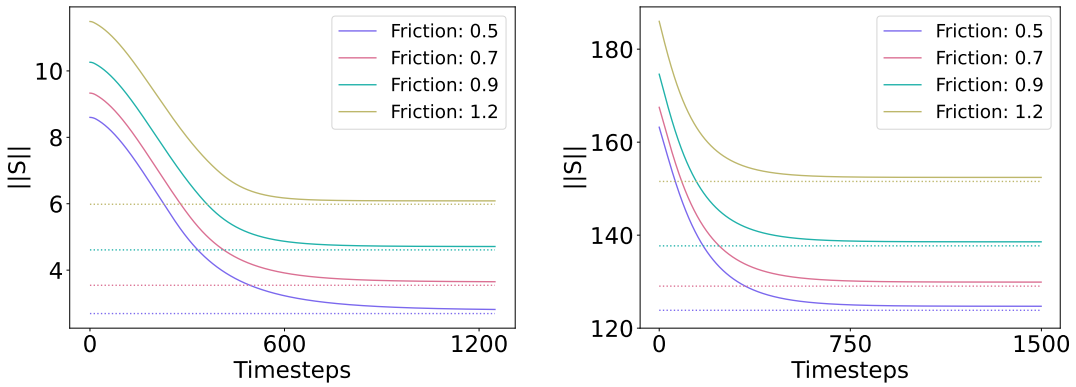
Figure 4.3: Illustration of the results achieved on the `InsertionPanda` task. The results are averaged over 10 random seeds, and the plots depict the mean as well as the standard deviation. (a) shows the average episode return and (b) shows the average success rate. While the REPS algorithm was unable to solve the task, the CREPS algorithm reaches a 100 percent success rate across all seeds.

the tasks presents a challenge for REPS, as it is unable to adapt its parameters to the current context. This limitation is problematic as a policy that exhibits proficiency in a low context value scenario may struggle to achieve successful insertions when faced with a high context value, resulting in lower returns and success rates. One potential approach to mitigate this issue could be to evaluate each parameter vector sample using a diverse range of context values and subsequently averaging the returns across these contexts to determine the parameter vector sample's overall performance. However, this approach faces practical feasibility challenges due to the significant increase in the number of required episodes. Furthermore, even if implemented, EREPS would still learn a context-independent set of parameters and the resulting policy would still be suboptimal, because of the inability to lower the gains of the controller according to the context value.

In contrast, the CREPS algorithm demonstrates remarkable performance, achieving a 100 percent success rate across all seeds. CREPS is specifically designed to effectively handle the stochasticity in the returns introduced by the context variable. By incorporating the context-dependency into the learning process, CREPS is able to adapt its parameters to the varying contexts, resulting in superior performance compared to EREPS. These findings highlight the significance of context-dependency in successfully learning insertion tasks with varying environmental properties like friction. By effectively adapting to the changing contexts, these algorithms provide a more reliable and versatile approach to tackle the challenges posed by such tasks.

Figure 4.4 illustrates the temporal evolution of the norm of the stiffness matrix for multiple friction contexts. The plot compares the learned context-dependent variable impedance controller with hand-tuned fixed gain impedance controllers. The solid lines represent the variable gain controllers, whereas the fixed gain controllers are represented by dotted lines.

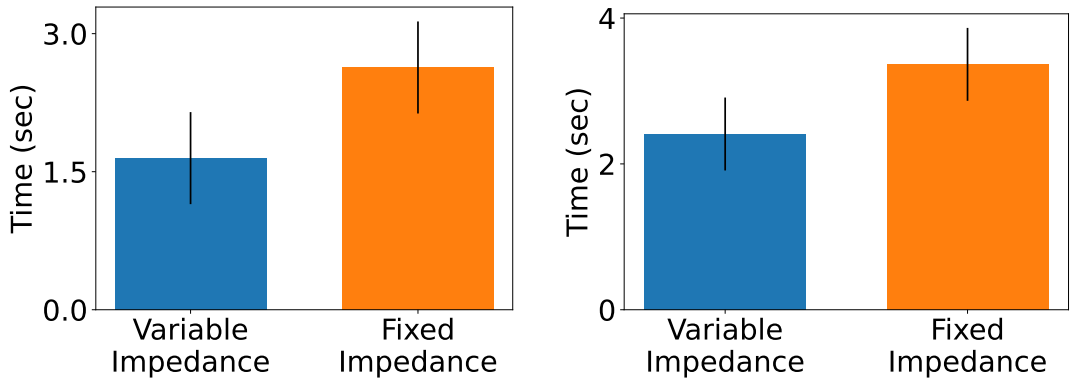
Firstly, it can be observed that the learned controller exhibit a desirable initial characteristic for all contexts, whereby they start with a high stiffness in the free-space movement phase of the task and lower it when approaching the goal position. This varying of the impedance parameters based on the current state allows the learned controller to achieve successful insertions with significantly fewer timesteps compared to the fixed gain controllers, as depicted in Figure 4.5. The figure shows the needed time in seconds averaged over multiple context values. Importantly, this reduction in timesteps is achieved without compromising safety as both the fixed gain and learned controllers maintain an similar level of stiffness near the goal position, where collisions are likely to occur. This similarity in stiffness profiles implies comparable levels of safety for the learned and hand-tuned controllers.



(a) `Insertion2D` (b) `InsertionPanda`

Figure 4.4: Comparison of the temporal evolution of the norm of the stiffness matrix between the learned context-dependent variable impedance controller and hand-tuned fixed gain controllers. The solid lines represent the variable gain controllers, whereas the fixed gain controllers are represented by dotted lines. The learned controller starts with a high stiffness in the free-space movement phase of the task and lowers it when approaching the goal position. Moreover, the learned controller proves to be successful in autonomously utilizing a lower stiffness for lower friction contexts.

Moreover, our method successfully learned a context-dependent variable impedance controller capable of adapting its stiffness to a wide range of different friction scenarios. Specifically, the learned controller exhibited reduced stiffness when dealing with objects characterized by very low friction coefficients and increased stiffness for objects with high friction By employing a lower stiffness when appropriate, the controller can mitigate excessive torques, thereby minimizing the risk of collisions and ensuring safe and reliable execution of the insertion task. While the fixed gain controllers require manual tuning for each specific context, the learned controller successfully adapts to varying contexts without the need for manual adjustment. This ability to generalize across contexts presents a significant advantage of the learned controller. By automatically adjusting its stiffness based on the context, the learned controller demonstrates robustness and adaptability, enabling it to effectively handle insertion tasks with varying friction properties.



(a) `Insertion2D` (b) `InsertionPanda`

Figure 4.5: Comparison of the amount of time needed for an successful insertion. The plot shows the needed time in seconds averaged over multiple context values. 'Variable Impedance' refers to the learned context-dependent variable impedance controller, 'Fixed Impedance' refers to fixed gain controllers which were hand-tuned to every context. The learned variable impedance controller needs significantly less time for a successful insertion.

5 Conclusion

This thesis addressed the challenge of learning stable context-dependent variable impedance controllers for assembly tasks, with a specific focus on peg-in-a-hole insertion tasks. The primary objective was to enable robots to autonomously adapt to different insertion scenarios, thereby improving performance and overall flexibility.

To achieve this goal, we proposed a framework for learning context-dependent variable impedance controllers with stability guarantees. We utilized a controller that integrates motion generation and state-dependent variable impedance into a single control law and proposed a novel parameter transformation for this controller to ensure stability of the system during training rollouts. This parameter transformation enabled the use of standard episodic reinforcement learning algorithms to learn the controllers parameters, while preserving stability properties. Furthermore, we employed contextualized policy search, specifically CREPS, to learn impedance profiles that incorporate context variables, such as the friction coefficient of the object being inserted. This allowed us to account for the impact of contextual factors on the optimal impedance profile. To evaluate our method, we conducted experiments in two benchmark environments where the agent had to insert a box into a hole. In these environments, the optimal impedance profile was not solely determined by the state or time variables but also by the friction coefficient of the insertion object Our results demonstrate that our method successfully learned a context-dependent variable impedance controller capable of adapting its stiffness to a wide range of different friction scenarios. Specifically, the controller exhibited reduced stiffness when dealing with objects characterized by very low friction coefficients and increased stiffness for objects with high friction.

However, our proposed method does have certain limitations that should be considered. Firstly, the controller learned by the approach is always asymptotically stable at the origin. This means that the goal position must be known in advance, because the state space needs to transformed such that the origin represents the desired goal position. Consequently, an accurate and reliable sensor is necessary for estimating the goal pose.

The reliance on such a sensor introduces a dependency on its performance and availability. Furthermore, our method is limited to simple contexts and controllers with a small state-space. Compared to more recent methods like deep reinforcement learning, CREPS can only handle relatively small parameter spaces. The dimension of the parameter space of the upper-level policy depends on both the dimension of the context's feature representation and the dimension of the controller's parameters. Thus, using CREPS becomes impractical for overly complex controllers or contexts. Lastly, the number of rollouts required until the algorithm converges can be demanding. The current approach may necessitate a large number of rollouts during training, which can be time-consuming and computationally expensive. A more sample-efficient approach would be desirable to reduce the training time and resource requirements.

In future work, our primary objective is to evaluate the proposed method on an real robot. However, this transition presents several challenges that must be carefully addressed. One key distinction between simulation and reality is the inability to arbitrarily set the friction of an object. To address this, we plan to involve a human operator in the training process, who can replace objects in the environment with materials possessing the requested friction value after each rollout. As objects, only materials for which the exact friction properties are known, will be used. Alternatively, if the exact friction properties are not known, we can order the available objects based on their supposed friction coefficient and assign increasing context values to each object, ensuring a consistent context for training. For instance, we would assign the lowest context value to the object with the lowest supposed friction coefficient and the highest context value to the object with the highest supposed friction coefficient However, it remains to be determined whether the generalization capabilities observed in simulation will effectively transfer to the real-world setting, as only a discrete set of objects will be available. Furthermore, in the real world the goal position needs to be estimated using sensors and, thus, the exact goal position will not be known. To address this, we plan to extend our approach by incorporating the learning of a transformation that maps the state-space, placing the goal position at the origin. This adaptation will enable the method to handle sensor inaccuracies and still effectively accomplish the task. Lastly, for a successful application in a real-world scenario, reducing the number of required rollouts would be desirable. We intend to improve the sample efficiency of the proposed method by employing a better initialization for the initial control parameters. This initialization could be obtained from simulation results or through manual tuning, enabling the method to converge faster and require fewer rollouts. Despite these challenges, we firmly believe that our approach holds significant promise for real-world scenarios. The stability properties it offers provide a secure exploration process, which is crucial for safety. Additionally, the ability to learn controllers that can

autonomously adapt to diverse assembly scenarios would be highly desirable for achieving full autonomy in the future.

Bibliography

[1] J. Xu, Z. Hou, Z. Liu, and H. Qiao, "Compare contact model-based control and contact model-free learning: A survey of robotic peg-in-hole assembly strategies," *CoRR*, vol. abs/1904.05240, 2019.

[2] M. Suomalainen, Y. Karayiannidis, and V. Kyrki, "A survey of robot manipulation in contact," *CoRR*, vol. abs/2112.01942, 2021.

[3] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 87–94, 2012.

[4] J. Hua, L. Zeng, G. Li, and Z. Ju, "Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning," *Sensors*, vol. 21, no. 4, 2021.

[5] N. Hogan, "Impedance control: An approach to manipulation," in *1984 American Control Conference*, pp. 304–313, 1984.

[6] C. Yang, C. Zeng, C. Fang, W. He, and Z. Li, "A dmps-based framework for robot learning and generalization of humanlike variable impedance skills," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 3, pp. 1193–1203, 2018.

[7] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," *CoRR*, vol. abs/1906.08880, 2019.

[8] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning variable impedance control for contact sensitive tasks," *CoRR*, vol. abs/1907.07500, 2019.

[9] P. Varin, L. Grossman, and S. Kuindersma, "A comparison of action spaces for learning manipulation tasks," *CoRR*, vol. abs/1908.08659, 2019.

[10] F. J. Abu-Dakka and M. Saveriano, "Variable impedance control and learning - A review," *CoRR*, vol. abs/2010.06246, 2020.

[11] M. Khansari, K. Kronander, and A. Billard, "Modeling robot discrete movements with state-varying stiffness and damping: A framework for integrated motion generation and impedance control," 07 2014.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[13] J. Kober, J. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, pp. 1238–1274, 09 2013.

[14] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *CoRR*, vol. abs/1712.01815, 2017.

[15] A. Charpentier, R. Elie, and C. Remlinger, "Reinforcement learning in economics and finance," 2020.

[16] C. Yu, J. Liu, and S. Nemati, "Reinforcement learning in healthcare: A survey," *CoRR*, vol. abs/1908.08796, 2019.

[17] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[18] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, May 1992.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[20] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, (Cambridge, MA, USA), p. 1057–1063, MIT Press, 1999.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

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

[23] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79 – 86, 1951.

[24] *Lagrange Multiplier*, pp. 292–294. New York, NY: Springer New York, 2008.

[25] R. Fletcher, *Practical Methods of Optimization*. New York, NY, USA: John Wiley & Sons, second ed., 1987.

[26] A. Kupcsik, M. Deisenroth, J. Peters, and G. Neumann, "Data-efficient contextual policy search for robot movement skills," 01 2013.

[27] B. Siciliano and O. Khatib, eds., *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer, 2008.

[28] P. Urhal, A. Weightman, C. Diver, and P. Bartolo, "Robot assisted additive manufacturing: A review," *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 335–345, 2019.

[29] C. Ott, *Cartesian Impedance Control: The Rigid Body Case*, pp. 29–44. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[30] J. Mawhin, *Alexandr Mikhailovich Liapunov, The general problem of the stability of motion (1892)*, pp. 664–676. 01 2005.

[31] B. Kim, J. Park, S. Park, and S. Kang, "Impedance learning for robotic contact tasks using natural actor-critic algorithm," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 40, pp. 433 – 443, 05 2010.

[32] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *I. J. Robotic Res.*, vol. 30, pp. 820–833, 06 2011.

[33] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel, "Reinforcement learning on variable impedance controller for high-precision robotic assembly," *CoRR*, vol. abs/1903.01066, 2019.

[34] L. Roveda, J. Maskani, P. Franceschi, A. Abdi, F. Braghin, L. M. Tosatti, and N. Pedrocchi, "Model-based reinforcement learning variable impedance control for human-robot collaboration," *Journal of Intelligent & Robotic Systems*, pp. 1–17, 2020.

[35] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3232–3237, 2010.

[36] S. A. Khader, H. Yin, P. Falco, and D. Kragic, "Stability-guaranteed reinforcement learning for contact-rich manipulation," *CoRR*, vol. abs/2004.10886, 2020.

[37] F. Abu-Dakka, L. Rozo, and D. Caldwell, "Force-based variable impedance learning for robotic manipulation," *Robotics and Autonomous Systems*, vol. 109, 08 2018.

[38] K. Kronander and A. Billard, "Learning compliant manipulation through kinesthetic and tactile human-robot interaction," *IEEE Transactions on Haptics*, vol. 7, no. 3, pp. 367–380, 2014.

[39] C. Chang, K. Haninger, Y. Shi, C. Yuan, Z. Chen, and J. Zhang, "Impedance adaptation by reinforcement learning with contact dynamic movement primitives," 2022.

[40] X. Zhang, L. Sun, Z. Kuang, and M. Tomizuka, "Learning variable impedance control via inverse reinforcement learning for force-related tasks," *CoRR*, vol. abs/2102.06838, 2021.

[41] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." `http://pybullet.org`, 2016–2021.