2025-01-29

Adaptive PD Gains for Energy-Conscious Control in Physical Human-Robot Interaction

Saqib, Danyal

UNIVERSITY OF CALGARY

Adaptive PD Gains for Energy-Conscious Control in Physical Human-Robot Interaction

by

Danyal Saqib

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN MECHANICAL ENGINEERING

CALGARY, ALBERTA
JANUARY, 2025

Abstract

Safe physical human-robot interaction (pHRI) is an important area of research. The dominant control approach in this domain is the impedance control method. However, such control schemes may require force feedback when implemented explicitly, translating to a need for several external force sensors. Impedance control is also a coordinate-dependent scheme, meaning the direction of applied forces must be tracked. These drawbacks indicate the need for alternate control schemes that can ensure safe behavior during pHRI.

An alternative to this is energy-based approaches. One such control method is the 'energy tank' approach, which tries to limit the total energy of a robot using 'virtual tanks'. Another energy-based control scheme is the 'artificial potential field' method, where attractive artificial potential fields are calculated to track a reference position while simultaneously limiting the robot's total energy. However, such methods can become relatively complex to implement. Energy tank approaches use the idea of energy to maintain stability while dynamically changing impedance parameters. Potential field methods are also complex to implement, as they may require constant recalculation of potential fields.

Practically, many robot developers still use Proportional-Derivative (PD) controllers as the mid-level controller of choice, to track reference trajectories. PD controllers have the advantage of being simple to implement and well-understood due to their extensive research and investigation. We hence propose an adaptive PD controller that can limit a robot's energy under any given limit to achieve safe pHRI, by extending these energy limitation concepts to PD controllers. The proportional and derivative gains of the controller change depending on the current energy of the robot. The proposed controller can limit both the kinetic and potential energy of the robot under any given limit, and the behavior of the controller can be shaped using various parameters. We construct a stability proof for the controller and obtain a condition to ensure the controller's stability. We tested the behavior and compliance of this controller on the TALOS Robot by PAL Robotics both in simulation and on the actual robot, verifying the expected safe and compliant behavior.

Preface

This thesis is an original work by the author. Part of the research conducted for the thesis has been presented as a poster titled "Adaptive PD Gains for Energy-Conscious Control in Physical Human-Robot Interaction" at the Alberta Robotics and Intelligent Systems Expo (Alberta RISE) 2024.

Acknowledgements

Foremost, I would like to offer my sincere gratitude to my supervisor, Dr. Marie Charbonneau, for her constant support and guidance throughout my master's studies. Her technical insights were key to the research process throughout, and I was able to avail some amazing research opportunities throughout my degree thanks to her mentorship. I am immensely grateful for her encouragement and enduring belief in my abilities.

I am deeply grateful to the Department of Mechanical and Manufacturing Engineering at the University of Calgary for their continued support and provision of facilities, to make my research work possible.

I also thank the researchers at the University of Waterloo's RoboHub, who graciously hosted me and helped me implement and test my research practically on the TALOS robot. My specific gratitude goes to Mr. Alexander Werner for his technical expertise, Dr. Francisco Andrade for the extensive in-lab support, and Dr. Brandon DeHart for general access to the RoboHub. Their help in obtaining practical results for my research was invaluable.

Finally, I would like to thank all my friends and family who have been a source of unwavering support throughout my degree.

Table of Contents

List of Figures

List of Tables

List of Symbols, Abbreviations, and Nomenclature

Symbol	Definition
CRBA	Composite Rigid Body Algorithm
DOF	Degrees of Freedom
HRI	Human Robot Interaction
PBC	Passivity Based Control
PD Control	Proportional-Derivative Control
pHRI	Physical Human Robot Interaction
PID Control	Proportional-Integral-Derivative Control
RBDL	Rigid Body Dynamics Library
ROS	Robot Operating System
URDF	Unified Robot Description Format

Mathematical Symbol	Meaning
$\mathbb{R}$	Set of all Real Numbers
$\mathcal{C}^1$	Continuous after being differentiated once

Chapter 1

Introduction

In this chapter, we introduce terminology related to human-robot interaction, defining Social Human-Robot Interaction (sHRI) and Physical Human-Robot Interaction (pHRI). The problem of safety within pHRI is discussed, and our research objectives are outlined. The contributions of our research are then listed, and the thesis chapters are then outlined to conclude this chapter.

1.1 Human-Robot Interaction

Human-Robot Interaction (HRI) is a vast field within robotics that specifically deals with humans and robots interacting in some manner. Currently, most robots reside within fairly constrained environments. The goal for HRI research is to provide safe and meaningful ways for humans and robots to interact. HRI has various areas of application such as teaching robots new tasks by demonstration [4] [5], healthcare robotics [6], collaborative robots [7], and social interaction [8]. One such example of HRI applications that has shown very promising results is in autism interventions, with social robots thought to be highly beneficial for children with Autism Spectrum Disorders (ASD) [9] [10].

Research within HRI encompasses a wide variety of topics such as imbuing robots with social intelligence [11], human intent recognition for robots [12] [13], communication by robots through body and oral language [14], human behavior modeling for better HRI [15], and physical Human-Robot Interaction (pHRI) [16]. HRI is generally classified into two main categories - Social Human-Robot Interaction (sHRI) where the robot interacts with humans in visual or auditory manner at a distance, and Physical Human-Robot Interaction (pHRI) when the robot interacts with humans physically [17]. Robots can often leverage both forms of interaction, being capable of social and physical interaction simultaneously.

1.2 Social Human-Robot Interaction (sHRI)

Social Human-Robot Interaction (sHRI) refers to a robot socially interacting with humans. There are certain qualities that have generally been identified as being characteristic of socially interactive robots. [8] describes certain characteristics that a social robot might possess, such as:

- express and/or perceive emotions
- communicate with high-level dialogue
- learn/recognize models of other agents
- establish/maintain social relationships
- use natural cues (gaze, gestures, etc.)
- exhibit distinctive personality and character
- may learn/develop social competencies

However, this is a much more interdisciplinary field involving research areas such as psychology and sociology and will require contributions from all the involved fields [18]. The main focus of our research, however, is Physical Human-Robot Interaction (pHRI).

1.3 Physical Human-Robot Interaction (pHRI)

Physical Human-Robot Interaction (pHRI) refers to interactions involving physical contact between robots and humans. As stated in [16] and [17], such interactions may take place in a variety of settings. Robots may interact physically with humans in an industrial setting, aiding in manufacturing and production tasks. Healthcare is another avenue where robots may leverage pHRI to assist the elderly or individuals facing physical disabilities. Social robots may also leverage physical interaction to establish physical cues, curating better social interaction via physical touch. Such interaction falls under the broader umbrella term of Social-Physical Human-Robot Interaction (spHRI). Hence, contact-based pHRI is a growing field of research with a vast array of possible applications.

1.3.1 Safety within pHRI

When considering applications of pHRI, safety is of utmost importance, since safety is a necessity for HRI applications to find wider applicability. Collisions between a robot and a human may risk a high transfer of

energy, which could result in serious human injury. Researchers have approached safety in pHRI in many different ways. These include mechanical designs that reduce sharp edges, employing lightweight robots with low inertia, and incorporating mechanical compliance by using either softer materials within the design or elastic actuation systems [16]. However, designing a robot that is mechanically compliant while not compromising on performance remains a challenge within robotics. Research has been undertaken to develop control methods that can ensure safe pHRI. These have the advantage of being applicable to more traditional robotic systems, which may have fewer compliant safety features built into them mechanically.

One of the most common ways of ensuring safety for pHRI from a controls perspective is through compliant control. Compliance within this context is defined as control strategies that result in compliant motion. Quite a lot of the work in this domain deals with shaping the mechanical impedance or admittance of the robot. The objective of such control schemes is to define the dynamic relation between the robot's position or velocity and the external forces imparted on the robot by a human [2]. The aim is to model the robot's manipulator as a spring-mass-damper system in response to the applied external forces. Feedback concerning the applied external forces is used to achieve this, and the robot manipulator thus acts in a compliant manner, in accordance with the desired values of elasticity and damping. Schemes such as admittance control and impedance control [19] [1] have been common for defining a control scheme that can react compliantly to external forces. However, such schemes usually require force sensing to accurately define the relationship between external forces and the robot's position/velocity. Such control schemes are also coordinate-dependent - force sensors need to determine not only the magnitude but also the direction of the force to react compliantly. Additionally, the safety certification process for such a robot is a complicated one [3].

It is also noted that robots are being placed within human environments at a rising rate [20] [17], faster than approaches to ensure their safe integration are being developed [21]. This may lead to safety hazards during pHRI [21] [22]. On the other hand, there is also research suggesting that overly conservative standards for robot safety might lead to robots potentially being limited in their capabilities to accomplish tasks [23]. It is hence important to have control approaches that can address safety while also maximizing robot productivity.

Human-robot collision scenarios generally fall into one of the following categories:

- Transient Contact (Collision) - This is dynamic contact with a moving robot - the robot collides with a human while the robot is moving, and a transfer of energy takes place.

- Quasi-Static Contact (Clamping) - This occurs when a robot continues to follow a pre-planned trajectory even though an obstacle has been encountered. While the robot may not be moving, it is still applying a force on the human due to the error between the desired position and current position.

Hence, any suitable control scheme for pHRI should be able to provide safety in both cases. As described in [3], even if collision detection and reaction schemes are used (as proposed in [24] and [25]), it is important that the detection of collision does not depend on the location of impact, nor the robot configuration. The ISO/TS 15066 specifies the limitations on maximum forces/velocities or energy transferred during human-robot contact [23]. The standards also specify that since a robot moving during a contact scenario is potentially dangerous, only a 'stop' action is considered safe. However, as [3] mentions, this policy is unsuitable for collaborative work in tight spaces, and in the case of a quasi-static contact scenario, means that the human may not have the means to escape the situation even after the stop action has occurred [26].

Energy-limiting control for safe pHRI is a recent area of research that has shown promise due to a few factors. [23] state that energy-based safety constraints are generally less conservative than force or velocity based safety constraints. [3] also detail an energy-limiting approach that can lead to safe robot behavior, as specified in the ISO/TS 15066 standards [27]. Energy as a quantity is also coordinate-independent, meaning that working with energy-based safety constraints can be more convenient in certain situations.

1.4 The Research Objectives

Broadly, the goal of this research is to ensure human safety during pHRI according the ISO/TS 15066 standards [27]. Specifically, the aim is to ensure the following criteria for the robot:

- Limit the robot's energy under any given energy limit.
- The proposed solution should be simple to implement.
- The robot should be stable at all times.
- The proposed solution should be experimentally validated both in simulation and in hardware experiments.

1.5 Contributions

The contributions of our research are in presenting an alternate control scheme that can tackle some of the current challenges in safe pHRI, and achieve the research objectives outlined in the previous section. We

present an adaptive PD controller that can limit the robotic system's energy to be under a given limit at all times (the energy limit may of course be dynamically allocated). We present an adaptive PD controller that can limit the robot's energy by adapting the gains of the controller depending on how close the current energy of the system is to the system's specified energy limit. The proposed controller has the following characteristics:

- The PD gains stay close to their 'nominally tuned values' when we are not close to the limit, and the gains only adapt when we are close to the limit.
- The controller can limit the robotic system's total energy, and it can also dampen the effects of high external torques to prevent unstable 'flailing' movements of the robot.
- A Lyapunov stability analysis is done for the proposed controller, and a stability condition is obtained.
- Finally, we test our controller on the TALOS Robot by PAL Robotics to verify that the robot does indeed limit the system energy, resulting in compliant behavior in response to external forces.

The advantage of our proposed adaptive PD controller is that the user can select the initial or 'nominal' gains for the controller. Since the controller gains will remain close to these nominal gains unless the energy limit is approached, the controller behavior will be fairly close to the behavior expected by a simple PD controller with the same gains. This is an advantage, since any of the traditional techniques for tuning the PD controller's nominal gains may also be used for this controller. We are also able to analyze the controller in terms of its gains using techniques from classical controls. In addition to all of these advantages, the controller does limit the robotic system's total energy, which leads to the robot exhibiting desirable behavior from a safe pHRI perspective - the robot offers low resistance (going soft) in response to gentler external forces, whereas higher and sudden forces are opposed to avoid uncontrollable behavior. There are also tunable parameters that can be used to modify the characteristic behavior of the controller with respect to the system's energy.

1.6 Thesis Outline

The rest of the thesis is structured in the following manner. Chapter 2 reviews the literature on compliant control algorithms and energy-limitation in detail. Chapter 3 gives an overview of the proposed adaptive PD controller, including all the mathematical equations that define the controller. Chapter 4 details the Lyapunov stability analysis of the controller and the obtained stability condition. Chapter 5 includes the details of the implementation of the controller. Chapter 6 discusses the experimental results obtained

practically, analyzing the performance of the controller in terms of the energy limitation and the compliance offered by the controller. Chapter 7 highlights potential future work for the controller, and offers concluding remarks.

Chapter 2

Literature Review

This chapter overviews the relevant literature on control algorithms for safe pHRI. We start off with a review of robot modeling. The Proportional Derivative (PD) Controller and the PD+ Controllers are discussed next, discussing their stability in the context of robotics. The next section discusses compliant control algorithms, which are widely discussed and used within pHRI. We start off by explaining what active compliance means, and such compliant controllers are described. The subsequent section deals with the concepts of energy-based control, and the relevant research that utilizes energy to develop controllers that guarantee safety and stability. We then discuss some of the gaps within the literature and the need for a controller that bridges some of these gaps, building up to our proposed adaptive PD controller.

2.1 Robot Modeling

This section is a brief discussion on general robot modeling. When modeling the dynamics of a robot, there are many approaches that may be followed. One approach that may be taken is the application of Newton's equation of motions for each of the robot's joints. However, this approach can get highly tedious as the number of joints increase. The dominant method is hence using the Lagrangian Approach for dynamical system modeling. This approach is widely discussed in literature [28] [29]. We present a brief overview of the method here.

2.1.1 Lagrangian Approach to Dynamical System Modeling

The Lagrangian of a general system, denoted by L, is defined as the difference between the system's kinetic energy (T) and potential energy(U - generally the system's gravitational potential energy):

$$L = T - U \tag{2.1}$$

The equations of motion of the n degree-of-freedom system can then be given by the equation:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \tag{2.2}$$

Where q_i denotes the joints of the robot, and τ_i denotes the torques for each joint. It can be shown that the robot's kinetic energy always takes the form:

$$T = \frac{1}{2}\dot{q}^\intercal M(q)\dot{q} \tag{2.3}$$

Where the $M(q)$ term is called the mass matrix or the inertia matrix. Substituting equation 2.3 into equation 2.1 gives us:

$$L = \frac{1}{2}\dot{q}^\intercal M(q)\dot{q} - U(q) \tag{2.4}$$

When this equation is substituted into equation 2.2, the robot's dynamic model simplifies to the form:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \tag{2.5}$$

where τ is the vector of joint torques, and:

$$C(q,\dot{q})\dot{q} = \dot{M}(q)\dot{q} + \frac{1}{2}\frac{\partial}{\partial q}\left[\dot{q}^\intercal \dot{M}(q)\dot{q}\right] \tag{2.6}$$

$$G(q) = \frac{\partial U(q)}{\partial q} \tag{2.7}$$

The $C(q,\dot{q})$ term is referred to as the coriolis/centrifugal matrix, and the $G(q)$ term is generally called the gravity vector. This is a very widely used form of the robot dynamic equation [28] [29].

2.1.2 Properties of the Robot Matrices

In this section, we discuss some of the properties of robot matrices that are going to be useful in some of the analysis done in future sections. Details of these properties may be found in [28] [29]:

- $M(q)$ is a positive definite matrix. This can generally be thought of as a consequence of the fact that the kinetic energy of the robot cannot be negative, and the mass matrix has to be positive definite. This property can be written down as:

$$q^\intercal M(q) q \geq 0 \quad \forall q \in \mathbb{R}^n \tag{2.8}$$

- $M(q)$ is a symmetric matrix. This means that:

$$M(q) = M(q)^\intercal \tag{2.9}$$

- $\dot{M}(q) - 2C(q, \dot{q})$ is a skew-symmetric matrix. This property can be written down mathematically as:

$$(\dot{M} - 2C)^\intercal = -(\dot{M} - 2C) \tag{2.10}$$

- The quadratic form for a skew-symmetric matrix equals zero. In other words, for a skew-symmetric matrix:

$$(A = -A^\intercal) \implies x^\intercal A x = 0 \;\; \forall x \in \mathbb{R}^n \tag{2.11}$$

  This property can be proven as follows:

$$x^\intercal A x = (x^\intercal A x)^\intercal = x^\intercal A^\intercal x = -(x^\intercal A x) \tag{2.12}$$

$$\tag{2.13}$$

$$\implies x^\intercal A x = 0 \tag{2.14}$$

2.1.3 Alternative Algorithms for Dynamic Modeling

Apart from the Lagrangian approach, there exists the Newton-Euler formulation for computing the dynamics of the robot [29]. In this approach, each link is recursively analyzed for its linear and angular motion. While the final result is the same as the Lagrangian approach, this approach is friendlier for computational

algortihmic implementations. Details of such algorithms and their implementation are discussed in [30] [31].

2.2 PD Controllers - Characteristics and Advantages

The Proportional-Derivative (PD) Controller is a widely used control method for robotic manipulators. The PD Control law can be though of as modeling the robot's manipulator as a mass-spring-damper system, with the equilibrium position being the reference or desired position. When there is a difference or 'error' between the desired and current position of the robot, the robot will tend towards the desired position according to the specified stiffness and damping values. The PD control method can be used in a variety of forms (position control, direct force control) [28] [32].

2.2.1 The PD Torque Controller

In this section, we mainly discuss the torque PD control law for tracking a reference position. Such a controller is a torque-based PD controller of the form:

$$\tau = -K_p \, q_e - K_d \, \dot{q}_e \tag{2.15}$$

Where:

- q_e is the difference between the reference position q_r, and the actual position q ($q_e = q - q_r$)
- The proportional gain K_p - Corresponds to the stiffness of the controlled joint
- The derivative gain K_d - Corresponds to the damping of the controlled joint

PD Controllers offer control over the stiffness and damping values for a robot. For the presented PD torque controller, we do not have force feedback from the environment, but PD controllers nevertheless provide a way of controlling stiffness and damping of joints [33].

2.2.2 PD Control with Gravity Compensation

Often, for robot control, gravity compensation is an important part of the control law. This is especially useful in adaptive control schemes, because we do not want the robot to collapse suddenly due to changing control parameters. For a PD controller with gravity compensation, the modified equation then becomes:

$$\tau = -K_p \, q_e - K_d \, \dot{q}_e + G(q) \tag{2.16}$$

Where the term $G(q)$ refers to the gravity vector of the robot. This is the equation of the gravity-compensated PD controller that we will be working with. Gravity compensation actually plays a crucial role in stability as well. Gravity compensated PD control is also one of the simplest control laws that can be proven to be stable for a robotic system. This is because of the fact that the robot does not have any gravitational potential energy, and the robot hence does not go unstable. Lyapunov stability proofs for such a controller can also be written to prove stability formally [28] [32] [34].

2.2.3 PD+ Control

An extension of PD control with gravity compensation is PD+ Control. This is a form of PD Control that utilizes model information to implement feedback linearization, containing compensation terms for inertial and coriolis/centrifugal forces. The equation for such a controller is given by:

$$\tau = -K_p\, q_e - K_d\, \dot{q}_e + M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + G(q) \tag{2.17}$$

When substituting this equation into 2.5, the closed loop system's equation becomes:

$$M(q)\ddot{q}_e + C(q,\dot{q})\dot{q}_e = -K_p\, q_e - K_d\, \dot{q}_e \tag{2.18}$$

This controller and its properties are widely explored in literature [28] [35]. This controller can also be proven to be asymptotically stable [28]. It may also be noted that this controller for $\dot{q}_r = 0$ and $\ddot{q}_r = 0$ is equivalent to a PD controller with gravity compensation (equation 2.16).

2.2.4 Computed Torque Control

Computed Torque Control is another control architecture that utilizes compensation terms. The equation for this controller is given as:

$$\tau = M(q)\left[\ddot{q}_r - K_p\, q_e - K_d\, \dot{q}_e\right] + C(q,\dot{q})\dot{q} + G(q) \tag{2.19}$$

The computed torque control law compensates for gravitational, inertial, and coriolis terms. When substituting equation 2.19 into equation 2.5, we obtain:

$$\ddot{q}_e + K_p\, q_e + K_d\, \dot{q}_e = 0$$

The computed torque control law is analyzed much more in depth in literature [28]. The controller can also be proven to be stable by performing a Lyapunov analysis.

2.2.5 A Comment on the Selection of the Gravity-Compensated PD Controller

We have discussed a variety of extensions of the PD Torque Controller in this section. To develop our own controller, we use the gravity compensated PD controller as the baseline. This is because the gravity compensated PD controller is the simplest of the discussed control schemes. While energy-based control schemes necessarily require model information about the mass matrix, choosing the gravity-compensated PD controller means that information about the coriolis/centrifugal matrix is not needed for the implementation of a gravity-compensated adaptive PD control law.

2.2.6 PD Gains and Robot Energy

Let us consider the equations for the torque of a PD controller with gravity compensation. If the current configuration (joint positions) of the robot is given as q, then for a reference position (or desired configuration) q_r, the torque is given as:

$$\tau = -K_p\, q_e - K_d\, \dot{q}_e + G(q) \tag{2.20}$$

$$q_e = q - q_r \tag{2.21}$$

Let us also take a look at the energy of the system. Such a system's energy is described as the sum of the Kinetic Energy (T) and Joint Potential Energy (U_q) [3] [29] [28]. The joint potential energy is a function of the error between the current position and reference position q_e. The proportional gain K_p can be thought of as analogous to stiffness, since it contributes a torque proportional to the error term q_e in the controller equation. We can hence define a joint potential energy as a function of the stiffness K_p, and the error term. This controlled potential energy corresponds to elastic potential energy stored in a linear spring due to displacement from its equilibrium position. The idea is that when there is a difference between the desired joint position and actual joint position of the robot, the robot possesses elastic potential energy due to the stiffness matrix (the proportional gain matrix). Remember that since the controller is compensating for

gravity at all times, the gravitational potential energy may be ignored:

$$T(\dot{q}, M(q)) = \frac{1}{2}\, \dot{q}^\intercal\, M(q)\, \dot{q} \tag{2.22}$$

$$U_q(q_e, K_q) = \frac{1}{2}\, q_e^\intercal\, K_p\, q_e \tag{2.23}$$

$$\tag{2.24}$$

$$\mathcal{L} = T + U_q \tag{2.25}$$

$$= \frac{1}{2}\dot{q}^\intercal\, M(q)\, \dot{q} + \frac{1}{2}\, q_e^\intercal\, K_p\, q_e \tag{2.26}$$

2.3 Compliant Control Algorithms

Compliant control algorithms generally refer to controllers that can shape the mechanical impedance or admittance of the robotic system. For robots, this generally means shaping the dynamic relation between the external force, and the robot's position/velocity [2]. This is usually done by modeling the robot's joints as a 'mass-spring-damper' system. We can then modify the desired stiffness and damping values for all joints, to exhibit the desired behavior.

2.3.1 Basic Terminology

As explained in [2], we can define compliance simply by taking a look at an equation modeling a mechanical system with a single degree of freedom. Let's suppose that the sum of external forces on this system is denoted by f_e. The governing equation is given by the following ordinary differential equation:

$$f_e = g(x, \dot{x}, \ddot{x}) \tag{2.27}$$

Where $g(x, \dot{x}, \ddot{x})$ is a function of the position, velocity, and acceleration that defined= the relationship of the sum of external forces to the system. We can then define the stiffness of such a system as the relationship between the external forces and the position of the system, at any given position x_0 as:

$$k = \left.\frac{f_e}{\partial x}\right|_{x=x_o} = \left.\frac{\partial g(x, \dot{x}, \ddot{x})}{\partial x}\right|_{x=x_o} \tag{2.28}$$

where k is the stiffness of the system. The stiffness of a system is the ability to oppose displacement if a force is applied. Mechanical compliance is hence defined as the inverse of the stiffness

$$c = k^{-1} \tag{2.29}$$

Compliance is the ability to exhibit displacement if a force is applied. It is also important to note that while stiffness and compliance refer to the static force-displacement relations, impedance and admittance refer to the dynamic force-displacement relations. In the frequency domain (or s-domain), the impedance - $I(s)$ - and admittance - $A(s)$ - of the system is defined as follows [2]:

$$I(s) = \frac{F_e(s)}{X(s)} \quad , \quad A(s) = \frac{X(s)}{F_e(s)} = I(s)^{-1} \tag{2.30}$$

2.3.2 Position Control and Force Control

When controlling a robotic manipulator, there may be considered to exist two main approaches - position control, and force control [2].

In a position controlled robot, the robot's joint positions are directly controlled. Any desired position would be translated directly to the joint positions, and servoing would mean that we are directly controlling joint positions [1]. In contrast, a force controlled robot is one where there is direct control over the forces (usually the end-effector forces) of the robot. In such a case, there is usually a reference torque to be tracked, and desired torques are produced for each joint via a control law. There are several variations within force control - direct force control refers to a force controller with a force feedback loop, whereas indirect force control refers to schemes that utilize force sensors in conjunction with position control to control force/position relationships - namely impedance and admittance.

2.3.3 Hybrid Position/Force Control

This hybrid approach is presented in its current form by M.H. Raibert and J. J. Craig [36]. The approach is presented in [36] specifically for robot arms with force sensors at the end-effectors of the manipulator. However, here we discuss a slightly modified version as presented in [1], which generalizes this approach to all control schemes involving force feedback. The major difference in this scheme as compared to [36] is the absence of explicit coordinate transforms.

In this schema, each degree of freedom of the manipulator is either force-controlled or position-controlled,

depending on the task constraints. Task constraints can be either natural (referring to physical constraints due to the mechanical or geometric nature of the task) or artificial (these are constraints due to the desired trajectory or motion of the manipulator). Motion-constrained directions must be force-controlled to have control over the force in that particular direction, whereas force-constrained directions will need to be motion-controlled to control the motion in that direction precisely. Each joint is hence either force-controlled, or position-controlled, depending on the task at hand. This selection between control schemes is done using a diagonal Matrix S, as is defined below:

$$[S] = diag(s) = \begin{bmatrix} s_1 & & & 0 \\ & s_2 & & \\ & & \cdots & \\ 0 & & & s_N \end{bmatrix}, \text{ where } \begin{cases} s_i = 1 & \text{Position Controlled Dimension} \\ s_i = 0 & \text{Force Controlled Dimension} \end{cases} \tag{2.31}$$

The matrix defines which orientations are to be force-controlled, and which directions are to be position-controlled. Figure 2.1 (taken from [1]) illustrates this hybrid position/force control scheme, where a few of the joints are position controlled (X_R is the reference position matrix), and the rest are force controlled (F_R is the reference force matrix). E denotes the identity matrix of size N, I_N.

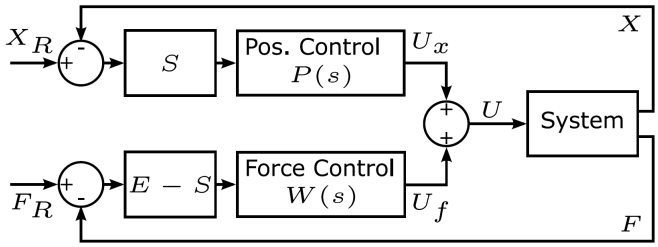

Figure 2.1: Hybrid Position/Force Control Schema [1]

However, as is mentioned in [2], this scheme is not of great interest when considering compliant control algorithms, as it essentially segregates the problem into two separate ones instead of monitoring both the force and position of each orientation - each joint at any given point is either being force-controlled with motion constraints, or motion controlled with force constraints. The force and position of each joint is not considered simultaneously. Hence, this scheme does not guarantee compliant behavior. Instead, [2] argue that schemes of greater interest are the ones that monitor both the joint position and force applied by the joint. These schemes - namely admittance and impedance control - are discussed in the next subsections.

2.3.4 Admittance Control

An example admittance control scheme is shown in figure 2.2. In this scheme, we have an inner loop that tracks the position of the system (with θ_r being the reference position), using any position controller $P(s)$. We also have an outer loop, that takes the external torques τ_e as an input, and uses the 'admittance shaping' block ($\frac{A(s)}{s}$) to shape the mechanical admittance of the system.

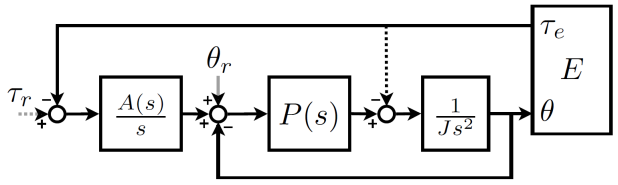

Figure 2.2: Admittance Control Schema [2]

Here, $\frac{A(s)}{s} = \frac{1}{d_d s + k_d}$, where d_d and k_d are desired damping and stiffness. τ_r and τ_e are the reference torque and environmental torque (external forces) respectively. The block labeled $\frac{1}{Js^2}$ denotes the model of the motor, and the dotted arrow denotes the effect of backdrivability, if it is present in the motors. As the environment forces are part of the feedback through the force loop, virtual backdrivability is created. This scheme is quite robust due to the inner position loop (the goal of the inner loop is to be fast, while the outer force loop shapes overall admittance). However, when low impedance is desired, high gains for $\frac{A(s)}{s}$ are required.

2.3.5 Impedance Control

Impedance control is the dual to the admittance control. The most general form of the impedance controller can be thought of as aiming to shape the impedance of the end-effector of a manipulator with respect to externally applied forces [32] [28]. The control objective is to achieve a dynamical relationship of the form:

$$\Lambda_d \ddot{x}_e + D_d \dot{x}_e + K_d x_e = F_{ext} \tag{2.32}$$

Here, x refers to the robot's task-space (end-effector) coordinates, and Λ_d, D_d, and K_d are the symmetric and positive definite matrices of the desired inertia, damping, and stiffness respectively. F_{ext} in this equation denotes the external forces applied to the end-effector of the robot. Note that the robot model discussed in equation 2.5 can also be written in task-space coordinates:

$$\Lambda(x)\ddot{x} + \mu(x, \dot{x})\dot{x} + F_g(x) = F_\tau + F_{ext} \tag{2.33}$$

The specific conversions between joint-space and task-space robot matrices are discussed in [32]. The F_τ (or end-effector torque vector) that gives us the desired dynamics from equation 2.32 can be shown to be given by:

$$F_\tau = F_g(x) + \Lambda(x)\ddot{x}_d + \mu(x,\dot{x})\dot{x} - \Lambda(x)\Lambda_d^{-1}(K_d x_e + D_d \dot{x}_e) + (\Lambda(x)\Lambda_d^{-1} - I)F_{ext} \tag{2.34}$$

The joint torques (τ) that give us this desired impedance can be computed by using the robot's Jacobian ($J(q)$):

$$\tau = J(q)^\intercal F_\tau \tag{2.35}$$

It is noted that this form of Impedance Controller is also called Explicit Impdance Control, since it requires feedback of the external forces being applied to the robot. This is a consequence of 'inertia-shaping', or having a desired robot inertia Λ_d that is different from the actual robot inertia. To avoid inertia-shaping, it is common practice to set the desired inertia equal to the robot's inertia matrix. This condition also helps us avoid explicit feedback of the external forces F_{ext}, since that term actually gets canceled out in equation 2.34.

An example impedance controller setup is shown in figure 2.3. Impedance control has an inner loop that tracks the generated torques for the system (with τ_r being the reference torque), using any force controller $F(s)$. We also have an outer loop, that takes the current configuration (θ) as an input, and uses the 'impedance shaping' block ($sI(s)$) to shape the mechanical impedance of the system.

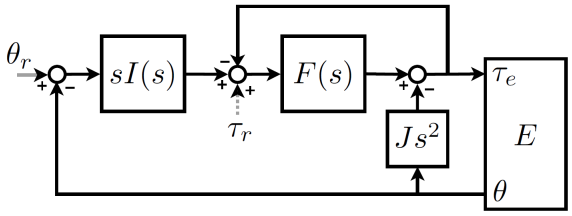

Figure 2.3: Impedance Control Schema [2]

Here, $sI(s) = d_d s + k_d$, where d_d and k_d are desired damping and stiffness. τ_r and τ_e are the reference torque and environmental torque (external forces) respectively. The block labeled $\frac{1}{Js^2}$ denotes the model of the motor. Due to the inner force control loop there is no need for backdrivability, and the inner force loop guarantees a highly accurate actual impedance. However, if high impedance is required, we do need a high gain for $sI(s)$. There is hence usually a tradeoff between tracking accuracy and high impedance. If a high impedance is desired, the outer loop leads to stiffening behavior whereas the inner loop leads to softening

behavior which could lead to antagonistic behavior.

2.3.6 Limitations of Traditional Compliant Controllers

As discussed in this section, traditional compliant controllers do succeed in exhibiting the desired impedance and admittance behavior for pHRI [2] [19] [1]. However, these algorithms require very accurate feedback for external torques. We not only need to know the magnitude of the torque, but the direction as well with great accuracy for the scheme to work. When the sensing data for external torques might not be adequate, there are estimation techniques that might be used for force estimation, such as the generalized momentum observer [37] [38]. This requirement for sensing can be a limiting factor in the controller's applicability in a wide variety of situations.

For torque-controlled (or force controlled) robots, there exist implicit control schemes [2] [1] that do not necessarily require explicit force sensing. In general, the problem of safety still persists. The desired impedance and admittance of the system may be achieved via the presented control methods, but such control schemes might not eliminate all risks. [3] reviews the safety certification process for traditional robots, and they conclude that such traditional control methods do not in general guarantee safe robot behavior. The ISO/TS 15066 standards [27] specify certain safety standards for humans and robots working in close proximity. Specifically, they specify biomechanical limits on the maximum permissible force and pressure values for each human body part. Works such as [3] and [23] show that these maximum force and pressure values are easily translatable to maximum permissible values of energy transfer for each human body part. Researchers have been looking for alternate control schemes for example, that may 'leverage the system's energy'. The next section will explore literature on the topic.

2.4 Energy-based approaches in Literature

The energy of the robot is a quantity that is particularly useful when analyzing both the stability of the controller and human safety during pHRI. We will first take a look at how the robot's energy is generally defined in literature, and then explore how the energy has been used in literature for both stability and safety considerations.

2.4.1 Defining the Robot's Energy

We first reiterate the general form of a robot's dynamic equation [29]:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q)$$

In general, for a controlled robot, the total energy of the robotic system is defined as follows:

$$\mathcal{L} = T(\dot{q}, M(q)) + U_q(q_e, K_q) + U_g(q) \tag{2.36}$$

Each of the terms in the equation is explained as follows:

- $T(\dot{q}, M(q))$ - Refers to the Kinetic Energy of the system, and is a function of joint velocities and the mass matrix.
- $U_q(q_e, K_q)$ - Refers to the Joint Potential Energy, due to the controller. It is a function of the error between the desired position and the actual position of the joints ($q_e = q - q_r$), and the stiffness matrix K_q.
- $U_g(q)$ - Refers to the Gravitational Potential Energy for the robot. As noted in [3], gravity needs to always be compensated in the case of pHRI, since we need to avoid the risk of the robot collapsing. Gravity Compensation is often also important for stability proofs as seen in [29] and [28].

Since it is almost always a good idea to have gravity compensation within your robot [28] [35] [39], we generally deal with a modified version of the robot's energy - one without the gravitational potential energy, since it is being compensated by the controller:

$$\mathcal{L} = T(\dot{q}, M(q)) + U_q(q_e, K_q) \tag{2.37}$$

2.4.2 The Significance of Robot Energy - A Controls Perspective

The energy of a robot is often useful when considering the stability of a controlled system. Passivity-Based Control (PBC) is an important area of research that explores the stability of systems in terms of their energy. The general idea, as outlined in [40] and [34], is based on the 'supplied' and 'stored' energies for a system.

Let us suppose we have a dynamical system of the form:

$$\dot{x} = f(x, u) \tag{2.38}$$

$$y = h(x, u) \tag{2.39}$$

Where $u, y \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$. x denotes the state vector of the system, u is the input vector, and y is the output vector. Equation 2.38 defines the dynamical system, and equation 2.39 denotes the output of the system. Let us also suppose that there is a 'supply rate' function $r : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ associated with the system. The system is then said to be dissipative if there exists a function $V : \mathbb{R}^n \to \mathbb{R}$ such that:

$$V(x(t)) - V(x(0)) \leq \int_0^t r(u(s), y(s))ds \qquad \forall t \geq 0 \tag{2.40}$$

The function V is the storage function, analogous to stored energy. Such a system is considered input to output passive if $r = u^\intercal y$. An alternate framing of the same inequality can be written by differentiating both sides of (2.40), assuming that V is $\mathcal{C}^1$:

$$\dot{V}(x(t)) \leq r(u, y)ds \qquad \forall t \geq 0 \tag{2.41}$$

As noted in [40], usually u and y i.e. the inputs and outputs are such that their product yields the unit of power - Watts (examples of such input/output pairs include currents and voltages for electrical systems, and force and velocity for mechanical systems). This argument is a powerful tool within controls to assert stability for a system. A stability proof for a PD controller with gravity compensation is actually developed in [34]. In that case as well, the input has the units of force, and the output has the units of velocity.

Energy is often an important concept when dealing with stability in the Lyapunov sense as well. The Lyapunov Stability Theorem [28] [41] [42] [43] [44] again considers a dynamical system, and states the following:

Theorem 2.1. *(Lyapunov Stability Theorem): Let $x = 0$ be an equilibrium point of the dynamical system $\dot{x} = f(x)$. Then, if there exists a continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ such that:*

i $V(0) = 0$

ii $V(x) > 0$ *in* $x \neq 0$ *(V is positive definite everywhere except $x = 0$)*

iii $\dot{V}(x) \leq 0$ *in* $x \neq 0$ *($\dot{V}$ is negative semi-definite everywhere except $x = 0$)*

Then the the equilibrium point $x = 0$ is said to be stable. Moreover, if $\dot{V}$ is negative definite everywhere except $x = 0$ ($\dot{V}(x) < 0$ in $x \neq 0$), then the equilibrium point is said to be asymptotically stable.

In this theorem, the function V is often analogous to the system energy, and it then makes sense that if the energy of a system is always decreasing when not at an equilibrium point and the energy is zero at the equilibrium, then that equilibrium point can be said to be stable. This theorem is again very powerful in analyzing system stability and has been used to analyze robot controllers [28] [41] [35] [39].

We can hence see that energy plays an important role in analyzing the stability of systems. If the energy of a robotic system can be said to be bounded in some way, we can make some observations and comments regarding the stability of that system. Hence, bounding the energy can be a useful idea in terms of controller stability, and is a strong argument in favor of control schemes that can limit the total energy of a system.

2.4.3 Robot Energy - Safety for pHRI

When considering the safety aspects of pHRI, research has generally been done on controlling the forces that can be applied by the robot to the environment [17] [45] [46]. That is indeed the case for traditional compliant controllers, that aim to shape the force-position or force-velocity relationships by shaping the impedance/admittance of the robot. However, this may be disadvantageous from a safety point of view. Forces are a coordinate-dependent quantity, meaning that we need to have sensors that can always detect the direction of maximum force accurately [23]. In general, this means having many six-axis force/torque sensors or a robot skin that can detect forces [47], since even an inadequate number of sensors may underestimate the true force being applied to the environment during physical interaction. The ISO/TS 15066 standards [27] state that the safety for humans in close proximity with robots is defined by limiting the maximum forces and pressures applied by the robot on the human. Even if implicit impedance control schemes are used, they do not directly limit the forces the robot applies to its environment, and these constraints might have to be incorporated as additional constraints.

Energy on the other hand presents an alternative to forces as a means of ensuring safe pHRI. It is a coordinate-invariant scalar quantity. As mentioned in [3], physical interaction between a robot and its environment can be described as an energy flow between the two. The energy transfer between the robot and its environment is detailed in figure 2.4 as well. This is especially useful, since the ISO/TS 15066 standards [27] describe the safety standards for human-robot collaboration in terms of the maximum permissible energy

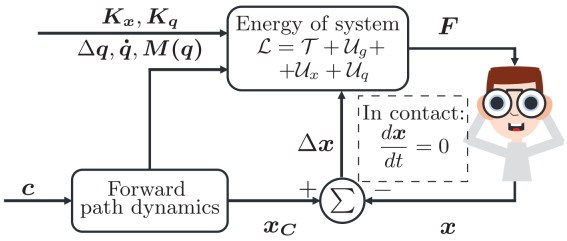

Figure 2.4: The schematic representing human-robot contact [3]. For a control input c, the robot computes a desired position x_c. The energy during an interaction is calculated from the mass matrix $M(q)$, the stiffness matrices (K_x, K_q), the error Δq, the joint velocities $\dot{q}$. The interaction between the human and the robot is modeled as an impedance, where the robot applies a force (F) on the human during interaction, and the human introduces a displacement into the system (x).

transfer, and works such as [23] detail these standards, and how energy limitation can lead to both stable and safe behavior. The ISO/TS 15066 standards [27] establish maximum permissible contact forces and pressures for various body areas, based on pain-sensitivity thresholds. As described in [23], the maximum allowable energy transfer between a robot and the environment ($\mathcal{L}_{max}$) can be calculated using the tables given in ISO/TS 15066. These tables detail the maximum permissible force (f_{max}) and the stiffness or the related effective spring constant (k) for various body parts, and the maximum permissible energy transfer may then be calculated using the following equation:

$$\mathcal{L}_{max} = \frac{f_{max}^2}{2k} \tag{2.42}$$

Both [23] and [3] use equation 2.42 to determine maximum permissible energy transfer for safe pHRI, and this is the dominant method of calculating values for maximum allowable energy transfer. There are other methods of calculating the energy transferred, such as the one discussed in [48]. Regardless of the exact method used to calculate the energy transfer, control methods that limit a robot's energy are emerging as a new way of ensuring safety during pHRI. Energy-based methods for pHRI are a relatively newer area of research. The ISO/TS 15066 was published in 2016, and there are currently only a few research papers directly addressing this research avenue ([23] [3] [48]). [23] and [48] present simulation results for their work, whereas [3] present hardware results for their proposed approach.

Monitoring energy instead of force transferred is also advantageous since force is a coordinate-dependent quantity and would require comprehensive force sensing capabilities, whereas energy-based approaches can use a single quantity to ensure safe pHRI. Because energy has such advantages in terms of determining safety, works such as [23] and [3] detail the process of limiting the energy transfer during pHRI collisions. Both papers utilize the ISO/TS 15066 (2016) [27] standards to limit the robot's energy at all times to ensure

energy-limitation. The authors of [23] utilize the idea of 'virtual energy tanks' (such as the one discussed in [49]) to limit the robot's kinetic energy at all times. This limits the energy transferred to a human during a collision with a robot under the maximum allowable limit. The authors of [3] on the other hand propose modifying an impedance controller's stiffness and damping adaptively to ensure that the total energy of the robotic system (sum of the kinetic and potential energy) stays below the maximum allowable limit.

Works such as [48] aim to analyze the ways in which we calculate the supplied energy during collisions for pHRI. This is done by modeling the human soft tissue as an impedance and the bone as a mass-type object. Collision models are then developed to accurately capture energy transferred to human tissue during collisions. Hence, energy can be seen to be an advantageous quantity when aiming to ensure safe behavior for robots, and is further motivation for energy-based control schemes.

2.4.4 Control Schemes that Leverage Robot Energy

The previous sections detail the two major advantages of considering robot energy for pHRI - stability, and safety. These advantages have been recognized within the literature, and various control schemes have been proposed that utilize the robot's energy.

It has been recognized in the literature that when humans physically interact with their environment, they rarely use fixed impedances, and human behavior in general is modeled much closer by a variable impedance model [49] [50] [45] [46]. There have been attempts at developing control schemes that can offer variable impedance. Such schemes were initially explored in the literature, mainly from the point of view of learning the variable (time-dependent) stiffness and damping parameters for a robot, concerning the task at hand [45] [46]. Little consideration was given to the fact that there are no stability guarantees with simple time-varying impedance parameters - variable stiffness can produce extra energy, which in turn violates the passivity of the controller. This problem was tackled with energy tank-based approaches for variable impedance parameters [49] [50]. In such approaches, the energy being dissipated by the system is tracked and limited using a virtual 'energy tank', so that the dissipated energy is limited even with varying stiffness, and passivity may be preserved. [50] details the various approaches in the literature concerning variable impedance learning and stability strategies. This concept of virtual energy tanks is also utilized in [23], where this idea is used to limit the maximum energy transferred from the robot to the human during collisions.

[3] is a comprehensive work on limiting energy transfer during pHRI, detailing how traditional compli-

ant controllers do not, in general, guarantee safe robot behavior. [3] then propose a control scheme that limits a robot's total energy (equation 2.26) to ensure that the energy exchanged during pHRI does not exceed a certain limit that has been determined to be safe. They achieve this in the following way:

- Scaling down the potential functions (or stiffness matrices) (used for generating the task torque), whenever the robot's energy exceeds the limit - this limits the potential energy of the robot.
- Introduce a quantity called 'effective time', computed as:

$$t_{eff} = \begin{cases} t_{eff} = t_{eff} + t_s, & \text{if } \mathcal{L} \leq \mathcal{L}_{max} \\ t_{eff} = t_{eff}, & \text{otherwise} \end{cases} \tag{2.43}$$

  where t_s is the controller's sample time, $\mathcal{L}$ is the robot's energy, and $\mathcal{L}_{max}$ is the maximum allowable energy limit for the robot. When the robot's energy exceeds the allowable energy limit, the effective time is not updated, and the reference position of the robot consequently does not change.

- A second joint potential $U(q)^*$ is introduced, that is only non-zero when the kinetic energy of the robot exceeds the limit (in a situation where the robot is pushed, for example). This potential acts in the direction of the last position of the robot before the kinetic energy exceeds the limit. Simply put, this potential is activated when the robot's kinetic energy exceeds the maximum allowable limit, and this potential acts to oppose this excessive kinetic energy (to stop the robot from moving at high velocities).

The controller presented in [3] has some characteristics that are desirable in a compliant controller - purely based on the energy and without the need for force sensors, it can exhibit compliant behavior:

- The controller limits the robot's total energy to limit energy transfer upon contact.
- The potential energy is also limited to counter clamping scenarios.
- Joint stiffness is increased if the robot is pushed by the human and the kinetic energy exceeds the limit - the robot is hence not pushed uncontrollably.

These are some desirable features from a pHRI perspective. However, it is worth noting that the control scheme does have a few drawbacks:

- Requires the recalculation of the elastic potentials for both joint space and cartesian space, and an additional elastic potential for countering external pushes. This can get complex, depending on the specific impedance controller implementation.

- The 'effective time' and $q*$ are additional parameters that need to be worked with, which may complicate the implementation.
- No explicit stability proof is presented for the proposed control law.

2.5 Desirable Controller Characteristics for pHRI

The idea of desirable pHRI characteristics is a challenging one. Since each physical interaction between a human and a robot will be different, it is tough to suggest definitive robot behaviors that might be considered safe. However, surveying the literature, there are certain qualities that might define safe robot behavior in the context of pHRI:

1. If a robot collides with a human, the interaction should be safe. The safety criteria is defined in the ISO/TS 15066 standards, and this safety criteria can either be treated as a velocity-limit [23] [51], or as an energy limit [23] [3].
2. When a robot is in a quasi-static contact (clamping) scenario with the human operator, the human operator should be able to push the robot away to free themselves [3] [21] [22].
3. When a robot is pushed away by human (for example to free themselves from a clamping scenario), the robot must not move arbitrarily fast, as that leads to high joint velocities and unstable robot behavior. Hence, the controller must oppose a push that may lead to unstable behavior [3].

These general characteristics are often seen in controllers that aim to make pHRI safe [3] [23]. However, one thing to note is that in the list above, points 2 and 3 are somewhat opposing desired behaviors. We want to be able to push the robot away during a clamping scenario - the robot should not be too stiff or damped so as to not be pushed away. However, once the robot is pushed away, we want the robot to not move in an unstable manner - the stiffness and damping should not be too low which may potentially lead to 'flailing' movement of the robot. The balance between these two scenarios should be struck by a controller for safe pHRI. When analyzing the performance of our proposed solution during the course of this thesis, we shall keep these characteristics in mind.

2.6 Limitations of Current Approaches

There is a great amount of interest and research in the domain of safe and compliant control for pHRI. Impedance and admittance control algorithms, as presented in [2], [19], and [1], succeed in synthesizing a

controller that can react compliantly to external forces. However, they require the accurate measurement of external forces, which in itself presents challenges, since both the magnitude and the direction of the external force needs to be accurately ascertained. This leads to the requirement of sensors that can measure external forces accurately in all directions. A six-axis force/torque sensor may be used for this application, but it will only be able to accurately measure contact forces when the force is applied directly at the sensor's location. Researchers have developed 'artificial skins' that are stretchable and fabric-based sensors that can be used as a layer covering a robotic manipulator, and may be able to detect forces from all directions [24] [47]. Apart from the cost of such setups, the appropriate fitting and 'resolution' of such sensors may present challenges. In addition, for controllers to offer variable impedance parameters (stiffness and damping), stability is often a cause for concern [49].

Energy-based controllers offer a good alternative - they can offer compliance without the need for explicit force sensing and are advantageous from a safety point-of-view as well. However, current energy-based methods do require some degree of complexity to be integrated within the controller. Such controllers also do not have any explicit stability analyses accompanying them. The control schemes such as the ones presented in [3] and [49] present energy-based control schemes. The controller presented in [49] uses energy mainly to establish stability for the controller by ensuring passivity. [3] on the other hand defines a controller that limits the energy of the robotic system, and is coordinate invariant while offering compliance. While their work provides insights into the use of energy limitation to ensure safety, their proposed controller is still more complex than traditional control schemes used within robotics. Their proposed controller utilizes potential fields to to describe the desired torque. There is a need for constant recalculation of the potential and damping functions, which can range from fairly simple to quite complex depending on the method for calculating the stiffness and damping matrices. The implementation of the 'effective time' quantity is also needed for the controller, which incorporates some motion planning ideas. While the controller certainly achieves energy limitation, the scheme is reactive rather than proactive - the torque is modified only when the energy exceeds the pre-defined energy limit. This can lead to jumps in the robot's torque, when the robot's energy approaches the pre-defined energy limit.

Energy limitation for the robot in a proactive manner could be beneficial since there would be no jumps in the robot's torque when approaching the pre-defined energy limit. However, to our knowledge, there is no controller in literature currently that can perform energy limitation in a proactive manner. It would also be advantageous to have a controller that could achieve this using a simple implementation. The controller that we propose in the following chapters aims to meet some of the gaps in the literature. We propose

an adaptive Proportional-Derivative (PD) Controller that can limit the robot's total energy, hence limiting energy transfer upon contact with the environment. The controller can also limit the potential energy under the energy budget, to reduce clamping dangers. Finally, the controller can oppose excessive external forces to limit the kinetic energy that may be induced from a push to the robot, hence reducing the risk of uncontrollable flailing for the robot [23].

The advantage of our adaptive PD controller is that it closely approximates a non-adaptive PD controller when the robot is not approaching the energy limit. The controller starts off with some initially tuned values of the proportional and derivative gain, the gains remain fairly close to these values as long as the energy is safely below the limit. This means that for these initial or 'nominal' gains of the controller, traditional PD tuning methods may be employed - traditional methods from classical controls can come in handy. Some of the novelty of our work is also in the fact that we perform a stability analysis, and obtain a stability condition for the controller. We then experimentally validate the proposed controller on the TALOS Robot (by PAL Robotics) [52], and verify the expected energy-limitation and compliant behavior.

Chapter 3

The Proposed Adaptive PD Controller

As seen in the literature, energy-based control is a promising alternative to traditional compliant controllers. However, the current energy-based controllers are relatively complex as compared to traditional controllers. PD controllers on the other hand are fairly simple control laws that have been widely used within robotics. Gravity-compensated PD control laws are some of the simplest control laws with stability guarantees within robotics. PD controllers also have a quite a few supporting techniques from classical control - literature on the tuning and ratios of PD gains is widespread. We hence propose an adaptive PD controller that can perform energy limitation for safe pHRI. We take the energy-limitation concepts and extend them to PD controllers, so that alongside the benefits of energy-based controllers, we may also be able to leverage some classical controls concepts related to PD controller design.

This chapter introduces the equations for our proposed adaptive PD controller, and the rationale behind some of the design choices we make.

3.1 The Controller

In this section, we introduce some of the functions, equations, and variables that we developed for our proposed adaptive PD controller. We discuss the reasons for our design choices, and illustrate the behavior of the controller that we develop.

3.1.1 The Setup

Here we introduce some of the very initial setup and terminology for our controller. This is needed to not only introduce our proposed controller more effectively, but also to draw comparisons with a non-adaptive PD controller.

Let's say we tune the PD gains for an n-DOF PD controlled system using any tuning method. The proportional and derivative gains are then denoted as $K_{po_{diag}}$ and $K_{do_{diag}}$ respectively, where $K_{po_{diag}}, K_{do_{diag}} \in \mathbb{R}^{n \times n}$ are $n \times n$ diagonal matrices. These 'nominal' gains will serve as our starting point. These gains shall be the ones we want to have when the system energy is well below the energy limit.

We also define the following two operations for forward and inverse diagonalization:

$$A = \begin{bmatrix} a_{11} & 0 & .. & 0 \\ 0 & a_{22} & .. & 0 \\ 0 & 0 & .. & a_{nn} \end{bmatrix} \tag{3.1}$$

$$B = \begin{bmatrix} a_{11} \\ a_{22} \\ .. \\ a_{nn} \end{bmatrix} \tag{3.2}$$

$$\tag{3.3}$$

$$diag(B) = A \tag{3.4}$$

$$diag^{-1}(A) = B \tag{3.5}$$

We can then define the column vectors for the nominal PD gains as follows:

$$K_{po} = diag^{-1}(K_{po_{diag}}) \tag{3.6}$$

$$K_{do} = diag^{-1}(K_{do_{diag}}) \tag{3.7}$$

3.1.2 The r function

Let's suppose we have an upper limit on the system's energy, denoted by $\mathcal{L}_{lim}$. Our first contribution is to define a dissipative function - r - as follows:

$$r = \frac{1}{1 + \left(\frac{\mathcal{L}}{\beta}\right)^{\alpha}} \tag{3.8}$$

Here, β is any cutoff energy point ($\beta < \mathcal{L}_{lim}$), close to which the PD gains start to change. The constant α is a tunable parameter ($\alpha > 1$), that defines how sharp the cutoff is. Figures 3.1 and 3.2 practically illustrate the effect of α and β values on this function. The idea is to have a dissipative function that can shape the behavior of the PD gains as the system's energy changes. We ideally want PD gains that remain close to the nominal value when the system energy is low, and we want the gains to be close to zero as the system's energy approaches the limit. We hence develop this r function that can be multiplied with the gains to achieve this dissipative behavior. The value of r is close to one when the system energy is low, and it tends towards zero as the system's energy increases.

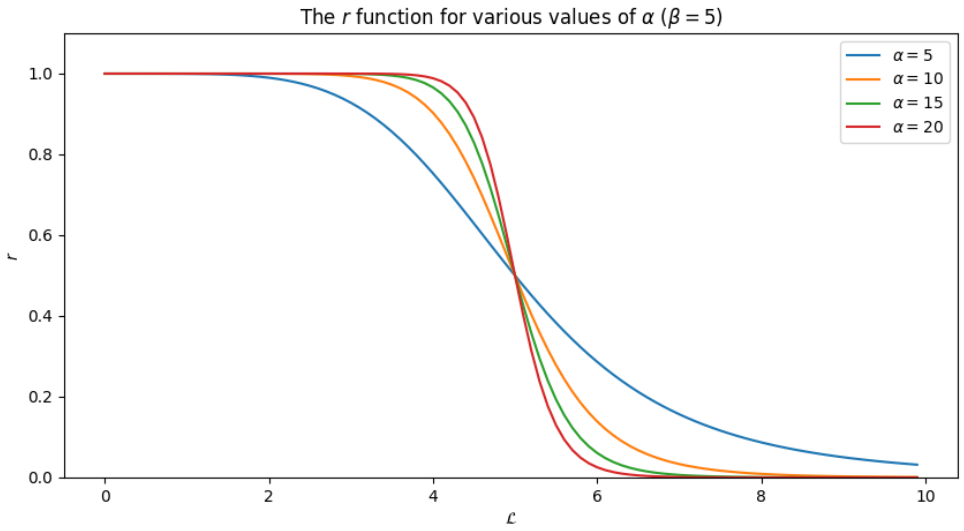

Figure 3.1: The r function for various α values

Note that this dissipative function does not explicitly depend on $\mathcal{L}_{lim}$. The role of this function is to define the general behavior of the PD gains as we approach the energy limit, and the aim is to select the values of α and β such that the value of r is close to zero when approaching the energy limit. The α and β values are

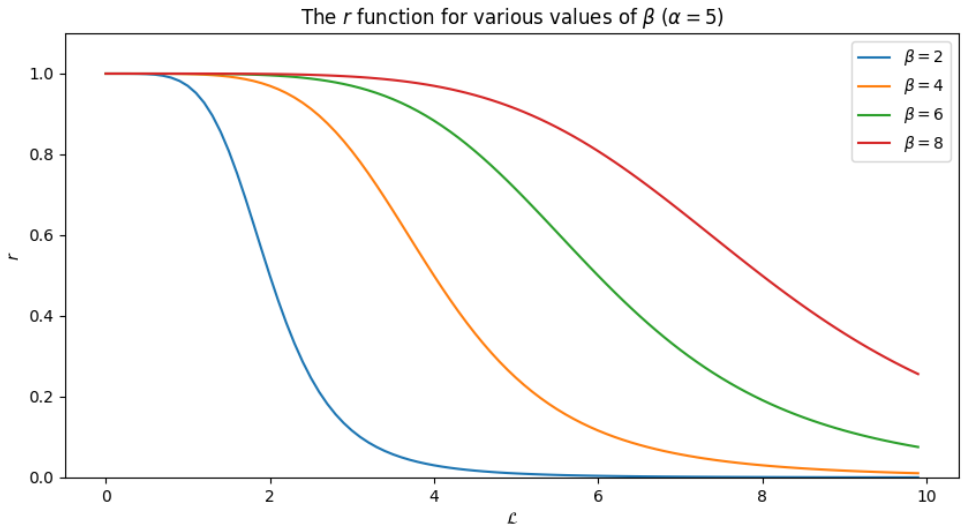

Figure 3.2: The r function for various β values

thus important tuning parameters. However, we recognize that there may be certain ill-chosen values of α (α value too low) and β (β value too close to $\mathcal{L}_{lim}$) that may lead to some high values of r even as we approach the energy limit. To tackle this, one possibility was to define the r function as a piecewise function, where the function becomes 0 when the system's energy exceeds the limit. However, we choose not to do that in order to keep all of the functions related to the controller continuous. Instead in the next subsection, we will discuss an additional term that will compensate for this effect by considering additional dissipative terms.

The proportional gain K_p

Now that we have our dissipative function r defined, the next step is to incorporate this function into our control law. To do this, we first consider the proportional gain in this subsection. For a given energy limit $\mathcal{L}_{lim}$, the adaptive proportional gain K_p for our proposed controller is denoted as the dot product (and subsequent diagonalization) of the terms p_1 and p_2:

$$K_{pc} = p_1 \cdot p_2 \tag{3.9}$$

$$K_p = diag(K_{pc}) \tag{3.10}$$

Where the terms p_1 and p_2 are obtained using the following functions:

$$p_1 = [rK_{po}] + \left[(1-r)\frac{\mathcal{L}_{lim}}{n}\right] \tag{3.11}$$

$$p_2 = \frac{1}{1+(1-r)q_e^2} \tag{3.12}$$

Where n is the degrees of freedom of the system. We will now discuss both of these functions in turn.

The first square bracket for p_1 multiplies the nominal proportional gain $K_{po_{diag}}$, with the dissipative function r. This results in the proportional gains being dissipated as the system's energy rises. The bracket on the right hand side for the p_1 function multiplies $(1-r)$ with the $\frac{\mathcal{L}_{lim}}{n}$. The rationale for the right hand side bracket being added to the p_1 function is that when the energy of the system is low, the value of r is close to one, and the value of the p_1 function is close to the nominal gain $K_{po_{diag}}$. However as the system's energy approaches the limit, the value of r is close to zero, and the value of the p_1 function approaches $\frac{\mathcal{L}_{lim}}{n}$.

The vector p_2 is a dissipative term, depending on the error between the desired position and current position ($q_e = q - q_r$). Expanding on the function p_2, it can be made clearer by elaborating as follows:

$$q_e = \begin{bmatrix} q_{e\_1} \\ \\ q_{e\_2} \\ \\ \dots \\ \\ q_{e\_n} \end{bmatrix} \tag{3.13}$$

$$p_2(r, q_e) = \begin{bmatrix} \frac{1}{1+(1-r)\cdot q_{e\_1}^2} \\ \\ \frac{1}{1+(1-r)\cdot q_{e\_2}^2} \\ \\ \dots \\ \\ \frac{1}{1+(1-r)\cdot q_{e\_n}^2} \end{bmatrix} \tag{3.14}$$

The first purpose such a term serves is to compensate for values of r that may be too high when approaching the system's energy limit - this is important if α and β values are not chosen suitably, since the 'r' function will have a high value when approaching the system's energy and the energy of the system may not be completely limited. The second advantage of having this term is that when the system's energy approaches the limit, r value approaches zero and the value of the adaptive proportional gain approaches:

$$K_{pc} = \left[\frac{\mathcal{L}_{lim}}{n}\right] \frac{1}{1+q_e^2}$$

This aim of such a setup is to have the proportional gain be as high as possible while not violating the limit.

The derivative gain K_d

In this subsection, we consider the derivative gain. For a given energy limit $\mathcal{L}_{lim}$, the adaptive derivative gain K_d for our proposed controller is denoted as the dot product (and subsequent diagonalization) of the terms d_1 and d_2:

$$K_{dc} = d_1 \cdot d_2 \tag{3.15}$$

$$K_d = diag(K_{dc}) \tag{3.16}$$

Where the terms d_1 and d_2 are defined using the following functions:

$$d_1 = r \cdot K_{do} \tag{3.17}$$

$$d_2 = \frac{1}{1+(1-r)\cdot \dot{q}^2} \tag{3.18}$$

We will now discuss both of these functions in turn.

Just like in the proportional gain case, the term d_1 multiplies the nominal proportional gain $K_{do_{diag}}$, with the dissipative function r. This results in the derivative gains being dissipated as the system's total energy rises. Notice that there is no term here dealing with the energy limit. This is because when the system's total energy is close to the energy limit, we want most of the energy of the system to be in the form of potential energy rather than kinetic energy. However in the case of the derivative gain, that term is no longer needed.

The vector d_2 is a dissipative term, depending on the joint velocities ($\dot{q}$). Expanding on the function d_2

in a similar fashion as we did with p_2, it can be made clearer as follows:

$$\dot{q} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dots \\ \dot{q}_n \end{bmatrix} \tag{3.19}$$

(3.20)

$$d_2(r, q) = \begin{bmatrix} \frac{1}{1+(1-r)\cdot\dot{q}_1^2} \\ \frac{1}{1+(1-r)\cdot\dot{q}_2^2} \\ \dots \\ \frac{1}{1+(1-r)\cdot\dot{q}_n^2} \end{bmatrix} \tag{3.21}$$

3.1.3 Limiting User-Added Energy

We now have a controller that will not violate the energy limit on its own. This is because as soon as the system's energy limit approaches, the gains of the PD controller adaptively dissipate. As soon as the energy approaches the limit, this controller will adaptively dampen the PD gains, so as to limit the energy. This is done in two major ways:

- As the energy approaches the limit, the decrease in the proportional gain K_p (equations 3.9, 3.11, 3.12) means a direct decrease in potential energy.
- As the energy approaches the limit, both the proportional and derivative gains decrease (equations 3.15, 3.17, 3.18), leading to a smaller desired torque, limiting the kinetic energy.

All the while, the controller will also try and keep the gains as close to nominal as possible without violating the limit. Hence the controller we have defined in the previous section has the effect of damping the nominally tuned PD gains in an adaptive manner. As the energy approaches the limit, these gains are dampened in order to keep the energy of the system under the limit. The controller we have described so far would result in any possible collisions between a robot and a human being safe under the ISO/TS 15066 standards

[27], since the energy transfer between the robot and human would be under the limit. However, the problem of user-added energy persists.

When a robot physically interacts with a human, there is a two-way energy transfer that takes place between them. If it is assumed that it is just the robot transferring energy to the human, the controller described so far would result in safe behavior. However, we still have to consider situations where the human might transfer energy to the robot as well. Such situations are described in [23] and [3]. This situation occurs when the robot is pushed away (during a clamping scenario for example) while tracking a trajectory - if a robot is pushed away from the current trajectory or goal position, the robot gains energy in two ways:

- The increase in velocity due to the push causes a rise in Kinetic Energy.
- The error between current position and desired position increases, leading to a rise in Potential Energy.

Since there is a rise in energy, if we start approaching the energy limit in this manner, the PD gains will tend to decrease, leading to the robot offering little to no resistance to the push. Hence, this 'user-added' energy will lead to the robot potentially violating the energy limit. While the energy limit will never be violated due to the controller itself, this situation involving user-added energy poses a dangerous problem, that the controller should be equipped to deal with. The energy limit should not be violated, even if that is not due to the controller, but because of the external forces.

To deal with this issue, we identify one key factor: if the energy limit is violated due to an external push to the robot, this will be reflected almost entirely in the kinetic energy of the system. This is because if the robot is pushed away and the energy approaches the limit, both the PD gains will adaptively decrease. This is desirable from a pHRI perspective since the torques applied by the robot will decrease with the PD gains. A decrease in the proportional gain has direct consequences on the potential energy, since it is defined as follows:

$$U_q = \frac{1}{2}\, q_e^\intercal\, K_p\, q_e \tag{3.22}$$

This means that when a robot is pushed and the energy limit is violated, the potential energy should be fairly small - the energy violation must be a consequence of high kinetic energy!

We can now try and formulate a solution to this problem. We recognize that high kinetic energy means high joint velocities, which mean a high rate of change of error. This means that the term $\dot{q}_e$ must be high as

well. Hence, increasing the derivative gain K_d as the kinetic energy approaches the limit can offer a great way of resisting the push, without violating the energy limit - this is equivalent to increasing the damping in case of high external forces, to resist uncontrollable motion of the robot. We hence define a new dissipative term r_T as follows:

$$r_T = \frac{1}{1 + \left(\frac{T}{\beta}\right)^{\alpha}} \tag{3.23}$$

Where α and β are the same terms defined previously, and $K_{d_{max}}$ is a column vector, that sets an upper limit on how high we want the derivative gains to be when resisting a push. Note that the r_T function is the exact same function as r, but is a function of the kinetic energy specifically (T) rather than the total energy of the system. This means that when the kinetic energy of the system is low, the value of this r_T function will stay close to one. However as the kinetic energy of the system approaches the limit, the value of this r_T function will approach zero. We can hence use the term $(1 - r_T)$ and multiply it with the maximum values of the derivative gain $K_{d_{max}}$. Thus when the kinetic energy of the system is approaching the system's defined energy limit, the derivative gains increase to oppose this high kinetic energy. Note that it is important that this opposition to the user-added energy is in the form of damping rather than elastic potential. The external force by the human is not causing a reactionary force to try and reduce tracking error. Instead, the opposition is due to the increased damping of the arm, meaning that high joint velocities are directly being addressed. While this may not seem like an obvious advantage, we avoid high reactionary forces due to high stiffness using this alternative approach.

$K_{d_{max}}$ is again a tunable parameter, with a higher value of $K_{d_{max}}$ corresponding to a higher degree of resistance to the push, and vice versa. We also want to set a maximum value for the derivative gains, since the main purpose of this term is to oppose uncontrollable movements of the robot in response to external pushes. An arbitrarily high value could result in extremely high resistance to external movements which is also not desirable. As noted in [3], it is not that we do not want the robot to move when pushed. For example if a clamping scenario occurs, we want the human to be able to push the robot away. It is the potential uncontrollable movement of the robot that is undesirable, and the aim of this added term is more to provide some damping when pushed away, rather than opposing pushes altogether.

Finally, we can denote our new adaptive derivative gain to be:

$$d_3 = (1 - r_T) \cdot K_{d_{max}} \tag{3.24}$$

$$K_{dc} = (d_1 \cdot d_2) + d_3 \tag{3.25}$$

$$K_d = diag(K_{dc}) \tag{3.26}$$

And just to reiterate, the desired torque equation will be given by utilizing these adaptive PD gains in the gravity-compensated PD controller for a torque controlled system 2.16. The final torque will hence be computed as:

$$\tau = -K_p \, q_e - K_d \, \dot{q}_e + G(q)$$

3.1.4 Summary of the Formulated Controller

This concludes the entire formulation of our proposed adaptive PD controller. All of the controller equations (and the subsequent stability analysis equations mentioned in the next Chapter 4) have been summarized in Appendix A. A very general summary of the controller response is shown in table 3.1.

The proposed controller succeeds in achieving two of the research objectives we have set out - the controller (for now, at least theoretically) limits the robot's energy under any given budget. We have a few tuning parameters for tailoring the controller response as well. The proposed system is an extension of a gravity compensated PD controller for a torque controlled system. Since this system, its characteristics, and its tuning are widely studied in literature, the controller offers a simpler path to implementation. As is summarized in table 3.1, the controller offers some desirable characteristics from a pHRI perspective. However we are still to address the other two research objectives - we need to perform a stability analysis of the controller, and we need to test the controller both in simulation and on hardware. The next chapter analyzes the controller's stability, and details the conditions for the controller's guaranteed stability. Chapters 5 and 6 elaborate on the practical implementation of this proposed controller, and the results we obtain to verify the expected behavior of the controller.

Energy Limitation Measures - The Proposed Adaptive PD Controller		
Energy Status	Energy Limitation Measures	Expected Robot Response
Energy Safely Below Limit	PD Gains remain Nominal	Acts as a traditional PD controller with gravity compensation for a torque-controlled system
Potential Energy Approaching the Limit (the robot is slowly pushed away from its tracking objective/an obstacle is encountered while the reference position is being tracked)	Both Proportional and Derivative Gains are Reduced (Reduced Stiffness and Damping)	The robot displays 'soft' and slower behavior, trading off precise trajectory tracking but reducing risk of large reactionary forces
Kinetic Energy Approaching the Limit (the robot is pushed away with a high force from its tracking objective)	Proportional Gains are Reduced, Derivative Gains are Increased (Reduce Stiffness and Increased Damping)	The stiffness is decreased, and precise tracking of the trajectory is traded off. Instead, high damping reduces the robot's kinetic energy, preventing uncontrollable movements.

Table 3.1: Expected Behavior of the Proposed Adaptive PD Controller for Energy Limitation

Chapter 4

Stability Analysis for the Proposed Controller

In this chapter, we take a look at the stability of the proposed adaptive PD controller we introduced in Chapter 3. This is an important aspect of developing control laws - having a guarantee of stability means there is a guarantee of the controller achieving its objective. Stability is also important from a pHRI perspective. Unstable controllers can behave unexpectedly (such as potential oscillations or erratic movements) that could have harmful implications for pHRI. We start off by taking a look at the stability of a system with a generic gravity-compensated PD control law, and we then move on to the adaptive case. Finally, we obtain a condition for the stability of the controlled system, and propose an algorithmic way of incorporating the stability condition for our controller's implementation.

4.1 Stability Proof for a Torque-Controlled System with a Gravity-Compensated PD Control Law

The equations for the torque for a generic PD controller with gravity compensation was introduced in Chapter 2:

$$\tau = -K_p\, q_e - K_d\, \dot{q}_e + G(q) \tag{2.16}$$

where $q_e = q - q_r$ represents the error between the reference trajectory and the actual positions of the joints, and K_p and K_d are positive definite matrices. Let us also take a look at the dynamic equation of a robotic

system, which has also been introduced in Chapter 2:

$$\tau = M(q)\ \ddot{q} + C(q, \dot{q})\ \dot{q} + G(q) \tag{2.5}$$

To obtain an equation that describes the system's closed loop behavior - behavior of the system with feedback - we can substitute equation 2.16 into equation 2.5 by equating the torques:

$$M(q)\ \ddot{q} + C(q, \dot{q})\ \dot{q} + G(q) = -K_p\ q_e - K_d\ \dot{q}_e + G(q)$$
$$M(q)\ \ddot{q} = -K_p\ q_e - K_d\ \dot{q}_e - C(q, \dot{q})\ \dot{q}$$

One other simplification we can make is that if the reference position q_r is a constant, then:

$$q_e = q - q_r$$
$$\dot{q_e} = \dot{q}$$

This is an assumption that often can be made for proving controller stability [28], since this proves controller stability for a constant position tracking problem. If the reference position is not a constant ($q_r(t)$), then it can be shown that the position error q_e becomes asymptotically small if the PD gains K_p and K_d are sufficiently large [28] [53]. However, that discussion is outside the scope of the current discussion.

The overall equation of the system can then be written as:

$$M(q)\ \ddot{q} = -K_p\ q_e - K_d\ \dot{q}_e - C(q, \dot{q})\ \dot{q}_e$$

Finally, an equation for the acceleration vector of the joints, $\ddot{q}$, may be obtained:

$$\ddot{q} = M^{-1}(q)\,(-K_p\ q_e - K_d\ \dot{q}_e - C(q, \dot{q})\ \dot{q}_e) \tag{4.1}$$

4.1.1 Stability in the Lyapunov Sense

Let us take the following example Lyapunov candidate function:

$$V = \frac{1}{2}\dot{q}_e^{\intercal}\ M(q)\ \dot{q}_e + \frac{1}{2}\ q_e^{\intercal}\ K_p\ q_e \tag{4.2}$$

Since both the mass matrix $M(q)$ and the proportional gain matrix K_p are positive definite matrices, we can say that this Lyapunov function is positive definite. Now, let us take the time derivative of this function:

$$\begin{aligned}
\dot{V} &= \dot{q}_e^\intercal \; M(q) \; \ddot{q}_e + \frac{1}{2}\dot{q}_e^\intercal \; \dot{M}(q) \; \dot{q}_e + \; q_e^\intercal \; K_p \; \dot{q}_e \\
&= \dot{q}_e^\intercal \; M(q) \; M^{-1}(q) \left(-K_p \; q_e - K_d \; \dot{q}_e - C(q, \dot{q}) \; \dot{q}_e\right) + \frac{1}{2}\dot{q}_e^\intercal \; \dot{M}(q) \; \dot{q}_e + \; q_e^\intercal \; K_p \; \dot{q}_e \\
&= \dot{q}_e^\intercal \; \left(-K_p \; q_e - K_d \; \dot{q}_e - C(q, \dot{q}) \; \dot{q}_e\right) + \frac{1}{2}\dot{q}_e^\intercal \; \dot{M}(q) \; \dot{q}_e + \; q_e^\intercal \; K_p \; \dot{q}_e \\
&= -\dot{q}_e^\intercal K_d \; \dot{q}_e + \frac{1}{2}\dot{q}_e^\intercal \; \dot{M}(q) \; \dot{q}_e - \dot{q}_e^\intercal C(q, \dot{q}) \; \dot{q}_e - \; \dot{q}_e^\intercal K_p \; q_e + q_e^\intercal \; K_p \; \dot{q}_e \\
&= -\dot{q}_e^\intercal K_d \; \dot{q}_e + \frac{1}{2}\dot{q}_e^\intercal \; \dot{M}(q) \; \dot{q}_e - \dot{q}_e^\intercal C(q, \dot{q}) \; \dot{q}_e
\end{aligned} \tag{4.3}$$

Equation 4.3 may seem complicated to analyze at first. However, we can use some of the properties of the robot matrices we discussed in Chapter 2 to simplify this equation:

- $\dot{M}(q) - 2C(q, \dot{q})$ is a skew-symmetric matrix (equation 2.10)
- The quadratic form for a skew-symmetric matrix equals zero (equation 2.11)

We can hence rewrite the equation for the time derivative of the Lyapunov function (equation 4.3) as follows:

$$\dot{V} = -\dot{q}_e^\intercal K_d \; \dot{q}_e \tag{4.4}$$

Since we have already established that K_d is a positive definite matrix, we can infer that $\dot{V}$ is a negative definite matrix, and that the controller stabilizes the system around the point $(q_e, \dot{q}_e) = (0, 0)$.

4.2 Extending Stability to the Proposed Adaptive PD Controllers

In this section, we extend the analysis done in the previous section to the adaptive PD controller we proposed in Chapter 3. Since in this case the proportional and derivative gains are adaptive (time dependent), the equation of the controlled robotic system may be rewritten (from equation 4.1) as:

$$M(q)\ddot{q} = -K_p(t) \; q_e - K_d(t) \; \dot{q}_e - C(q, \dot{q}) \; \dot{q}_e \tag{4.5}$$

Our equation for the acceleration vector of the joints, $\ddot{q}$, now becomes:

$$\ddot{q} = M^{-1}(q) \left(-K_p(t) \; q_e - K_d(t) \; \dot{q}_e - C(q, \dot{q}) \; \dot{q}_e\right) \tag{4.6}$$

4.2.1 Lyapunov Stability

The same candidate Lyapunov function used in equation 4.3 may be used again:

$$V = \frac{1}{2}\dot{q}_e^\intercal\ M(q)\ \dot{q}_e + \frac{1}{2}\ q_e^\intercal\ K_p(t)\ q_e \tag{4.7}$$

Now, let us take the time derivative of this function:

$$\dot{V} = \dot{q}_e^\intercal\ M(q)\ \ddot{q}_e + \frac{1}{2}\ \dot{q}_e^\intercal\ \dot{M}(q)\ \dot{q}_e + \ q_e^\intercal\ K_p(t)\ \dot{q}_e + \ \frac{1}{2}\ q_e^\intercal\ \dot{K}_p(t)\ q_e \tag{4.8}$$

Notice that since the proportional gain (K_p) is now time-dependent, we have an additional term involving the time derivative of the proportional gain. Substituting the expression for the joint acceleration error $\ddot{q}_e$ into this equation, we obtain:

$$\begin{aligned}
\dot{V} &= \dot{q}_e^\intercal\ M(q)\ \ddot{q}_e + \frac{1}{2}\ \dot{q}_e^\intercal\ \dot{M}(q)\ \dot{q}_e + \ q_e^\intercal\ K_p(t)\ \dot{q}_e + \ \frac{1}{2}\ q_e^\intercal\ \dot{K}_p(t)\ q_e \\
&= \dot{q}_e^\intercal\ M(q)\ M^{-1}(q)\ (-K_p(t)\ q_e - K_d(t)\ \dot{q}_e - C(q,\dot{q})\ \dot{q}_e) + \frac{1}{2}\ \dot{q}_e^\intercal\ \dot{M}(q)\ \dot{q}_e + \ q_e^\intercal\ K_p(t)\ \dot{q}_e + \ \frac{1}{2}\ q_e^\intercal\ \dot{K}_p(t)\ q_e \\
&= \dot{q}_e^\intercal\ (-K_p(t)\ q_e - K_d(t)\ \dot{q}_e - C(q,\dot{q})\ \dot{q}_e) + \frac{1}{2}\ \dot{q}_e^\intercal\ \dot{M}(q)\ \dot{q}_e + \ q_e^\intercal\ K_p(t)\ \dot{q}_e + \ \frac{1}{2}\ q_e^\intercal\ \dot{K}_p(t)\ q_e \\
&= -\dot{q}_e^\intercal K_d(t)\ \dot{q}_e + \frac{1}{2}\ \dot{q}_e^\intercal\ \dot{M}(q)\ \dot{q}_e - \dot{q}_e^\intercal C(q,\dot{q})\ \dot{q}_e - \dot{q}_e^\intercal K_p(t)\ q_e + \ q_e^\intercal\ K_p(t)\ \dot{q}_e + \ \frac{1}{2}\ q_e^\intercal\ \dot{K}_p(t)\ q_e \\
&= -\dot{q}_e^\intercal K_d(t)\ \dot{q}_e + \frac{1}{2}\ \dot{q}_e^\intercal\ \dot{M}(q)\ \dot{q}_e - \dot{q}_e^\intercal C(q,\dot{q})\ \dot{q}_e + \ \frac{1}{2}\ q_e^\intercal\ \dot{K}_p(t)\ q_e
\end{aligned} \tag{4.9}$$

Just like we did previously, we can use the properties of skew-symmetry of the $\dot{M}(q) - 2C(q,\dot{q})$ matrix (equations 2.10 2.11) to simplify equation 4.9 as follows:

$$\dot{V} = -\dot{q}_e^\intercal K_d(t)\ \dot{q}_e + \frac{1}{2}\ q_e^\intercal\ \dot{K}_p(t)\ q_e \tag{4.10}$$

To establish stability using the Lyapunov Stability Theorem, we need the $\dot{V}$ function to be negative definite. For this to be true in our case, we can write an inequality as follows:

$$\begin{gathered}
\dot{V} = -\dot{q}_e^\intercal K_d(t)\ \dot{q}_e + \frac{1}{2}\ q_e^\intercal\ \dot{K}_p(t)\ q_e < 0 \\
\boxed{\implies \dot{q}_e^\intercal K_d(t)\ \dot{q}_e > \frac{1}{2}\ q_e^\intercal\ \dot{K}_p(t)\ q_e}
\end{gathered} \tag{4.11}$$

4.2.2 The Stability Condition

By performing a Lyapunov analysis for the controller we proposed in Chapter 3, we have obtained an equation for the stability of this controller. For the controller to be stable, the condition in equation 4.11 should remain true. We can now analyze the implications of this stability condition.

The first observation we make is that because the matrix $K_d(t)$ is positive definite, this means that:

$$\dot{q}_e^\intercal K_d(t)\, \dot{q}_e \geq 0 \quad \forall \quad \dot{q}_e \in \mathbb{R}^n \tag{4.12}$$

This implies that the left hand term in equation 4.11 is always positive (or zero). It may then be ascertained that there is no limit on the rate of change of the proportional gain when $\dot{K}_p(t) < 0$ - the proportional gain may decrease as fast as is desired. The condition imposes a limit or an upper bound on $\dot{K}_p(t)$ - the proportional gain may not rise arbitrarily fast. It is noted that for this analysis of the stability condition to hold, we assume that all of the individual proportional gains (all of the values on the diagonal of the proportional gain matrix $K_p(t)$) rise and fall at the same time - there is no case where one of the proportional gain values is increasing and the rest are decreasing, or vice versa. This is true in our case, since all of the gains depend on the dissipative function 'r'. However, for a controller where this is not the case, this interpretation of the obtained stability condition may no longer be valid.

Let us now write down how this rate of change is defined:

$$\dot{K}_p(t) = \frac{K_{p_c} - K_{p_p}}{dt} \tag{4.13}$$

Here, dt denotes the sample time of the controller, the $K_{p_p} = K_p(t - dt)$ term denotes the value of the proportional gain at the previous time step, and the $K_{p_c} = K_p(t)$ term denotes the value of the proportional gain at the current time step. Overall, the $\dot{K}_p(t)$ term denotes the difference between the current proportional gain and the proportional gain at the previous time step, divided by the sample time.

4.2.3 Algorithm for $K_p(t)$

In this section, we discuss an algorithm to ensure that our proposed controller does not violate the stability condition we obtained in equation 4.11. The objective is to check and ensure that the stability condition is always fulfilled, and to modify the proportional gain if that is not the case.

The algorithm we propose is as follows: we first calculate K_{d_c} and K_{p_c} (the current calculated values of gains) according to the controller algorithm equations detailed in Chapter 3 (equations 3.9, 3.15). Now, the $\dot{K}_p(t)$ value may be evaluated as follows:

$$\dot{K}_p(t) = \frac{K_{p_c} - K_{p_p}}{dt}$$

We can then define a new quantity, s as follows:

$$a = \dot{q}_e^\intercal K_{d_c}\ \dot{q}_e \tag{4.14}$$

$$b = \frac{1}{2}\ q_e^\intercal\ \dot{K}_p(t)\ q_e \tag{4.15}$$

$$s = \frac{a}{b} \tag{4.16}$$

If $s > 1$, that means that the stability condition is indeed fulfilled, and we do not need to modify the proportional gain. However, if $s \leq 1$, this means that the stability condition is not being fulfilled - the increase in the proportional gain K_{p_c} is too high. In this case, we propose modifying K_{p_c} according to the following formula:

$$K'_{p_c} = \eta(sK_{p_c} + (1-s)\,K_{p_p}) \tag{4.17}$$

where $0 < \eta \leq 1$ is the so called 'step size' for the controller. This η term determines how close or far away from the upper bound of the $\dot{K}_p(t)$ we operate. If η is close to 1, we operate close to this upper bound, and vice versa. This algorithm is also summarized in Algorithm 1.

The incorporation of this stability condition into the controller equations completes the formulation of our proposed controller in a manner that ensures stability as well. All of the controller equations and the stability condition have been summarized in Appendix A.

Algorithm 1 An Algorithm to Enforce the Obtained Stability Equation 4.11

Input: $K_{d_c}, K_{p_c}, K_{p_p}, \eta$

Output: K'_{p_c}

$\dot{K}_p \leftarrow \frac{K_{p_c} - K_{p_p}}{dt}$

$a \leftarrow \dot{q}_e^\intercal K_{d_c}\ \dot{q}_e$

$b \leftarrow \frac{1}{2}\ q_e^\intercal\ \dot{K}_p(t)\ q_e$

$s \leftarrow \frac{a}{b}$

if $s > 1$ **then**

    $K'_{p_c} \leftarrow K_{p_c}$

else if $s \leq 1$ **then**

    $K'_{p_c} \leftarrow \eta(sK_{p_c} + (1-s)\,K_{p_p})$

end if

Chapter 5

Testing the Controller - Coding and Implementation

This chapter details the implementation and coding of the controller. We discuss the particulars of how the controller was programmed and some snippets related to the code are discussed in detail. We then elaborate on the practical implementation of the controller, and how testing was done. The results and their analysis are discussed in Chapter 6.

5.1 The Robot Operating System (ROS)

Since the controller was to be implemented for torque-controlled robots, we wanted to have a generic framework for the controller, so that it may be implemented on various different robots. Many collaborative robots in research ([52], [54], [55]) use the Robot Operating System (ROS) [56] as the underlying architecture for robots - ROS not only provides a set of software tools and libraries to communicate with low level hardware such as actuators and sensors, but is also used to communicate between different robot peripherals to implement sensing, controls, and motion planning algorithms. ROS is generally considered the standard for programming robots in both industry and academia [57] [58]. ROS gives robots an extremely convenient way of communicating information within peripherals, and provides an underlying architecture for implementing controllers as well via the 'ros_control' module [59] (detailed in the next section). ROS works with both Python and C++, though we work with C++ in our case since it runs much faster on hardware.

5.1.1 The ros_control module

The 'ros_control' package [59] includes different modules including controller interfaces, controller managers, transmissions and hardware interfaces. All of these modules help users write code for controllers that can be implemented on any robot running ROS. ros_control provides various ways of implementing control strategies, such as position and torque control. Since our controller is a torque control law, we can use the 'EffortJointInterface' class, which allows torque commands to be computed and sent to the robot's joints. Once the torque commands have been sent, ROS manages the specific low-level details (how the torques are implemented within actuators, how sensor feedback is processed and returned to us). This means that we simply need to write the controller logic, instead of having to deal with low-level hardware directly. We hence chose to implement our proposed controller in code using ros_control, to maximize portability and versatility across robots. The ros_control library has a class called 'pid_controller', which can also be used to implement PD controllers with the integral gain always being equal to zero. We can compute the adaptive gains of the controller on each iteration of the controller update loop, and set these adaptive PD gains for the controller using this 'pid_controller' class.

While we will only be discussing the major parts of the code dealing with the algorithmic implementation, the entire code can be found in the corresponding GitHub repository [60].

5.1.2 The Rigid Body Dynamics Library (RBDL)

The Rigid Body Dynamics Library (RBDL) is a very widely used library for modeling and computation of robot kinematics and dynamics. We can often generate an initial overall model of the robot using a URDF (Unified Robot Description Format) file that contains information about the robot's links and joints [61]. A URDF file is an XML file that contains details about each link (such as the geometry and inertia) and each joint (such as the joint type, parent and child links). We can then use various algorithms to compute any required quantities. The two major quantities we are concerned about are the mass (or inertia) matrix for our robot, and the gravity vector (for gravity compensation). In the code, we make use of RBDL's implementation of the Composite Rigid Body Algorithm (CRBA). Similarly for the Gravity Vector, we utilize RBDL's Inverse Dynamics algorithm. RBDL is also a fast library since it is implemented in C++.

5.2 Coding the Controller

In this section, we will be discussing the code related to the controller in three major parts. The first two subsections discuss the 'set_adaptive_pid_gains' function, with the first subsection discussing the adaptive PD gains being computed, and the second subsection discussing the incorporation of the stability condition. The third subsection discusses the 'updateExtras' function as part of the update loop. Note that there was a lot more coding that needed to be done to implement this controller, and we only discuss the algorithmic implementations specific to our proposed controller in this section. The pieces of code referenced in the following sections are all included in Appendix B. For the entire code, please feel free to take a look at our GitHub repository [60].

5.2.1 Evaluation of Adaptive PD Gains

(The code referenced in this subsection is included in Appendix B.1)

The code described in this section goes step by step in computing the adaptive PD gains according to our proposed equations (3.9 - 3.15). The function 'set_adaptive_pid_gains' takes the robot's energy (evaluated according to equation 2.26), the position error, the joint velocities, and the controller parameters (α, β, $K_{d_{max}}$, as detailed in Chapter 3) as the input. We can then proceed to compute our adaptive gains.

The function 'r' as defined in equation (3.8) is computed in code as seen on line 17 (once again, the code may be found in Appendix B.1). We then proceed to calculate the adaptive proportional gain, $K_{p\_adaptive}$. We calculate p_1 according to equation (3.11) on line 22, and then p_2 according to equation (3.12) on line 23. Finally, element-wise multiplication is performed to obtain $K_{p\_adaptive}$ on line 24. A very similar process is followed for obtaining the adaptive derivative gains $K_{d\_adaptive}$. We calculate d_1 according to equation (3.17) on line 28, and then d_2 according to equation (3.18) on line 29. Finally, element-wise multiplication is performed to obtain $K_{d\_adaptive}$ on line 30.

Remember that to limit the user-added energy, we also need to add the d_3 term to the adaptive derivative gain $K_{d\_adaptive}$. To do this, we compute the r_T term as detailed in equation (3.23). This is done on line 31 (referred to in the code as r_ke). Then, we can proceed to add the d_3 term from equation (3.24) to the adaptive derivative gains to obtain the final form for $K_{d\_adaptive}$. This is done on line 32, where the terms on the right hand side being added to the adaptive derivative gains are the d_3 term.

5.2.2 The Stability Condition

(The code referenced in this subsection is included in Appendix B.2)

In this section, we discuss the incorporation of the stability condition within the algorithm. We are still continuing within the 'set_adaptive_pid_gains' function, after determining the adaptive PD gains as discussed in the previous section.

Remember that a Lyapunov analysis of the system (presented in Chapter 4) led us to the stability condition obtained in equation (4.11). We also discussed an algorithm to enforce this stability condition, which is given in equation (4.17). This is what has been implemented in this section of code. On line 3, we calculate the a variable according to equation (4.14), and on line 5, we calculate the b variable according to equation (4.15). Finally, we calculate the variable s by dividing a by b, as shown in equation (4.16). Now, if $b < 0$, that means a decrease in the proportional gain is in order, and the stability condition will not be violated in any case - hence this case does not need to be dealt with explicitly. If $b = 0$, that means that the adaptive proportional gain does not need to change, which is what the first condition from line 12 to 16 asserts. Now, if $b > 0$, that means an increase in the proportional gain. To achieve this, we specify, as an additional tunable parameter, a small-valued variable called 'gain_rise_rate', which the gain is increased by when $b > 0$. This helps us avoid being stuck in the $\dot{K}_p = 0$ zone, while also having control over how fast the proportional gain may rise.

Finally, once we have enforced the stability condition on the adaptive PD gains, we set these gains for the robot. Since we are working with 'ros_control' library, we use the pid_controller function to set these gains. This entire process is done from line 27 to 38.

5.2.3 The Update Loop

(The code referenced in this subsection is included in Appendix B.3)

In this section, we discuss the code for the update loop. The update loop - denoted by the 'updateExtras' function - is the piece of code that runs on a certain frequency, and contains the bulk of the controller logic. We will discuss how the update loop is structured, and how the controller is implemented in real time.

The 'updateExtras' function begins by getting the current robot state (current position, desired position, cur-

rent velocity) from line 5 to 8. The robot energies (according to equation 2.26) are calculated on line 13. We then check if a control variable 'set_gains_adaptively' is true, and if it is, we call the 'set_adaptive_pid_gains' function (that we discussed in the two previous subsections). It is important to have this control variable 'set_gains_adaptively', since it allows us to switch between a non-adaptive simple PD controller, and our proposed adaptive PD controller. After that, we initialize some variables from lines 22 to 33. These variables are all either used within the update loop, or are used to publish controller-related variables to a ROS topic. We then enter a *for* loop that iterates through all of the controlled joints.

On line 45, we call a function named 'enforceJointLimits', to ensure that the reference position is within the joint limits for that joint. From line 47 to line 64, we calculate the error between the current position and the desired position differently depending on whether the joint is prismatic or revolute (for prismatic joints the error is simply the difference between current and reference positions, whereas for revolute joints, the shortest angular distance is calculated). We then use our adaptive PD gains to compute the effort or torque that should be commanded using the 'computeCommand' function of the 'pid_controllers' class. As stated previously, this class with the integral gain always being equal to zero, may be used to implement PD controllers as well. Then we enter a series of if conditions that perform various checks. If that particular joint has been activated (if the 'activate_joints' value is 1 for that particular joint - this is once again a control variable we use so that we can select the joints we want to actuate while the controller is running), we send the computed torque (according to equation 2.16) to the joint. If the 'add_gravity_comp' variable is true, we add the torque due to gravity compensation to our computed torque (this is the value of the gravity vector associated with that particular joint). Finally, on line 81, we check if the variable 'send_gen_torques_to_robot' is true, and if it is, we use the 'setDesiredJointTorque' function to finally send the torques to the robot. The reason for having so many check variables is so that we may be able to control when the generated torques are sent to the robot, as well as being able to control each joint individually. This is an important feature from a safety perspective as well, since we want to able to test the controller with external forces and pushes as well, and having control over activation of individual joints is a good safety feature to have.

5.3 The TALOS Robot - Implementation

This section details the experimental and testing setup we used for the testing of our proposed adaptive PD controller. We discuss the robotic setup used to test the controller and its performance. We then describe the specific tests we performed to validate the expected controller behavior.

5.3.1 The Setup

To test our proposed controller, we implemented our controller on the TALOS robot by PAL Robotics [52]. The TALOS robot is a humanoid biped robot capable of walking. It is a torque-controlled robot with torque sensing feedback for all joints, which is what was needed to implement our controller experimentally. The robot also runs on the Robot Operating System (ROS), making it very easy to work with. The ros_control package is already implemented for the robot, and that made it very easy for us to run our custom ros_controller package on the robot. We got to work on the TALOS robot housed at the University of Waterloo's Robohub. A picture of the TALOS Robot at the University of Waterloo's RoboHub is shown in Figure 5.1. I was lucky enough to have had the chance to spend a couple of weeks at the University of Waterloo, and get to work at the Robohub. We implemented the proposed controller for both the TALOS robot's arms, and tested it for a variety of scenarios. The TALOS Robot's arms both have 7 degrees-of-freedom (DOF), and the robot has the URDF file for the complete model available as well to access the robot's dynamics and kinematics, which is also an important part necessary for the implementation of our proposed controller. The robot also has a friction compensation module that can act between a controller and the robot actuators, acting to minimize the effect of friction while the joints are actuated according to the control law.

Figure 5.1: A picture of the TALOS Robot

5.3.2 The Tests

To test our proposed controller's performance, we first needed some sort of a baseline to compare our proposed method to. For that, we used a simple, non-adaptive Proportional-Derivative (PD) Controller with gravity compensation 2.16. The performance for 'safe' energy levels (low velocities and no external forces) for the robot should be fairly similar across both the non-adaptive PD Controller, and our proposed PD Controller. However, the main goal of our testing was to verify three major things related to the limitation of the total energy of the robot - the three main criteria are listed below:

1. **Testing how well our proposed Potential Energy Limitation works -** Limiting the potential energy of the robot is done by decreasing the proportional and derivative gains of the controller as the total energy of the robot approaches the limit. We want to test if the controller does indeed lead to safe robot behavior by reducing the PD gains when pushed away, and whether the robot's potential energy is indeed limited.

2. **Testing how well our proposed Kinetic Energy Limitation works -** Limiting the kinetic energy of the robot is done by increasing the derivative gain of the controller when the robot's kinetic energy approaches the limit. We want to test that when the robot's kinetic energy goes very high due to an external push by a human, the robot behaves safely by increasing its derivative gains to dampen this external force, and the robot's kinetic energy is indeed limited.

3. **Testing and comparing the tracking performance of a simple PD controller and our proposed controller -** While the performance of both controllers will be very similar when tracking a reference position using a trajectory generator, the comparison of interest for us will be to compare the performances of the two controllers after the robot's arm is pushed away. Is the energy limit followed while tracking a desired position as well? It is important to note that when describing a desired trajectory, we usually refer to the desired positions and velocities over a given timeframe. There may be some trajectories that could potentially violate energy limits. For example, if the desired position for a joint jumps from zero to a high value, would the controller be able to limit the energy of the robot in this case? And how does the tracking response of the two controllers differ after a push?

To test these three different criteria, we devised two major tests for the controllers to undergo. For these tests, the robot would first be made to track a certain reference position. When the robot achieves this reference position and is tracking it continuously, we would perform two main tests:

- **An external force is applied gradually to the robot arm -** the robot arm is pushed slowly - while the reference position is being tracked. We want to test whether our proposed adaptive controller

indeed limits the energy of the robot (which in this case will mainly be reflected in the potential energy). We look out for both criteria 1 and 3 during this test.

- **A high external force is applied very suddenly to the robot arm -** the robot arm is pushed suddenly - while a reference position is being tracked. We want to test whether our proposed adaptive controller indeed limits the energy of the robot in this case (which will mainly be reflected in the kinetic energy). We look out for both criteria 2 and 3 during this test.

We perform these two tests with the robot for both a non-adaptive, simple PD controller, and for our proposed adaptive PD controller. We compare the results for these two controllers to test whether our proposed controller limits the total energy of the robot. We look out for the three main criteria we specified in this section, and compare the results in both simulation and hardware experiments.

Chapter 6

Results and Discussion - The Controller's Performance

In this chapter, we discuss the performance and results obtained with the proposed controller. We split these results into two major sections - simulation, and hardware results. Within each section, we first list the exact controller parameters and variables used for the tests. We then move on to discussing the performance of our proposed controller in the presence of external forces while a reference trajectory is being tracked. We also compare the results of our proposed controller to a standard, non-adaptive PD controller with the same initial gains, to get an idea about the performance difference between the two, as well as our proposed controller's energy-limitation capabilities. We analyze these results in the context of the three main criteria listed in Chapter 5. Finally, we discuss some of the limitations of our current approach.

6.1 Simulation

To test our controller in simulation, we run a simulation of the TALOS Robot in Gazebo [62]. Gazebo is a physics simulator very commonly used to simulate robots and test control, planning, and navigation strategies. We can load different robots into a variety of world environments, and can even simulate their interactions with other objects, as well as external forces. Gazebo also integrates seamlessly with ROS, meaning that the code we develop for our actual robots can be used in combination with Gazebo to form a complete simulation and testing suite.

The TALOS Robot already has instructions for setting up a Gazebo simulation [63]. This prepackaged

simulation setup comes with the TALOS Robot and its dynamic model, and various accompanying libraries - including some essential controllers set up with ros_control. I was able to run the code for the proposed controller for a simulated TALOS robot. A picture of the running Gazebo simulation for the TALOS Robot is shown in figure 6.1.

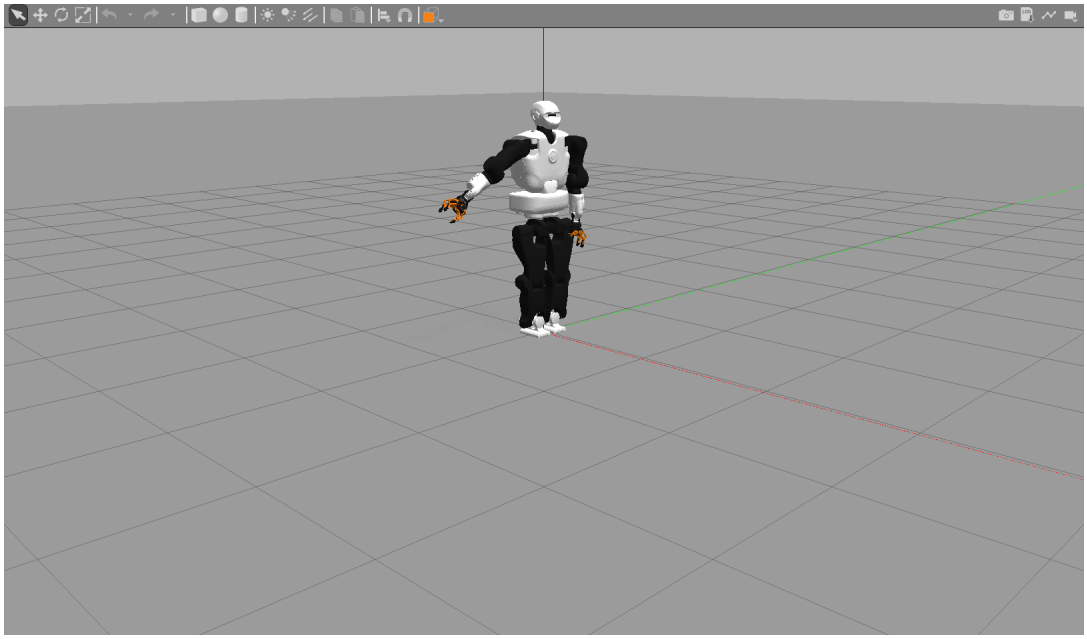

Figure 6.1: The TALOS Robot running in the Gazebo Simulator

6.1.1 Adaptive PD Controller Parameters

Here, we list the parameters for our controller that were implemented for the robot, and were used to evaluate the controller's performance. We start off with the initial proportional and derivative (PD) gains (also referred to as the nominal PD gains) used:

- $K_{po} = [10,\ 10,\ 10,\ 10,\ 10,\ 10,\ 10] \quad \frac{N \cdot m}{rad}$
- $K_{do} = [6,\ 6,\ 6,\ 6,\ 6,\ 6,\ 6] \quad \frac{N \cdot m}{rad}$
- $K_{d_{max}} = [20,\ 20,\ 20,\ 20,\ 20,\ 20,\ 20] \quad \frac{N \cdot m}{rad}$

These are the gains that were tuned with the gravity compensation enabled, and were tuned very simply. We used a heuristic method to choose the gains that would reduce the overshoot and tracking time. While a more rigorous tuning method may have been employed, since that was not the main goal of the experiments, any PD gains that perform reasonably well would be sufficient. For the maximum derivative gain ($K_{d_{max}}$), we choose a value of 20 $\frac{N \cdot m}{rad}$ across all joints. This is the maximum value the derivative gains will take on

when the kinetic energy gets too high, and high kinetic energy values will be dampened by the derivative gains taking on a fraction of these maximum values depending on how high the kinetic energy is. Choosing an appropriate value for $K_{d_{max}}$ is very important, since a high value could lead to very high resistance to external pushes due to high damping, and a low value could lead to the robot offering not enough resistance to external pushes. The exact tuning of this parameter will depend on the particular application scenario.

The next parameters to be defined are the three parameters that determine the behavior of our adaptive controller:

- $\mathcal{L}_{lim} = 0.8 \quad J$
- $\alpha = 10$
- $\beta = 0.6$

The choice for these parameters was done keeping in mind the desired behavior. Since we wanted a smoother transition from the nominal gains to their adaptive values, we choose a relatively lower value of α. The choice for β was made to be about 75% of the total energy limit, so that the gains adapt at a relatively lower energy level. If this value were to be higher (such as 90%), the gains would start adapting at a later stage. Once again, this choice will be completely dependent on the desired behavior of the controller. The graphs in figures 3.1 and 3.2 may be referred to for an idea of how the α and β parameters may be determined.

6.1.2 Experimental Protocol

The right arm of the TALOS Robot is first set to the goal position of $[0, -1, 0, 0, 0, 0, 0]$ radians. Then, as the robot is holding this position, a constant external force is applied using Gazebo's inbuilt functionality to the arm at link 5, in the positive direction of the second joint. The direction and location of this force is illustrated in 6.2. When plotting the results for this interaction, the energy and its components are shown for the entire right arm, whereas the PD gains, the position, and the torque are plotted for just the second joint since that is the joint position most directly changing due to the force. Both the reference torque (τ_r) and the actual torque (τ_0) are shown in the graphs. The blue shaded region in each plot indicates the time period when the external force is applied to the arm.

6.1.3 Traditional PD Controller Performance - The Baseline

We first take a look at the simple PD controller's performance in response to external forces. This simple PD controller has the exact same gains as the nominal gains for the adaptive PD controller:

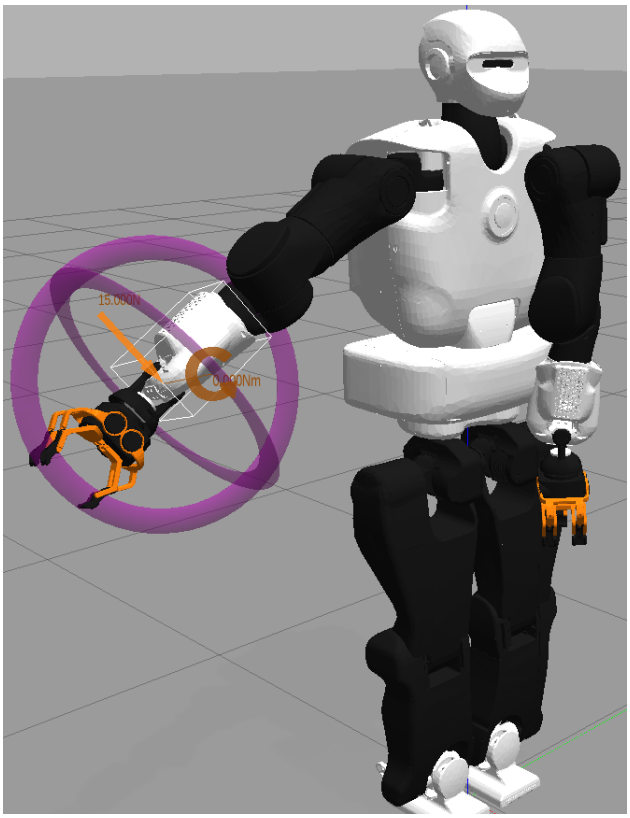

Figure 6.2: The Direction of Applied Force in Simulation. The arrow in orange denotes the force being applied on link 5 of the right arm.

- $K_{po} = [10,\ 10,\ 10,\ 10,\ 10,\ 10,\ 10]$ $\frac{N \cdot m}{rad}$
- $K_{do} = [6,\ 6,\ 6,\ 6,\ 6,\ 6,\ 6]$ $\frac{N \cdot m}{rad}$

15 N External Force

For this test, we apply a force of 15 N as the external force. The plots that show the resulting behavior of this interaction are shown in figure 6.3. Though this simple PD controller does not perform any energy limitation, we have also plotted the target energy limit in this plot.

We can divide the entire interaction into three key phases:

1. **Before Application of the Force:** Initially, the robot arm's second joint is holding its target joint position of -1 radians. Since the error between the desired position and actual position is nearly zero, the potential energy at this point is consequently almost zero. Since the velocity of the joint is also nearly zero at this point, the kinetic energy is consequently almost nearly zero. This phase runs from time 0 to 1 second.

2. **During Application of the Force:** As the force is applied, the joint position starts changing slightly,

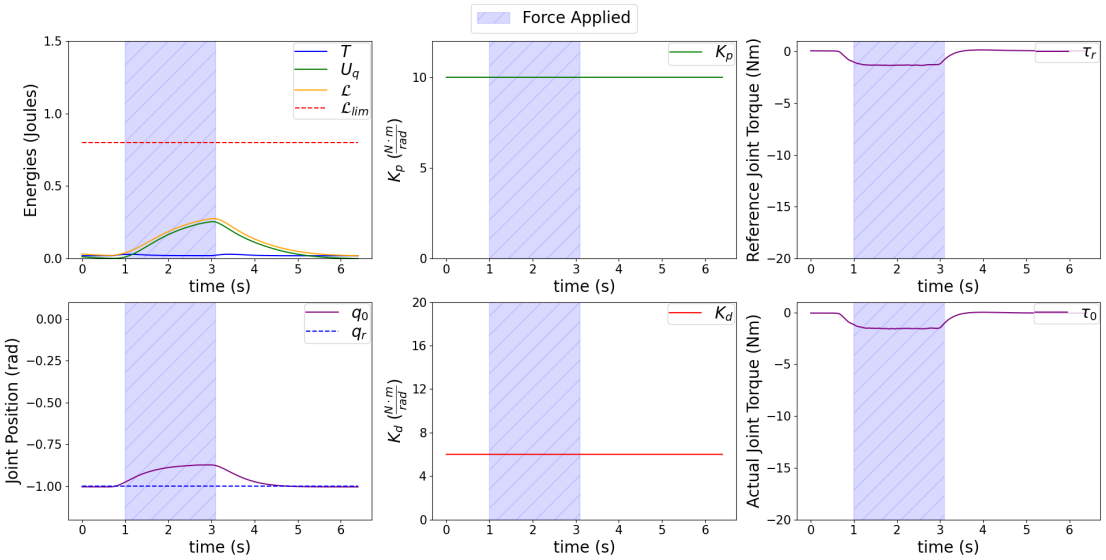

Figure 6.3: Traditional PD Controller's Response to a 15 N Force (Simulation)

with the error reaching a maximum value of 0.127 radians. The energy of the system goes up slightly, mostly due to the potential energy. However, the energy does not go close to the defined energy limit of the system, since the arm proves to be resistant to change. The maximum value of energy reached is 0.29 Joules.

3. **After Application of the Force:** As the applied external force is removed, the robot's energy starts decreasing as the robot's corrective behavior starts. The joint takes 1.06 seconds after the release of the force to settle back within 2% of the desired joint position.

Since this test is part of the controller performance we are going to use as the baseline performance, there are some general observations we can make about this simple PD controller's performance when subjected to an external force:

- For the simple PD controller, a force of 15 N only moved the robot's arm a small amount. This is due to the arm's stiffness being high as compared to the external applied force. This is not necessarily a desired behavior from a pHRI safety perspective, since we would like the arm to behave softer or less stiff when pushed with small forces. If the arm collides with a human while tracking a reference trajectory, stiff behavior by the arm might not be desirable.

- The 15 N force is not high enough to test the controller's energy-limitation, since this force is not able to move the arm very much. This leads to low potential and kinetic energies that are not close to the limit. We would like to see the behavior of the controller when it is subjected to a higher force!

- Not much can be said about this corrective behavior on its own, since the error induced due to the external force is very small, and we would expect the settling time to be quick.

In summary, the controller acts fairly stiffly with an external force of 15 N, not moving much, and not gaining much energy. While the stiffness is not an ideal behavior in the context of pHRI, the controller will need to be tested with higher forces to make better conclusions about the controller performance. In the next subsection, we will evaluate the traditional PD controller's performance in the presence of larger external forces.

6.1.4 30 N External Force

For this test, we apply a force of 30 N as the external force. The plots that show the resulting behavior of this interaction are shown in Figure 6.4.

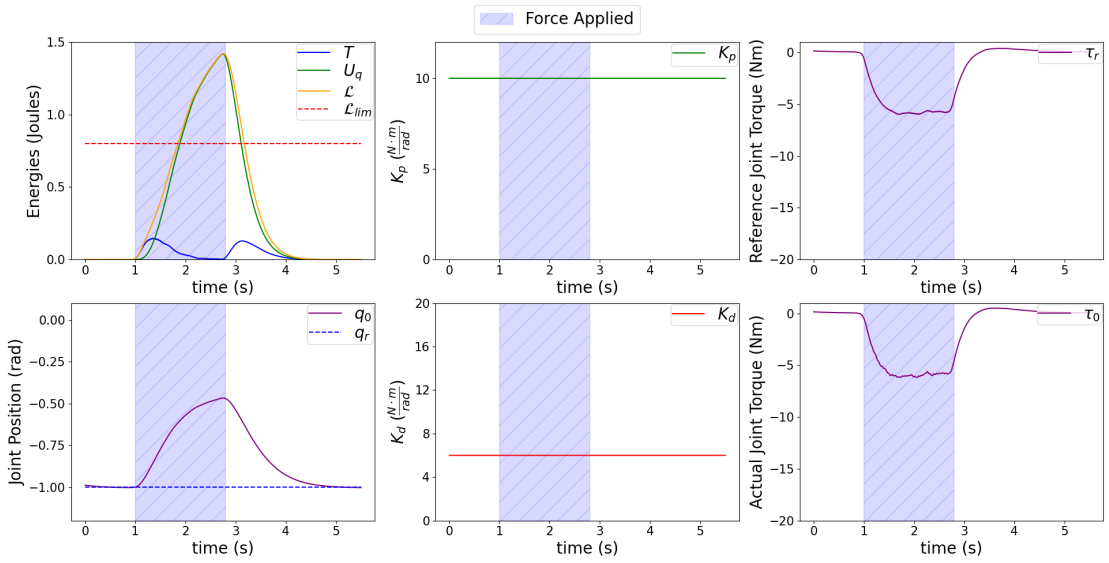

Figure 6.4: Traditional PD Controller's Response to a 30 N Force (Simulation)

To analyze this plot, we can divide the interaction into three key phases:

1. **Before Application of the Force:** Taking a look at the plot, initially, the robot arm is tracking the target joint position of $[0, -1, 0, 0, 0, 0, 0]$ radians. Since the error between the desired position and actual position is nearly zero, the potential energy at this point is consequently almost zero. Since the velocity of the joint is also nearly zero at this point, the kinetic energy is consequently almost nearly zero. This phase runs from time 0 to 1 second.

2. **During Application of the Force:** At the 1 second mark, the external force is applied (denoted

by the shaded area). This is reflected in both a change in the position of the joint, and the change in energy levels. The robot is not moving too fast - hence the relatively low kinetic energy. However, the error increases to 0.53 radians. Since the proportional gain K_p is high relative to the error, the potential energy of the robot rises quite quickly as the error increases - rising to a maximum value of 1.417 Joules, which is higher than the energy limit. The total energy of the robot crosses the energy limit at the 1.85 second mark, giving rise to high energy, and consequently, high reaction forces.

3. **After Application of the Force:** As the external force is removed, the controller corrects the error, with the joint taking 1.67 seconds after the release of the force to settle back within 2% of the desired joint position.

Once again, there are some general observations we can make about this simple PD controller's performance when subjected to an external force:

- The reason an external force of 30 N was chosen for this test was because forces of lower magnitude did not cause the robot arm to move much. When we apply smaller forces, the robot moves a bit, but the force is not large enough to overcome the torque the proportional gain causes, and the robot arm does not end up moving too much. While this is great for tracking accuracy and 'robustness' to external disturbances, it does not necessarily represent safe behavior in the context of human-robot interaction. In case of a collision or a push, the robot will not move much and display stiffness, where instead in the case of pHRI, a softer response might be desired.

- Such a controller cannot guarantee energy-limitation under any given limit. This of course is the goal of our proposed adaptive PD controller, and in the plots for our adaptive PD controller, this is what we will be looking for as one of the key differentiators.

- The corrective behavior of the controller can be fast since no limit exists on the robot's energy. This means that the controller's response to an external force can induce high joint velocities. Again, while this is a good thing from a purely 'tracking' perspective, this might not be the desired behavior in terms of human-robot interaction - we might want this correction to be more deliberate and slower for safety, since we want to avoid high forces. This is again one of the differentiators we will be keeping an eye out for.

To summarize, the PD controller performs well in terms of the pure tracking performance. The controller, depending on the gains, can be resistant to change from external forces, and could be quick to perform corrective actions if an error accumulates. However, from a pHRI perspective, these behaviors exhibited by the controller are not ideal. High resistance to change is not always desirable from a pHRI perspective, and

according to the ISO/TS 15066 standards (2016), energy limitation is an effective way of ensuring safety [27] [3] [23]. Hence in the next subsection, we will evaluate the performance of our proposed adaptive PD controller according to these criteria - resistance to change, energy limitation, and tracking performance of the controller.

6.1.5 Proposed Adaptive PD Controller - Simulation Results

In this section, we take a look at our proposed adaptive PD controller's performance in response to external forces. The parameters for the proposed adaptive PD controller have been listed in Section 6.1.1 (Remember that the nominal gains for the adaptive PD controller are the same as the gains for the simple PD controller as described in Section 6.1.1).

15 N External Force

For this test, we apply a force of 15 N as the external force, in the manner described in section 6.1.2.

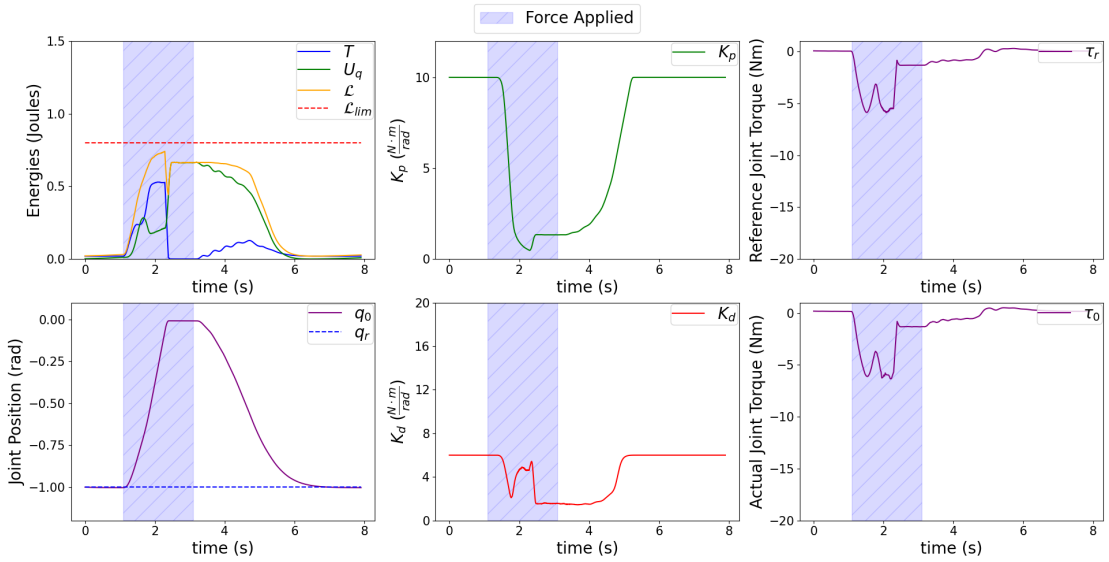

Figure 6.5: The Proposed Adaptive PD Controller's Response to a 15 N Force (Simulation)

As before, we can divide the entire interaction into three key phases:

1. **Before Application of the Force:** Initially, the robot arm is holding its target joint position of -1 radians. Since the error between the desired position and actual position is nearly zero, the potential energy at this point is consequently almost zero. Since the velocity of the joint is also nearly zero at this point, the kinetic energy is consequently almost nearly zero. This phase runs from time 0 to 1 second.

2. **During Application of the Force:** As the force is applied, the joint position starts changing, just as it does for the traditional PD controller (shown in Figure 6.3). However, immediately we see our first change in the controller's behavior - as the energy rises, the gains start adaptively decreasing. Both the proportional and derivative gains decrease as the energy increases. As a consequence, the robot stiffness decreases - meaning the robot becomes more compliant or 'soft'. Hence why we see the kinetic energy of the robot rise to a higher level due to the energy injected into the system by the external force. However, as opposed to the traditional PD controller, the robot does not exceed the energy limit for the arm due to the way the adaptive gains change. The decrease in the proportional gain means that the total energy (in particular the potential energy) of the robot does not exceed the given energy limit. Even though the error between the desired and actual position goes to 1 radian (as opposed to 0.127 radians and 0.53 radians in the traditional PD controller's test cases), the energy of the robot does not exceed the limit.

3. **After Application of the Force:** As the applied external force is removed, the robot's energy starts decreasing. This leads to the gains slowly increasing, amounting to a gradual increase in the kinetic energy as the robot's corrective behavior starts, and leads to a slow decrease in the potential energy as the joint error decreases. The arm finally settles back to the desired position, and the joint position curve gradually increases as opposed to the sharp increase from the traditional controller. The joint takes 3.14 seconds after the release of the force to settle back within 2% of the desired joint position.

When comparing this performance to the baseline performance of the simple PD controller, we note several key differences:

- For the simple PD controller, a force of 30 N was needed to move the robot's arm a substantial amount, while the 15 N force was not able to move the arm much (the maximum error was 0.127 radians). In this case however, the 15 N force is able to move the robotic arm compliantly. This is evident by the fact that even though the force in this case is half the magnitude of the 30 N case for the traditional controller, we are able to move the joint position all the way from -1 radians to 0 radians, whereas in the baseline case, the joint only moved from the -1 radians to about -0.47 radians. The arm is much 'softer' or compliant, and such behavior is considered safer in the context of pHRI. The robot is more easily pushed away when a smaller force is applied to the robot. As discussed in Chapter 2, compliance can lead to safe behaviors within the context of pHRI, particularly in clamping scenarios.

- The energy-limitation feature works well in this case. The decrease in gain as the energy approaches the limit leads to a decrease in the potential energy - this means that even though in the baseline case

the robot's energy (the potential energy in particular) exceeds the limit, in this case the adaptive PD gains help the robot keep its energy under the limit. This is one of the major goals of our proposed controller!

- The corrective behavior of the controller is much more gradual. Since we are at the limit of the energy when the external force is released, the gains start increasing gradually, and the robot returns to its desired joint position in a much safer manner since a more gradual movement means lower energies and torques for the arm. This is evident in the fact that returning to the goal position takes 3.14 seconds in this case as compared to 1.06 seconds (15 N external force) and 1.67 seconds (30 N external force) values we obtained for the traditional PD controller.

In summary, the adaptive PD controller is able to successfully limit the robot's energy under the given limit, apart from a slight violation that we at this point are attributing to the simulated force being discontinuous at times (the further tests performed in simulation and on hardware generally support this). The gains decrease adaptively and lead to not only energy-limitation (in this specific case, it is mostly the potential energy being limited), but some desirable behavior in the context of pHRI. The robot acts 'softer' or more compliant in the presence of external forces, and is much more gradual in its corrective behavior. Even though the tracking performance may have been traded off since the adaptive controller takes longer to reach the desired joint position, the energy limitation indicates a betterment in the safety performance of the robot. In the next part, we will evaluate the adaptive PD controller's performance in the presence of a larger external forces.

30 N External Force

For this test, we apply a force of 30 N as the external force. Remember that the simple PD controller was tested for a force of 30 N as well, and we want to draw a comparison with this case directly. The graph that shows this interaction is shown in figure 6.6.

Once again, we can divide the entire interaction into three key phases:

1. **Before Application of the Force:** Initially, the robot arm is holding its target joint position of -1 radians. Since the error between the desired position and actual position is nearly zero, the potential energy at this point is consequently almost zero. Since the velocity of the joint is also nearly zero at this point, the kinetic energy is consequently almost nearly zero. This phase runs from time 0 to about 1 second.

2. **During Application of the Force:** As the force is applied, the joint position starts changing, just as it does for the simple PD controller. However, this time we observe behavior that is specific to large

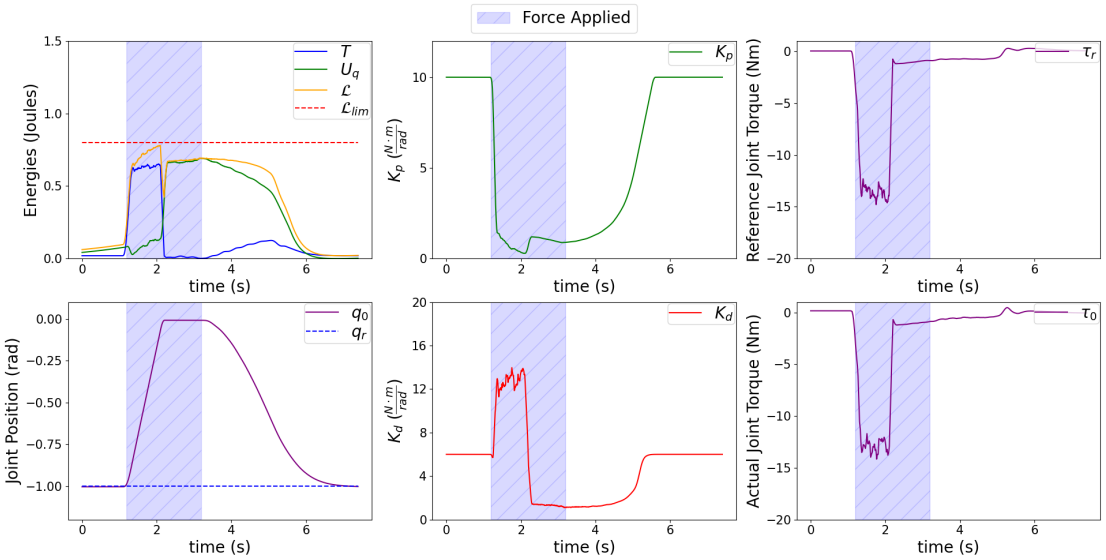

Figure 6.6: The Proposed Adaptive PD Controller's Response to a 30 N Force (Simulation)

forces - there is a sudden increase in the kinetic energy of the robot since it is moving at a high speed. Since the total energy of the robot is increasing, the proportional gains start decreasing. However, since it is the kinetic energy that is rapidly increasing to cause an increase in the total energy, the derivative gains start increasing as well, to increase the damping of the arm. We see that this increase in damping leads to the kinetic energy of the arm being arrested to a constant. Notice that the source of opposition has changed - instead of the opposing force being present due to potential energy increase (or proportional gain), the opposing force is now a damping force (due to the derivative gain) of the robot. This showcases the limitation of high kinetic energy by the controller.

3. **After Application of the Force:** As the applied external force is removed, the robot's energy starts decreasing - in particular, the kinetic energy now starts decreasing and the potential energy starts increasing. The reduction in kinetic energy is due to the external force being removed and the arm slowing down - this allows the potential energy to increase by increasing the proportional gains, since there is freed up energy budget to be utilized now. The robot's corrective behavior starts, and leads to a slow decrease in the potential energy as the joint error decreases. The arm finally settles back to the desired position, and this action takes 3.31 seconds from the time the applied force is released.

As compared to the baseline performance of the simple PD controller, we note several key differences:

- The proposed adaptive PD controller succeeds in limiting the total energy of the arm under the given budget, when subjected to a 30 N force. In this case, it is the kinetic energy with a larger contribution to the robot's total energy. Remember that while we do want our robot to be 'soft' or compliant, we

also do not want it producing large reactionary forces to external pushes (as would be the case for a simple PD controller), nor do we want the robot to move in a flailing manner, as the controller might be unable to stabilize the arm in that case. Hence, resisting high external forces via damping rather than 'elastic' behavior (based on higher proportional gains) might be more desirable when the aim is to reduce both the kinetic and potential energy of the system.

- The energy-limitation is done by reducing both the potential energy (by reducing the proportional gain) and reducing the kinetic energy (increasing the derivative gain to increase damping). The controller hence is capable of dealing with both forms of energy increase simultaneously.
- The utilization of energy budgets is done by balancing the kinetic and potential energy. As a rise in the kinetic energy of the robot occurs, while the controller attempts to limit the increase in joint velocity by increasing the damping (via the derivative gain), the proportional gains decrease simultaneously to decrease potential energy and stay below the energy limit. Similarly when the external force is removed, the kinetic energy of the robot decreases. This leads to an amount within the energy budget freeing up. This free budget is taken up by the potential energy by increasing the potential gains, so that a faster tracking response may be obtained while still staying within the energy budget.

In summary, the adaptive PD controller succeeds in limiting the robot's kinetic energy as well. The gains adapt rapidly to the situation and lead to energy-limitation and desirable behavior in the context of pHRI. The robot acts 'softer' or more compliant in the presence of smaller external forces, and dampens larger external forces to avoid unstable behavior by the arm. Even though the tracking performance may have been traded off (3.31 seconds of settling time as compared to the 1.67 seconds for the traditional PD controller case), there is once again evidence of better safety performance of the robot.

6.2 Hardware Results

In this section, we present the results of the tests we performed on hardware for the proposed adaptive PD controller. As explained in Chapter 5, the proposed controller was implemented using the ROS Control package, and was tested practically on the TALOS robot by PAL Robotics. The particular TALOS robot that was used for our testing is present at the University of Waterloo RoboHub. A picture of the setup is shown in Figure 6.7.

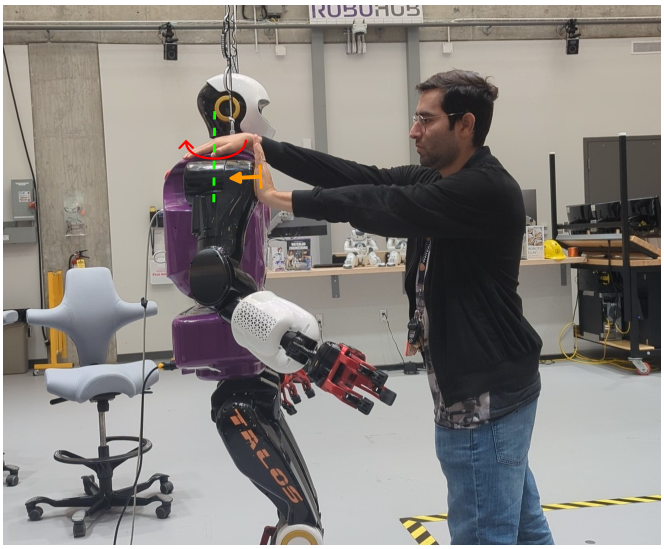

Figure 6.7: The TALOS Robot Experimental Setup. The orange arrow denotes the direction of the applied force. The green dotted line denotes the axis of rotation for the joint, whereas the red arrow shows the resulting joint movement.

6.2.1 Adaptive PD Controller Parameters

Here, we list the parameters for our controller that were implemented for the robot, and were used to evaluate the controller's performance. We start off with the initial proportional and derivative (PD) gains used:

- $K_{po} = [5,\ 5,\ 5,\ 5,\ 5,\ 5,\ 5] \quad \frac{N \cdot m}{rad}$
- $K_{do} = [0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0] \quad \frac{N \cdot m}{rad}$
- $K_{d_{max}} = [0.5,\ 0.5,\ 0.5,\ 0.5,\ 0.5,\ 0.5,\ 0.5] \quad \frac{N \cdot m}{rad}$

The reason for starting with such small gains is mainly due to safety reasons. While we were able to test in simulation with larger gains, TALOS is actually quite a powerful humanoid robot. Since we wanted to test the controller's performance in the presence of a variety of external forces as well, we decided to work with smaller gains for our hardware testing phase. Because of the natural damping of the robot's joints due to any unmodeled friction, we choose to start with nominal derivative gains of zero. Note that while $K_{do} = 0$, for our adaptive PD controller, the derivative gain should increase if the kinetic energy approaches the limit (a case that we explicitly test as well).

The next parameters to be defined are the three parameters that determine the behavior of our adaptive controller:

- $\mathcal{L}_{lim} = 0.15 \quad J$

- $\alpha = 3$
- $\beta = 0.13$

As in the case of the simulation, the choice for these parameters was made keeping in mind the desired behavior. Since we wanted a smoother transition from the nominal gains to their adaptive values, we chose a relatively lower value of α. The choice for β was made to be about 86% of the total energy limit. This choice for these values was completely dependent on the desired behavior of the controller. Once again, graphs in figures 3.1 and 3.2 can be referred to for an idea of how the parameters α and β may be determined.

6.2.2 Experimental Protocol

The right arm of the TALOS robot is first set the goal position of $[-0.25, 0, 0, 0, 0, 0, 0]$ radians. The configuration chosen is different from the one used in simulation to present results for different joints. Then, as the robot is holding this position, an external force is applied to the arm by a person at link 1, in the negative direction of the first joint, as shown in Figure 6.7. Since the forces are being applied by a person, consistent experiments are much more difficult to perform. However, the focus of these experiments is on ensuring that the controller behaves as expected - in a safe and stable manner. When plotting the results for the interactions, the energy and its components are shown for the entire right arm, whereas the PD gains, the position, and the torque are plotted for just the second joint since that is the joint position most directly changing due to the force. Both the reference torque (τ_r) and the actual torque (τ_0) are shown in the graphs. The reference torque is the desired torque obtained by the controller logic. This desired torque is then sent to a lower-level controller which controls the motors to generate the corresponding motor torque. The lower-level controller was developed and implemented by PAL Robotics, and the difference between the desired and actual torques will be due to the specific controller implementation and the motor dynamics/friction. The blue shaded region in each plot indicates the time period when the external force is applied to the arm.

6.2.3 Simple PD Controller Performance - The Baseline

We first take a look at the simple PD controller's performance in response to external forces to establish a baseline performance. This simple PD controller has the same gains as the nominal gains for the adaptive PD controller:

- $K_{po} = [5,\ 5,\ 5,\ 5,\ 5,\ 5,\ 5] \quad \frac{N \cdot m}{rad}$
- $K_{do} = [0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0] \quad \frac{N \cdot m}{rad}$

The graph that shows this interaction is shown in figure 6.8.

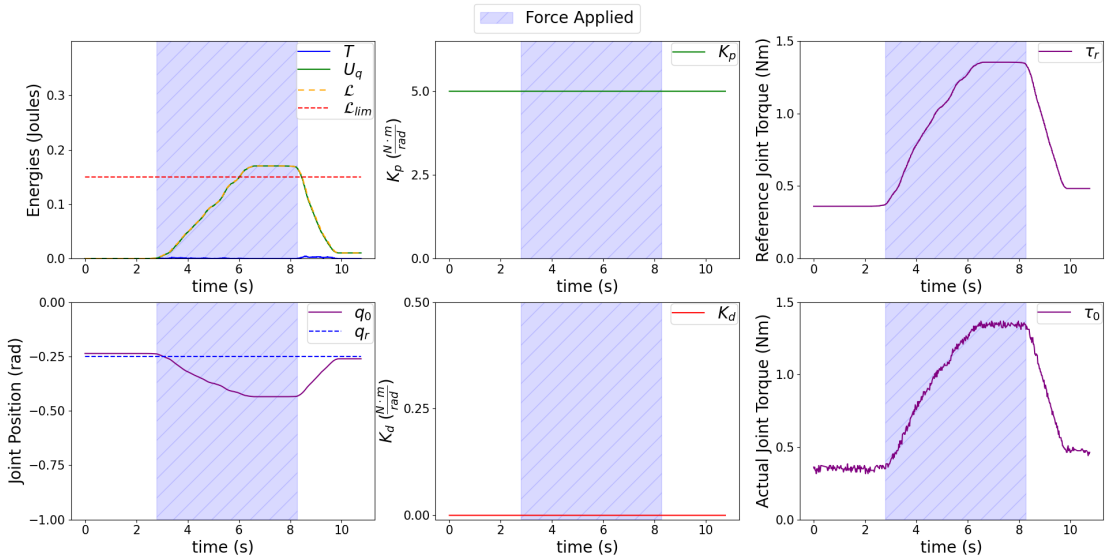

Figure 6.8: A Simple PD Controller's Response to an External Force - Hardware Result

Though this simple PD controller does not perform any energy limitation, we have also plotted the target energy limit in this plot. To analyze this plot, we can divide the interaction into three key phases:

1. **Before Application of the Force:** Taking a look at the plot, initially, the robot arm is holding its target joint position of -0.25 radians. Since the error between the desired position and actual position is nearly zero, the potential energy at this point is consequently almost zero. Since the velocity of the joint is also nearly zero at this point, the kinetic energy is consequently almost nearly zero. This phase runs from time 0 to about 2.8 seconds.

2. **During Application of the Force:** At about 2.8 seconds, the external force is applied (denoted by the shaded area in Figure 6.8). This is reflected in both a change in the position of the joint, and a change in energy levels. The joint position error is increasing, and the potential energy of the robot rises as the error increases. The total energy of the robot crosses the energy limit at 6.02 seconds, giving rise to high energy and consequently a peak torque of 1.37 Nm.

3. **After Application of the Force:** As the external force is removed, the controller corrects the error, and the joint moves back to the goal position and settles back. The joint position settles back in 1.52 seconds to within 5% of the desired position.

Since this is the controller performance we are going to use as the baseline performance, there are some general observations we can make about this simple PD controller's performance when subjected to external

force:

- Note that the joint position reaches a maximum value of -0.43 radians, potentially indicating a lack of 'softness' for the simple PD controller (this will become clearer when compared with the adaptive PD controller). A deflection of 0.18 radians causes the robot's total energy to go over the limit of 0.15 Joules (primarily due to the potential energy).
- Such a controller does not do any active energy-limitation. This of course is the goal of our proposed adaptive PD controller, and in the plots for our adaptive PD controller, this is what we will be looking for as one of the key differentiators.

To summarize, the simple PD controller achieves joint position tracking, as was the case in the simulation as well. The controller seems to be resistant to change from external forces, though this will have to be compared to the adaptive controller case. However, from a pHRI perspective, once again these behaviors exhibited by the controller are not ideal. As discussed in Chapter 2, high resistance to change is not necessarily desirable from a pHRI perspective, and we are considering energy-limitation as a way of ensuring safety. In the next subsection, we will evaluate the performance of our proposed adaptive PD controller and compare it with this case.

6.2.4 Proposed Adaptive PD Controller Performance - The Obtained Hardware Results

In this section, we take a look at our proposed adaptive PD controller's performance in response to external forces. The parameters for the proposed adaptive PD controller have been listed already in Section 6.2.1 (Remember that the nominal gains for the adaptive PD controller are the same as the gains for the simple PD controller).

A Small External Force

For this experiment, a gradual force is applied to the robot's arm at link 1, as shown in figure 6.7. The reason for applying the force gradually is so that the arm does not move fast, and most of the energy injected to the system is in the form of potential energy. This specifically allows us to validate the controller's behavior in terms of the potential energy limitation. The graph that shows this interaction is shown in figure 6.9.

Once again to analyze this plot, we can divide the interaction into three key phases:

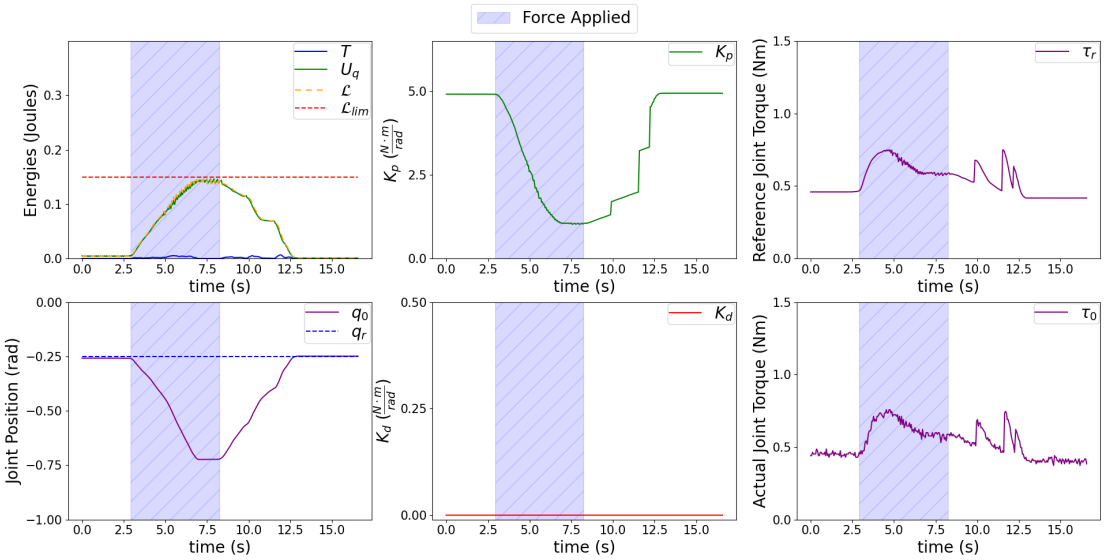

Figure 6.9: The Proposed Adaptive PD Controller's Response to a Small External Force - Hardware Result

1. **Before Application of the Force:** Taking a look at the plot, initially, the robot arm is holding its target joint position of -0.25 radians. Since the error between the desired position and actual position is nearly zero, the potential energy at this point is consequently almost zero. Since the velocity of the joint is also nearly zero at this point, the kinetic energy is consequently almost nearly zero. This phase runs from time 0 to about 2.9 seconds.

2. **During Application of the Force:** At about 2.9 seconds, the external force is applied (denoted by the shaded area). This is reflected in both a change in the position of the joint, and the change in energy levels. As the error increases (to a maximum value of 0.47 radians), there is an increase in the potential energy of the robot while the kinetic energy stays relatively constant. This increase in the robot's energy leads to the robot's proportional gain decreasing, and the robot's energy hence does not exceed the energy limit.

3. **After Application of the Force:** As the external force is removed, the error starts decreasing, the potential energy of the robot starts decreasing. The proportional gains slowly start increasing back to their nominal values. The reason the joint position and proportional gain are slightly fluctuating during this time is because the robot encounters friction during this time, and that leads to the joint position error being slightly wobbly, and in turn the proportional gain being wobbly as well (note that this is not a problem we encounter in simulation, and something that should be addressed in future work). The robot gradually returns to the desired joint position, taking 4.3 seconds after the force is released to go back to within 5% of the desired position.

As compared to the simple PD controller, there are some differences in the performance of the proposed adaptive PD controller when subjected to a small external force:

- In this case the robot's arm moves to a greater extent when subjected to an external force (0.47 radians as compared to the baseline controller's 0.18 radians). Since the external force is not measured explicitly, it is tougher to make a direct comparison with the simple PD controller. However, comparing the graphs for the joint torques, we see that the joint torque in this case (peak of 0.76 Nm) is lower as compared to the simple PD controller (peak of 1.37 Nm). We may hence conclude that the arm acts 'softer' or more compliant when compared to the simple PD controller. This is a desirable quality from the pHRI standpoint. This 'softness' was also felt by the experimenters when interacting with the robot, as experimenters noted that the robot was noticeably easier to move in this case as compared to the simple PD controller, especially as the joint position error increased.

- Due to the controller's adaptive gains, we see that the given energy limit is not violated. This is despite the fact that the deflection seen in the joint position is higher as compared to the baseline controller. A consequence of this energy limitation is reflected in the joint torques being lower when compared to the baseline case. Energy limitation is hence achieved (in this case, it is mostly the robot's potential energy being managed by the controller).

- The controller does sacrifice tracking performance, since the controller takes much longer to get back to desired joint position (4.3 seconds, as compared to the baseline controller's settling time of 1.52 seconds). This is due to the fact that the proportional gains decreased adaptively, and increased gradually to conform to the energy limit.

In summary, the proposed adaptive PD controller succeeds in limiting the energy of the robot below the proposed energy limit. Particularly, the potential energy of the robot is limited by reducing the proportional gains of the robot. The robot also behaves softer in this case, requiring less effort to move the robot the same amount of distance. This is good from a pHRI safety perspective, though the tracking performance is sacrificed to an extent due to the position tracking taking longer in this case. In the next section, we will analyze the response of the proposed adaptive PD controller to larger external forces.

Larger External Force

For this experiment, a sudden force is applied by the experimenter to the robot's arm at link 1, as shown in figure 6.7. The graph that shows this interaction is shown in figure 6.10.

Once more, to analyze this plot, we can divide the interaction into three key phases:

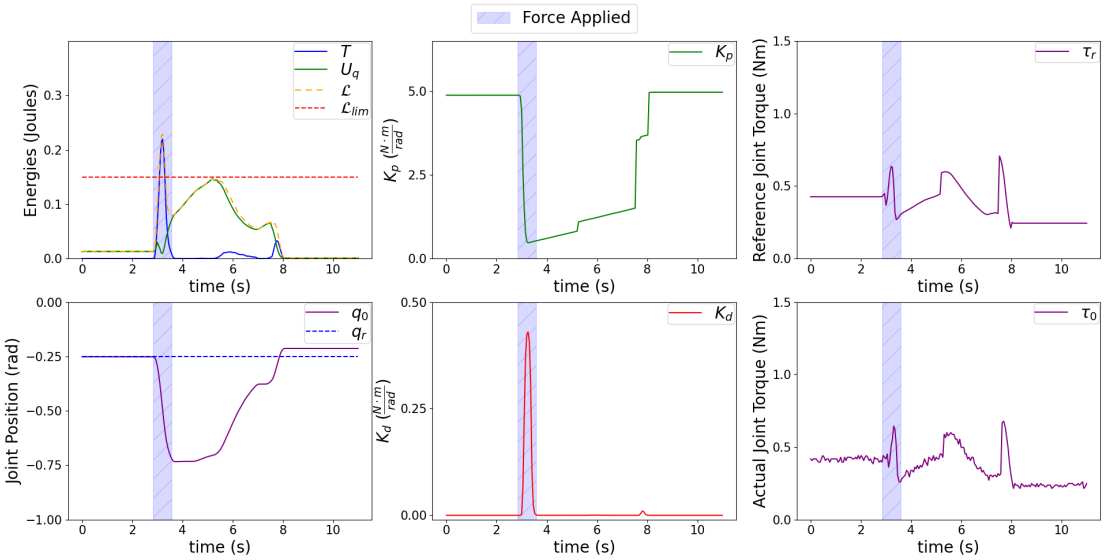

Figure 6.10: The Proposed Adaptive PD Controller's Response to a Large External Force - Hardware Result

1. **Before Application of the Force:** Taking a look at the plot, initially, the robot arm is holding its target joint position of -0.25 radians. Since the error between the desired position and actual position is nearly zero, the potential energy at this point is consequently almost zero. Since the velocity of the joint is also nearly zero at this point, the kinetic energy is consequently almost nearly zero. This phase runs from time 0 to about 2.8 seconds.

2. **During Application of the Force:** At about 2.8 seconds, the external force is applied (denoted by the shaded area). This is reflected in both a change in the position of the joint, and the change in energy levels. This time since the force is much larger and sudden, the energy change is mostly reflected in the kinetic energy. The energy change is also much more sudden this time, with the energy limit being crossed within 0.3 seconds of the application of the external force. We see that the kinetic energy (and consequently the total energy) of the robot exceeds the energy limit. However, eventually the derivative gain does succeed in arresting the motion of the robot, and the kinetic energy of the robot decreases. Note that the potential energy of the robot is successfully limited due to the decrease in proportional gains. However, limiting the kinetic energy in this case proves more difficult, and the kinetic energy does violate the energy limit before going back down (this is discussed in much more detail in the observations below).

3. **After Application of the Force:** As the external force is removed, the error starts decreasing, the total energy of the robot starts decreasing. This leaves a 'gap' in the energy budget, which is filled up by the potential energy increasing due to a slow increase in the proportional gains. Once again due

to the joint friction, the joint trajectory is not completely smooth, and some jumps are seen within the proportional gain as a consequence. The robot slowly and gradually returns back to its desired position, settling within 5% of the desired position within 4.26 seconds.

We note some interesting things about the proposed adaptive PD controller's performance when subjected to a large external force:

- The controller does not immediately succeed in energy-limitation for the robot. The kinetic energy initially violates the defined energy limit, and the controller is only able to bring the robot's energy under the limit after a delay of 0.2 seconds. This is due to the limit of the derivative gain being too low for this force, as well as perhaps some more suitable tuning for the α and β values. This is one of the limitations of this approach - it is highly dependent on what value we select as the maximum value for the derivative gain ($K_{d_{max}}$) to oppose the kinetic energy. We will demonstrate that the energy violation is indeed due to the low values of the derivative gains by replicating such a result in simulation. This will be done when discussing the limitations of our controller

- While pushing the robot, a noticeable resistance was felt by the experimenters when pushing the robot with a high force, due to the increase in damping caused by the increase in derivative gains. Clearly, the controller responds with a higher joint torque in the direction opposite to the applied force due to the increase in the robot's damping when the kinetic energy approaches the energy limit. However, in this case, the maximum values of the derivative gain were not high enough to completely contain the robot's energy below the limit. Hence we see the importance of the ($K_{d_{max}}$) value being tuned correctly. The values of α and β being more appropriately tuned (higher values for α, lower value of β) might also help in the kinetic energy limitation. We see that the maximum value of the derivative gain does not quite reach the specified value of 0.5, and higher values of α and lower values of β will lead to the derivative gain reaching the maximum value earlier. While the controller will not violate the energy limit while tracking a position, external forces can potentially violate the kinetic energy limit if they are large enough to overcome the maximum damping gain $K_{d_{max}}$, or if the α and β values are not tuned appropriately.

- Once again, the controller does sacrifice tracking performance, since the controller takes much longer to get back to the desired joint position. This is due to the fact that the proportional gains decreased adaptively and increased back gradually to conform to the energy limit. However, this is a natural consequence of energy limitation, and could potentially be a desirable quality for pHRI.

In summary, the proposed adaptive PD controller succeeds in limiting the energy of the robot after a 0.2

second delay, but one key point is identified - the limitation of kinetic energy is highly dependent on the maximum value of the derivative gain ($K_{d_{max}}$). The smaller gains very noticeably offer resistance to the external force, but may not be able to restrict the kinetic energy if the force is much larger.

6.3 Limitations of the Controller

6.3.1 Access to the Robot Model

The first obvious limitation of this controller is that we need access to an acceptably accurate robot model for the robot dynamics. To calculate the kinetic energy of the robot, we need to model and calculate the mass matrix ($M(q)$). In addition, gravity compensation is essential for the controller to be stable, and this requires access to the gravity vector ($G(q)$) for the robot. This access to an accurate dynamic model of the robot could hence prove a limitation for the implementation of this controller.

6.3.2 High External Forces

One of the limitations of our proposed adaptive PD controller is that the successful limitation of kinetic energy is highly dependent on the $K_{d_{max}}$ value - the maximum value for the derivative gain to oppose forces that induce high kinetic energy values. Appropriate tuning of other controller parameters such as the α and β values are also important in this regard. This became clear when analyzing the TALOS robot's hardware response to high external forces (figure 6.10). In that case, due to the $K_{d_{max}}$ value not being high enough, the controller does not quite succeed in limiting the robot's energy. To illustrate that this is indeed the case, we replicate a similar response in the simulation setup.

For this test, we apply a force of 50 N as the external force. The rest of the controller parameters remain the same as the simulation setup, and in particular, with $K_{d_{max}} = [20,\ 20,\ 20,\ 20,\ 20,\ 20,\ 20] \quad \frac{N \cdot m}{rad}$. The graph that shows this interaction is shown in figure 6.11. As can be seen, there is a very direct similarity between the energy graphs in figure 6.10 and 6.11. When the force is applied to the arm, the robot's arm moves very fast, leading to a fast rise in the robot's kinetic energy. While the derivative gains rise to dampen the robot's velocity, they fail to limit the robot's energy under the limit. The robot's energy stays above the limit until the applied force is removed, and the robot's energy immediately returns under the limit.

To compare, we run another test in simulation with the same parameters as before, but this time with a higher value of $K_{d_{max}} = [30,\ 30,\ 30,\ 30,\ 30,\ 30,\ 30] \quad \frac{N \cdot m}{rad}$. This result is shown in Figure 6.12. As can be

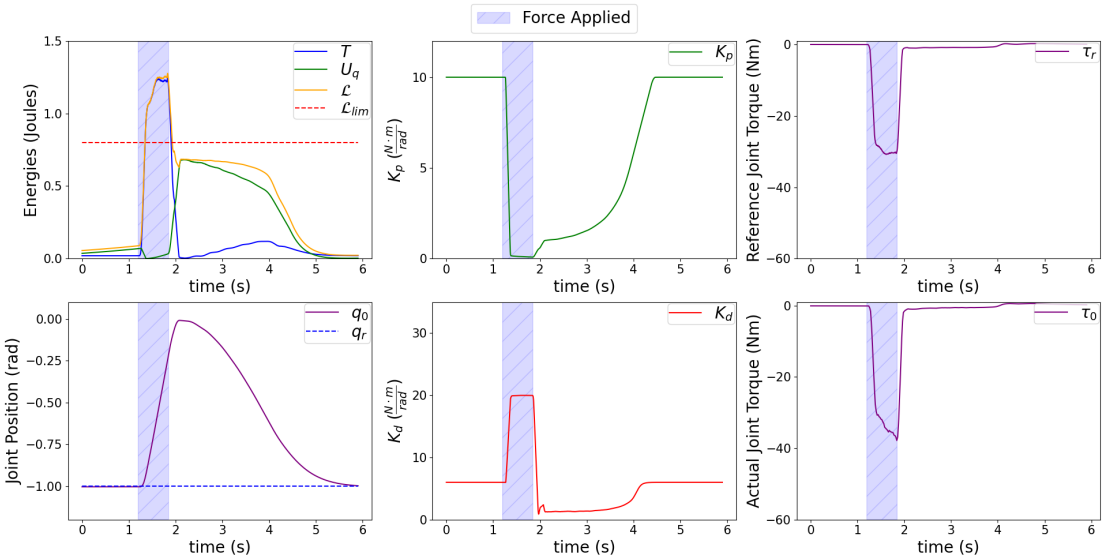

Figure 6.11: The Proposed Adaptive PD Controller's Response to a 50 N force (Simulation)

seen, this time the controller is able to successfully limit the robot's energy under the energy limit, because the kinetic energy is resisted by a higher damping, due to a higher value of $K_{d_{max}}$. This is hence one of

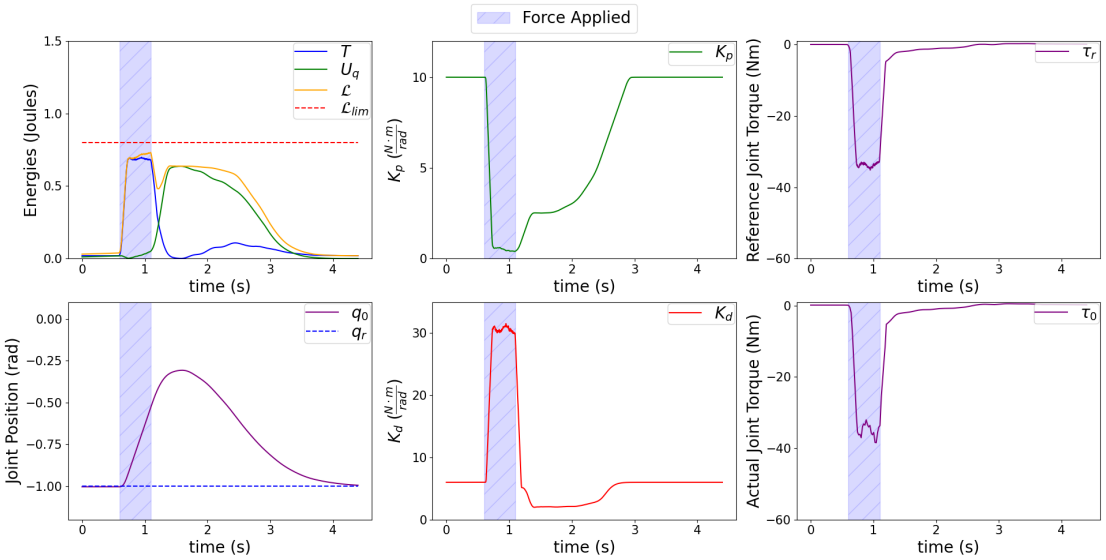

Figure 6.12: The Proposed Adaptive PD Controller's Response to a 50 N force, higher $K_{d_{max}}$ (Simulation)

the controller's limitations. While the controller will not violate the robot's energy limit while tracking a desired position in the absence of an external force, high forces may result in the kinetic energy violating the energy limit if the $K_{d_{max}}$ value is too low to oppose this action. Since we want to avoid very high values of derivative gains to avoid the arm being completely immovable, the appropriate selection of the $K_{d_{max}}$ hence becomes highly important.

It is worth noting however that this is not necessarily a downside of the controller. A very high resistance to movement, even in the presence of large external forces, may not be necessarily desirable in the pHRI context. This case is also discussed in [3], where to counter excessive kinetic energy, they propose a similar solution. Instead of increasing the joint damping, they propose an additional joint potential (an additional joint stiffness component, in addition to the generic proportional gain) to counter excessive kinetic energy. It is noted that this condition is necessary in clamping scenarios - if the robot is pushed in a clamping scenario, it should not have excessive kinetic energy. However, [3] also note that this extension is important primarily to reduce the uncontrollable motion of the robot in response to a push, rather than limiting the kinetic energy itself. From the point-of-view of our controller as well, a high stiffness and damping might mean that the robot is unable to be pushed away at the cost of reducing kinetic energy, which could be highly disadvantageous during quasi-static contact. We hence choose to leave the maximum derivative gain value $K_{d_{max}}$ as a tunable parameter, subject to the desired behavior of the robot.

6.4 Summarizing the Features and Limitations of the Proposed Adaptive PD Controller

In this section, we aim to summarize the main advantages and limitations of our proposed adaptive PD controller. The controller succeeds in limiting the energy of the robot under a given limit, given that the applied force is not large enough to overcome the specified maximum damping value $K_{d_{max}}$. We can limit both kinetic and potential energy, and the total energy of the manipulator will generally stay under the predefined limit. This energy-limitation in turn leads to some desirable characteristics from a pHRI perspective - the robot is soft or compliant when subjected to smaller external forces, whereas large and abrupt forces cause the joint damping to increase. We also developed a stability proof for the controller, and obtained a stability constraint.

On the other hand, there are some limits to the controller's performance. The limitation of kinetic energy is dependent on the maximum value of the derivative gains ($K_{d_{max}}$), since larger forces will be able to overcome the energy limit via excessive kinetic energy. However, this is not necessarily a downside since we don't want damping to be too high, as it could lead to harmful collisions. The controller behavior also generally depends on the chosen values of α and β, and ill-chosen values for these tunable parameters (α too small, β too close to the energy limit) may lead to the controller not completely limiting the robot's energy.

These features and limitations are summarized below in table 6.1.

The Proposed Adaptive PD Controller	
Features	Limitations
Total Energy Limitation	Appropriate tuning of α and β values is required for proper energy limitation
Can limit both kinetic and potential energies	Kinetic Energy Limitation is Dependent on $K_{d_{max}}$
Soft behavior with gentler forces (forces that do not induce kinetic energy close to the energy limit)	
Dampens sudden/abrupt forces to avoid unstable behavior (the robot moving too fast to stabilize, could cause harmful collisions)	

Table 6.1: The Features and Limitations of our Proposed Adaptive PD Controller

Chapter 7

Future Work and Conclusion

In this section, we discuss the potential future work to be done on the controller. This is presented in the context of the current features and limitations of our proposed controller as discussed in Chapter 6. We then conclude this thesis by summarizing the work, results, and learning, and discussing the controller in the broader context of robotics and safe pHRI.

7.1 Future Work

In this section, we highlight some of the future work that should be done to test and refine our proposed controller further.

7.1.1 Inertia-Based Energy Budget Allocation

One of the problems that may need to be explored further is the joint-wise energy budget allocation for a controlled joint. If a complete manipulator is allocated an energy budget, do we simply work with the entire budget at a time, or should the budget be further broken down and allocated to each joint? This would be an interesting question to explore, and one of the proposed solutions we have is to allocate the budget to each of the joints based on their respective inertia values. Joints associated with high inertia links would be allocated a higher budget, whereas joints associated with lower inertia links would be allocated a lower budget. This is simply a proposed solution, and much more theoretical and practical assessment would be necessary before reaching any specific conclusions regarding such a method.

7.1.2 Testing the Controller - Friction Compensation

One of the additional tests we would like to conduct would be for our proposed PD controller working in conjunction with a friction compensation controller. This is due to the fact that the TALOS Robot already has a friction compensation controller implemented for it by PAL Robotics. Compensating for the joint friction would also help address the wobbly behavior seen during hardware testing of the controller. Some very preliminary tests were conducted during my brief time at the University of Waterloo, using the friction compensation control. Friction compensation control models the various sources of friction within the robot joints, and compensates for them via joint torques. This control layer acts between any implemented controller and the hardware, so that the upper-level controller may feel minimal effects of joint friction. Using this controller, we will be able to test with higher values of derivative gains, since there is negligible natural damping when friction is being compensated. We would like to further test our proposed controller comprehensively with friction compensation in place.

7.1.3 Conservation of PD Parameters

One aspect of our controller that we also want to explore in the future is the conservation of PD parameters. In particular, we want to conserve the ratios for the proportional and derivative gains. Under the current scheme, our derivative gain rises to its nominal value much faster than the proportional gains. This means that while the gains are returning to their nominal values, the ratio between the proportional and derivative gains are not preserved. This leads to two quantities changing:

- The natural frequency (ω_n). For a second order system, this is defined as:

$$\omega_n = \sqrt{\frac{K_p}{m}} \tag{7.1}$$

- The damping ratio (ζ_n). For a second order system, this is defined as:

$$\zeta_n = \frac{K_d}{2\sqrt{K_p m}} \tag{7.2}$$

We are particularly interested in the damping ratio for the controller, since if the damping ratio becomes very high, we could have instabilities occur. This is due to the fact that the derivative gain is sensitive to noise, and noise or disturbances with a high derivative gain (and consequently a high damping ratio) may cause unstable behavior. A high derivative gain as compared to the proportional gain may also oppose what the proportional gain is aiming to do. We would thus like to explore the effects of limiting the ratio of the

proportional and derivative gains on our proposed controller.

7.1.4 Collision Testing

The current tests we have performed are mostly with regards to the robot's arm being pushed away from a reference position. We would like to conduct further testing with the controller, in particular with regards to collision testing. We would like to test the controller's response when the robot's arm is moving while tracking a reference path, and the arm collides with an object in its path. The energy and safety aspects of the controller can then be better understood from a pHRI perspective.

7.2 Conclusion

7.2.1 Summary of Work and Results

The thesis proposes an adaptive PD controller that can limit the total energy of the robot under a given energy limit. We discuss why energy is a convenient quantity to work with in terms of pHRI, and the current literature around energy-based analysis of pHRI. We discuss an adaptive control law that leads to both the joint potential energy and the kinetic energy of the robot being limited. The major contributions of this thesis may be summarized as follows:

- A novel adaptive PD control approach to limit the robot's energy under any given limit.
- A Lyapunov analysis of the proposed controller, and a stability condition to ensure stability of the controller.
- Controller performance results, shown on both simulation and hardware, verifying the expected behavior of the robot.

After presenting literature surrounding compliant control and energy-based control, we present the advantages of energy-based control as compared to traditional compliant controllers. We present some desirable characteristics of controllers in the context of pHRI. In this context, an adaptive PD controller is proposed, with proportional and derivative gains that change adaptively based on the current energy of the system. The gains change in a manner that aims to limit the robot's total energy. The controller is shown to be capable of limiting the robot's energy under a given energy limit, provided that the external force is not too high. A Lyapunov stability analysis is presented for the proposed adaptive PD controller. Through this analysis, we obtain a condition for ensuring the proposed controller's stability. We then present an implementation of this proposed adaptive PD controller in code, discussing some of the key parts of the code

in this thesis. Finally, we present the performance of our proposed adaptive PD controller on the TALOS robot, in both simulation and on hardware. We show that the controller is indeed capable of limiting the robot's energy under a given energy limit, provided that the external applied forces are not high enough to overcome maximum damping values (and eventually, the maximum joint torque values).

In our research, we learn that a robot's energy may be used to draw a natural correspondence to compliant behavior. The potential energy is directly related to the stiffness of a robot, and a robot's damping may relate to a robot's resistance to kinetic energy. By limiting the robot's energy as we propose, compliant behavior is shown by the robot without it being specified explicitly. In our case, the robot's arm acts softer - we are able to push the robot's arm with more ease than we would be able to with a traditional PD controller. However, the robot also displays additional damping when pushed with a high enough force (a force that induces a kinetic energy close to the energy limit), to avoid unstable behavior by moving at high joint velocities. Energy as a quantity provides a compelling alternative to force-based safety standards. While literature in this space is fairly recent, work is being done regarding all aspects of energy-based safe pHRI (such as energy-limiting control schemes, calculating supplied energy during pHRI accurately).

7.2.2 Broader Context - Applications in Robotics

Physical Human-Robot Interaction (pHRI) is an active area of research, and safety is of utmost importance if physically interacting robots are to become more commonplace. Applications such as ones in healthcare and manufacturing require all physical interaction to be safe. The controller we propose in this thesis is one small step towards that goal. A potential scenario where the proposed controller might be useful is outlined as follows: One possible application of robots within healthcare is one where a robotic manipulator might interact with patients to perform a small medical procedure. For this particular application, appropriate allowable limits on energy transfers may be determined. Once that is done, the controller may be implemented for that particular robotic system, with appropriately tuned parameters according to the desired behavior. The hope is that the robots will then be able to interact with humans in a safe manner. In this case, the robotic manipulator would be able to perform the medical procedure, and would also display safer behavior in case of any collisions or external pushes.

We hope that future work in this area may be able to build upon the controller we have proposed in this thesis, leading to developments in the safe pHRI domain.

Bibliography

[1] M. Schumacher, J. Wojtusch, P. Beckerle, and O. von Stryk, "An introductory review of active compliant control," *Robot. Auton. Syst.*, vol. 119, p. 185–200, Sept. 2019.

[2] A. Calanca, R. Muradore, and P. Fiorini, "A review of algorithms for compliant control of stiff and fixed-compliance robots," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 2, pp. 613–624, 2016.

[3] J. Lachner, F. Allmendinger, N. Hogan, and S. Stramigioli, "Energy budgets for coordinate invariant robot control in physical human–robot interaction," *The International Journal of Robotics Research*, vol. 40, 05 2021.

[4] S. Calinon, *Learning from Demonstration (Programming by Demonstration)*, pp. 1–8. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018.

[5] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems* (S. Becker, S. Thrun, and K. Obermayer, eds.), vol. 15, MIT Press, 2002.

[6] D. Silvera-Tawil, "Robotics in healthcare: A survey," *SN Computer Science*, vol. 5, 01 2024.

[7] R. Jahanmahin, S. Masoud, J. Rickli, and A. Djuric, "Human-robot interactions in manufacturing: A survey of human behavior modeling," *Robotics and Computer-Integrated Manufacturing*, vol. 78, p. 102404, 2022.

[8] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and Autonomous Systems*, vol. 42, no. 3, pp. 143–166, 2003. Socially Interactive Robots.

[9] I. Berk-Smeekens, M. Dongen-Boomsma, M. de Korte, J. Boer, I. Oosterling, N. Peters-Scheffer, J. Buitelaar, E. Barakova, T. Lourens, W. Staal, and J. Glennon, "Adherence and acceptability of

a robot-assisted pivotal response treatment protocol for children with autism spectrum disorder," *Scientific Reports*, vol. 10, 05 2020.

[10] P. Pennisi, A. Tonacci, G. Tartarisco, L. Billeci, L. Ruta, S. Gangemi, and G. Pioggia, "Autism and social robotics: A systematic review," *Autism Research*, vol. 9, no. 2, pp. 165–183, 2016.

[11] M.-A. Williams and M. J. C. Chair, "Designing human-robot interaction with social intelligence," in *2021 16th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 3–4, 2021.

[12] S. Jain and B. Argall, "Probabilistic human intent recognition for shared autonomy in assistive robotics," *J. Hum.-Robot Interact.*, vol. 9, Dec. 2019.

[13] S. Iregui, J. De Schutter, and E. Aertbeliën, "Reconfigurable constraint-based reactive framework for assistive robotics with adaptable levels of autonomy," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7397–7405, 2021.

[14] A. Bicchi, "Of robots, humans, bodies and intelligence: Body languages for human robot interaction," in *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 1–1, 2015.

[15] A. Zgonnikov, S. Thill, P. Beckerle, and C. M. Jonker, "Modeling human behavior in human-robot interactions," in *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 1296–1297, 2022.

[16] A. De Santis, B. Siciliano, A. Luca, and A. Bicchi, "An atlas of physical human-robot interaction," *Mechanism and Machine Theory*, vol. 43, pp. 253–270, 03 2008.

[17] M. Farajtabar and M. Charbonneau, "The path towards contact-based physical human–robot interaction," *Robotics and Autonomous Systems*, vol. 182, p. 104829, 2024.

[18] S. Restivo, "Bringing up and booting up: social theory and the emergence of socially intelligent robots," vol. 4, pp. 2110 – 2117 vol.4, 02 2001.

[19] A. Q. Keemink, H. van der Kooij, and A. H. Stienen, "Admittance control for physical human–robot interaction," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1421–1444, 2018.

[20] R. Rawassizadeh, T. Sen, S. Kim, C. Meurisch, H. Keshavarz, M. Mühlhäuser, and M. Pazzani, "Manifestation of virtual assistants and robots into daily life: vision and challenges," *CCF Transactions on Pervasive Computing and Interaction*, vol. 1, 10 2019.

[21] A. Pervez and J. Ryu, "Safe physical human robot interaction-past, present and future," *Journal of Mechanical Science and Technology*, vol. 22, pp. 469–483, 2008.

[22] A. Bicchi, M. A. Peshkin, and J. E. Colgate, *Safety for Physical Human–Robot Interaction*, pp. 1335–1348. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[23] F. Benzi, F. Ferraguti, and C. Secchi, "Energy tank-based control framework for satisfying the iso/ts 15066 constraint," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1288–1293, 2023. 22nd IFAC World Congress.

[24] M. Fritzsche, N. Elkmann, and E. Schulenburg, "Tactile sensing: A key technology for safe physical human robot interaction," in *2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 139–140, 2011.

[25] S. A. B. Birjandi, J. Kühn, and S. Haddadin, "Observer-extended direct method for collision monitoring in robot manipulators using proprioception and imu sensing," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 954–961, 2020.

[26] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the dlr-iii lightweight manipulator arm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1623–1630, 2006.

[27] I. 15066:2016, "Robots and robotic devices - collaborative robots," 2016.

[28] R. Kelly, V. Davila, and J. Perez, *Control of Robot Manipulators in Joint Space*. Advanced Textbooks in Control and Signal Processing, Springer London, 2005.

[29] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2005.

[30] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer US, 2014.

[31] M. L. Felis, "Rbdl: an efficient rigid-body dynamics library using recursive algorithms," *Autonomous Robots*, pp. 1–17, 2016.

[32] C. Ott, "Cartesian impedance control of redundant and flexible-joint robots," in *Springer Tracts in Advanced Robotics*, 2008.

[33] J. Won, S. Stramigioli, and N. Hogan, "Comment on "the equivalence of second-order impedance control and proportional gain explicit force control"," *The International Journal of Robotics Research*, vol. 16, no. 6, pp. 873–875, 1997.

[34] M. Lemmon, "Passivity based control," 2017. https://www3.nd.edu/~lemmon/courses/ee580/lectures/chapter10.pdf, Last accessed on 2024-10-15.

[35] H. Berghuis, P. Löhnberg, and H. Nijmeijer, "Adaptive 'pd+' control of rigid robot manipulators," *IFAC Proceedings Volumes*, vol. 24, no. 9, pp. 153–158, 1991. 3rd IFAC Symposium on Robot Control 1991 (SYROCO'91), Vienna, Austria, 16-18 September 1991.

[36] M. H. Raibert and J. J. Craig, "Hybrid Position/Force Control of Manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 103, pp. 126–133, 06 1981.

[37] Y. Lu, Y. Shen, and Z. Chungang, "External force estimation for industrial robots using configuration optimization," *Automatika*, vol. 64, pp. 1–24, 01 2023.

[38] M. Van Damme, P. Beyl, B. Vanderborght, V. Grosu, R. Van Ham, I. Vanderniepen, A. Matthys, and D. Lefeber, "Estimating robot end-effector force from noisy actuator torque measurements," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1108–1113, 2011.

[39] B. PADEN and R. PANJA, "Globally asymptotically stable 'pd+' controller for robot manipulators," *International Journal of Control*, vol. 47, no. 6, pp. 1697–1712, 1988.

[40] R. Ortega, A. Van Der Schaft, I. Mareels, and B. Maschke, "Putting energy back in control," *IEEE Control Systems Magazine*, vol. 21, no. 2, pp. 18–33, 2001.

[41] H. Khalil, *Nonlinear Systems*. Pearson Education, Prentice Hall, 2002.

[42] H. Marquez, "Nonlinear control systems: Analysis and design," 11 2002.

[43] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control*. Interdisciplinary Applied Mathematics, Springer New York, 2013.

[44] M. Vidyasagar, *Nonlinear Systems Analysis: Second Edition*. Classics in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2002.

[45] J. Buchli, E. Theodorou, F. Stulp, and S. Schaal, "Variable impedance control a reinforcement learning approach," 07 2010.

[46] T. Tsumugiwa, R. Yokogawa, and K. Hara, "Variable impedance control based on estimation of human arm stiffness for human-robot cooperative calligraphic task," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, pp. 644–650 vol.1, 2002.

[47] J. Wade, T. Bhattacharjee, R. D. Williams, and C. C. Kemp, "A force and thermal sensing skin for robots in human environments," *Robotics and Autonomous Systems*, vol. 96, pp. 1–14, 2017.

[48] J. Liu, Y. Yamada, and Y. Akiyama, "Calculating the supplied energy for physical human-robot interaction," in *2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*, pp. 157–160, 2021.

[49] F. Ferraguti, C. Secchi, and C. Fantuzzi, "A tank-based approach to impedance control with variable stiffness," in *2013 IEEE International Conference on Robotics and Automation*, pp. 4948–4953, 2013.

[50] M. Sharifi, A. Zakerimanesh, J. K. Mehr, A. Torabi, V. K. Mushahwar, and M. Tavakoli, "Impedance variation and learning strategies in human–robot interaction," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 6462–6475, 2022.

[51] F. Ferraguti, M. Bertuletti, C. T. Landi, M. Bonfè, C. Fantuzzi, and C. Secchi, "A control barrier function approach for maximizing performance while fulfilling to iso/ts 15066 regulations," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5921–5928, 2020.

[52] PAL Robotics, "The TALOS Robot." https://pal-robotics.com/robot/talos/, Last accessed on 2024-11-29.

[53] S. Kawamura, F. Miyazaki, and S. Arimoto, "Is a local linear pd feedback control law effective for trajectory tracking of robot motion?," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pp. 1335–1340 vol.3, 1988.

[54] Pollen Robotics, "The Reachy Robot." https://www.pollen-robotics.com/reachy/, Last accessed on 2024-12-10.

[55] Universal Robots, "Develop with ROS/ROS2." https://www.universal-robots.com/developer/communication-protocol/ros-and-ros2-driver/, Last accessed on 2024-12-10.

[56] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," vol. 3, 01 2009.

[57] A. Koubaa, *Robot Operating System (ROS): The Complete Reference (Volume 3)*. Studies in Computational Intelligence, Springer International Publishing, 2018.

[58] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[59] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ros_control: A generic and simple control framework for ros," *The Journal of Open Source Software*, 2017.

[60] Danyal Saqib, Marie Charbonneau, "Github Repository - ECAPD Controller." https://github.com/Calgary-Human-Robot-Interaction-Lab/custom_controller_talos_simulation, Last accessed on 2024-12-10.

[61] D. Tola and P. Corke, "Understanding urdf: A survey based on user experience," 2023.

[62] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154 vol.3, 2004.

[63] Víctor López, Sai Kishor Kothakota, "Installing TALOS Simulation." https://wiki.ros.org/Robots/TALOS/Tutorials/Installation/Simulation, Last accessed on 2024-11-10.

Appendix A

Summarizing the Controller

This section summarizes the equations and tunable parameters that define our novel proposed controller.

A.0.1 Some initial gains

- $K_{po_{diag}}$ - Initial proportional gains, tuned using any method
- $K_{do_{diag}}$ - Initial derivative gains, tuned using any method
- $K_{d_{max}}$ - Column vector for upper limit on derivative gains, defining resistance to external pushes

A.0.2 Behavior defining parameters

- $\mathcal{L}_{lim}$ - Upper limit on system energy
- α - Defines how fast we want the system to respond, as energy approaches limit - higher value corresponds to faster change in behavior
- β - Defines how close to the energy limit, the system behavior changes ($\beta < \mathcal{L}_{lim}$)
- η - Defines how close to the stability condition the controller operates ($0 < \eta \leq 1$)

A.1 Controller Equations

$$r = \frac{1}{1 + \left(\frac{\mathcal{L}}{\beta}\right)^{\alpha}} \quad , \quad r_T = \frac{1}{1 + \left(\frac{T}{\beta}\right)^{\alpha}}$$

A.1.1 Proportional Gain - K_p

$$p_1 = [r \cdot K_{po}] + \left[(1-r) \cdot \frac{\mathcal{L}_{lim}}{n}\right] \quad , \quad p_2 = \frac{1}{1 + (1-r) \cdot q_e^2}$$

$$K_{pc} = p_1 \cdot p_2$$

$$K_p = diag(K_{pc})$$

A.1.2 Proportional Gain - K_d

$$d_1 = r \cdot K_{do} \quad , \quad d_2 = \frac{1}{1 + (1-r) \cdot \dot{q}^2} \quad , \quad d_3 = (1 - r_T) \cdot K_{d_{max}}$$

$$K_{dc} = (d_1 \cdot d_2) + d_3$$

$$K_d = diag(K_{dc})$$

A.1.3 The Torque Equation

$$\tau = -K_p \; q_e - K_d \; \dot{q}_e + G(q)$$

A.2 Stability Condition

$$\dot{V} = -\dot{q}_e^\intercal K_d(t) \; \dot{q}_e + \frac{1}{2} \; q_e^\intercal \; \dot{K}_p(t) \; q_e < 0$$

$$\boxed{\implies \dot{q}_e^\intercal K_d(t) \; \dot{q}_e > \frac{1}{2} \; q_e^\intercal \; \dot{K}_p(t) \; q_e}$$

A.2.1 Stability Related Variables

$$a = \dot{q}_e^\intercal K_{d_c} \; \dot{q}_e \quad , \quad \dot{K}_p(t) = \frac{K_{p_c} - K_{p_p}}{dt} \quad , \quad b = \frac{1}{2} \; q_e^\intercal \; \dot{K}_p(t) \; q_e$$

$$s = \frac{a}{b}$$

If $s \leq 1$:

$$K'_{p\_c} = \eta(sK_{p_c} + (1-s) \, K_{p_p})$$

Appendix B

The Controller Code - ROS Control Library

B.1 The Controller Logic

Code: joint_group_position_controller_modded.cpp

```
// IMPROTANT - Algorithms for setting gains adaptively
 void JointGroupPositionControllerGC::set_adaptive_pid_gains(double KineticEnergy, double
     PotentialEnergy, double nominalKineticEnergy, VectorNd position_error_array, VectorNd
     current_velocity_array, double L_lim, double beta, double alpha, bool ke_exceeded)
 {
   VectorNd K_p_adaptive = VectorNd::Zero (rbdl_model->q_size);
   VectorNd K_d_adaptive = VectorNd::Zero (rbdl_model->q_size);

   ke_exceeded = false;
   if (ke_exceeded_written == true)
   {
     ke_exceeded_written = false;
   }

   double gamma = (nominalKineticEnergy + PotentialEnergy) / (KineticEnergy + PotentialEnergy);
   double L_d_lim = L_lim;
   double L_d_m = beta;
```

```
    double L_p = KineticEnergy + PotentialEnergy;
    double r = 1 / (1 + (pow((L_p / L_d_m), alpha)));
    int n_dof = rbdl_model->q_size;

    // Adaptive K_p
    VectorNd qe_sqrd = JointGroupPositionControllerGC::square_array(position_error_array);
    VectorNd p1 = JointGroupPositionControllerGC::add_scalar_to_array((r * initial_p_vector), ((1 - r)
        * (L_d_lim / n_dof)));
    VectorNd p2 = JointGroupPositionControllerGC::divide_scalar_by_array(1,
        JointGroupPositionControllerGC::add_scalar_to_array(((1 - r) * qe_sqrd), 1));
    K_p_adaptive = JointGroupPositionControllerGC::array_multiply_elementwise(p1, p2);

    // Adaptive K_d
    VectorNd qdot_a_sqrd = JointGroupPositionControllerGC::square_array(current_velocity_array);
    VectorNd d1 = r * initial_d_vector;
    VectorNd d2 = JointGroupPositionControllerGC::divide_scalar_by_array(1,
        JointGroupPositionControllerGC::add_scalar_to_array(((1 - r) * qdot_a_sqrd), 1));
    K_d_adaptive = JointGroupPositionControllerGC::array_multiply_elementwise(d1, d2);
    double r_ke = 1 / (1 + (pow((KineticEnergy / L_d_m), alpha)));
    K_d_adaptive = K_d_adaptive + (1 - r_ke) * ke_exceeded_d_gains;
```

B.2 The Stability Condition

Code: joint_group_position_controller_modded.cpp

```
    // Checking stability condition
    VectorNd q_e_dot = (position_error_array - error_prev) / dt;
    double a_rate = q_e_dot.transpose() *
        JointGroupPositionControllerGC::square_matrix_from_diagonals(K_d_adaptive) * q_e_dot;
    VectorNd K_p_dot = (K_p_adaptive - K_p_adaptive_prev) / dt;
    double b_rate = 0.5 * position_error_array.transpose() *
        JointGroupPositionControllerGC::square_matrix_from_diagonals(K_p_dot) * position_error_array;
    double s = a_rate / b_rate;

    msg_controller_info.a_rate = a_rate;
```

```
    msg_controller_info.b_rate = b_rate;
    msg_controller_info.s_ratio = s;

    if(b_rate == 0)
    {
      // std::cout << "Entering first if condition (w.r.t b_rate)!" << std::endl;
      K_p_adaptive = K_p_adaptive_prev;
    }
    else if(b_rate > 0)
    {
      // std::cout << "Entering second if condition (w.r.t a_rate and b_rate)!" << std::endl;
      K_p_adaptive = JointGroupPositionControllerGC::add_scalar_to_array(K_p_adaptive_prev,
          gain_rise_rate);
    }

    VectorNd K_p_dot_modified = (K_p_adaptive - K_p_adaptive_prev) / dt;
    double b_rate_modified = 0.5 * position_error_array.transpose() *
        JointGroupPositionControllerGC::square_matrix_from_diagonals(K_p_dot_modified) *
        position_error_array;
    msg_controller_info.b_rate_modified = b_rate_modified;

    double p_gain;
    double i_gain;
    double d_gain;
    double i_max_gain;
    double i_min_gain;
    bool antiwindup_bool;

    for(unsigned int i=0; i<n_joints_; i++)
    {
        pid_controllers_[i].getGains(p_gain, i_gain, d_gain, i_max_gain, i_min_gain, antiwindup_bool);
        pid_controllers_[i].setGains(K_p_adaptive[i], 0, K_d_adaptive[i], 0, 0, antiwindup_bool);
    }

    K_p_adaptive_prev = K_p_adaptive;
```

```
41    K_d_adaptive_prev = K_d_adaptive;
42    error_prev = position_error_array;
43
44  }
```

B.3 The Update Loop

Code: joint_group_position_controller_modded.cpp

```
1   // IMPROTANT - The Update Loop
2   void JointGroupPositionControllerGC::updateExtras(const ros::Time& time, const ros::Duration& period)
3   {
4     // Get robot state
5     VectorNd current_position_array = JointGroupPositionControllerGC::get_joint_position();
6     VectorNd current_command_array = JointGroupPositionControllerGC::get_joint_command();
7     VectorNd current_velocity_array = JointGroupPositionControllerGC::get_joint_velocity();
8     VectorNd position_error_array = current_command_array - current_position_array;
9
10    bool ke_exceeded = false;
11
12    // Get robot energies
13    JointGroupPositionControllerGC::calcKineticAndPotentialEnergy(KineticEnergy, PotentialEnergy,
          nominalKineticEnergy);
14
15    // If the variable 'set_gains_adaptively' is true, we set the gains adaptively
16    if(set_gains_adaptively == true)
17    {
18      JointGroupPositionControllerGC::set_adaptive_pid_gains(KineticEnergy, PotentialEnergy,
            nominalKineticEnergy, position_error_array, current_velocity_array, overall_L_lim,
            overall_beta, overall_alpha, ke_exceeded);
19    }
20
21    // Initializing variables
22    VectorNd Tau_Gravity = VectorNd::Zero (rbdl_model->qdot_size);
```

```
23      Tau_Gravity =
            JointGroupPositionControllerGC::get_gravity_vector_overall(current_position_array_overall);
24      std::vector<double> & commands = *commands_buffer_.readFromRT();
25      std::vector<double> commands_msg (n_joints_);
26      std::vector<double> error_msg (n_joints_);
27      std::vector<double> generated_torque (n_joints_);
28      std::vector<double> gravity_torque (n_joints_);
29      std::vector<double> pid_p_err (n_joints_);
30      std::vector<double> pid_d_err (n_joints_);
31      std::vector<double> pid_p_torque (n_joints_);
32      std::vector<double> pid_d_torque (n_joints_);
33      std::vector<double> torque_pid_compute (n_joints_);
34
35      // Iterating through the joints
36      for(unsigned int i=0; i<n_joints_; i++)
37      {
38          double command_position = commands[i];
39          commands_msg[i] = command_position;
40          double error; //, vel_error;
41          double commanded_effort;
42          double current_position = current_position_array[i];
43
44          // Make sure joint is within limits if applicable
45          enforceJointLimits(command_position, i);
46
47          // Compute position error
48          if (joint_urdfs_[i]->type == urdf::Joint::REVOLUTE)
49          {
50           angles::shortest_angular_distance_with_limits(
51              current_position,
52              command_position,
53              joint_urdfs_[i]->limits->lower,
54              joint_urdfs_[i]->limits->upper,
55              error);
56          }
```

```
57        else if (joint_urdfs_[i]->type == urdf::Joint::CONTINUOUS)
58        {
59          error = angles::shortest_angular_distance(current_position, command_position);
60        }
61        else //prismatic
62        {
63          error = command_position - current_position;
64        }
65
66        error_msg[i] = error;
67        commanded_effort = pid_controllers_[i].computeCommand(error, period);
68
69        if(activate_joints[i] == 1)
70        {
71          torque_pid_compute[i] = commanded_effort_pid_compute;
72
73          if (add_gravity_comp == true)
74          {
75            commanded_effort_pid_compute = commanded_effort_pid_compute + (grav_comp * Tau_Gravity[i]);
76          }
77
78          generated_torque[i] = commanded_effort_pid_compute;
79          gravity_torque[i] = Tau_Gravity[i];
80
81          if (send_gen_torques_to_robot == true)
82          {
83            // joints_[i].setCommand(commanded_effort_pid_compute);
84            setDesiredJointTorque(joint_names_[i], commanded_effort_pid_compute);
85          }
86          else
87          {
88            setDesiredJointTorque(joint_names_[i], 0.);
89          }
90        }
91        else
```

```
92        {
93          setDesiredJointTorque(joint_names_[i], 0.);
94        }
95    }
96
97    // Gravity compensation is added slowly
98    if (add_gravity_comp == true)
99    {
100     if(grav_comp < 1)
101     {
102       grav_comp += 0.005;
103     }
104   }
105   else
106   {
107       grav_comp = 0;
108   }
109 }
```